CoIoT: Uma Arquitetura Distribuída para IoT Direcionada à Consciência de Contexto das Aplicações Ubíquas

Rodrigo Souza[1], **João Lopes**[1], **Patrícia Davet**[2], **Gizele Gadotti**[2], **Adenauer Yamin**[2], **Cláudio Geyer**[1]

[1] Universidade Federal do Rio Grande do Sul (UFRGS)
Porto Alegre – RS

[2]Universidade Federal de Pelotas (UFPel)
Pelotas – RS

{rssouza, jlblopes, geyer}@inf.ufrgs.br

{ptdavet, gizele.gadotti, adenauer}@inf.ufpel.edu.br

Abstract. *The advent of the Internet of Things (IoT) through increased availability of sensors connected to a means of global communication, which is the Internet, brought new dimension to research in the context-awareness in Ubicomp. In this sense, the main contribution of this paper is the proposition of the CoIoT as an architecture for proactive management of EXEHDA interactions with the physical environment, through rules, using networks of sensors and actuators. To evaluate the functionalities of the proposed architecture, we present a case study in the agriculture area, highlighting the prototypes and tests performed. The results were promising.*

Resumo. *O advento da Internet das Coisas (IoT) através da crescente disponibilidade de sensores conectados a um meio de comunicação global, que é a Internet, trouxe nova dimensão às pesquisas em consciência de contexto na Ubicomp. Neste sentido, a principal contribuição deste artigo consiste na proposição do CoIoT enquanto uma arquitetura para o gerenciamento proativo das interações do EXEHDA com o ambiente físico, através de regras, empregando redes de sensores e atuadores. Para avaliar as funcionalidades da arquitetura proposta, apresentamos um estudo de caso na área da agricultura, com destaque para os protótipos e testes realizados, cujos resultados se mostraram promissores.*

1. Introdução

Na Computação Ubíqua (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 as quais devem ser coletadas em ambientes largamente distribuídos [Knappmeyer et al. 2013]. 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 que constituem fontes geradoras de informações contextuais para as aplicações ubíquas [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 o perfil operacional de baixo nível típico

dos dispositivos da IoT [Perera et al. 2014]. A contribuição central deste artigo visa suprir essa lacuna, através da concepção do CoIoT (C*ontext* + IoT), uma arquitetura de software integrada ao Middleware EXEHDA que tem por objetivo o tratamento de sensores e atuadores. O EXEHDA (*Execution Environment for Highly Distributed Applications*) [Lopes et al. 2012] provê uma arquitetura de software que visa criar e gerenciar um ambiente ubíquo, bem como promover a execução, sob este ambiente, das aplicações da Ubicomp. Seu foco é permitir que as aplicações possam obter informações de seus contextos de interesse e reagir às variações que acontecem nos mesmos.

O CoIoT foi concebido para ser autonômico, gerenciado através de regras e, capaz de agir de forma proativa, tanto na captura de informações contextuais do ambiente físico, como na atuação remota sobre o mesmo.

O artigo está estruturado da seguinte forma: a seção 2 apresenta o modelo proposto para o CoIoT, detalhando sua arquitetura e funcionalidades. A seção 3 destaca o estudo de caso da proposta. Os trabalhos relacionados são apresentados na seção 4. Por fim, na seção 5 são feitas as considerações finais deste trabalho.

2. Concepção e Modelagem da Arquitetura Proposta

Na proposta do CoIoT o ambiente computacional é constituído por células em que se distribuem os dispositivos computacionais, conforme mostra a Figura 1. 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 da IoT.

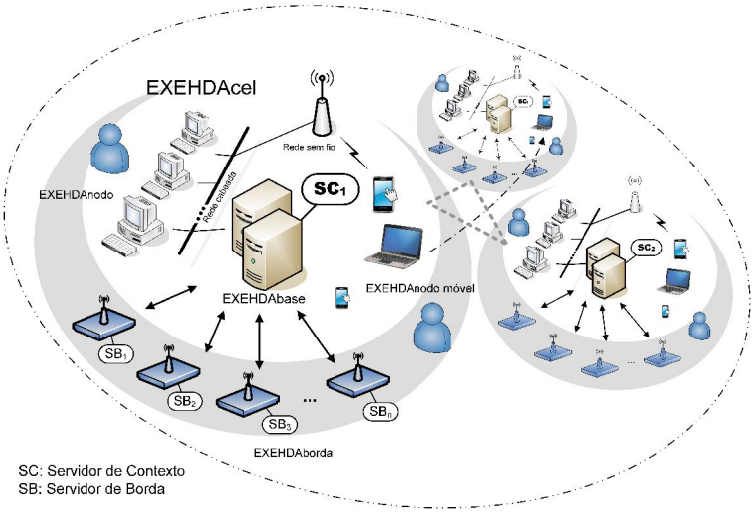


Figura 1. Organização Celular do Ambiente Ubíquo

Na abordagem de gerenciamento de contexto proposta para o EXEHDA, as responsabilidades são organizadas em dois tipos de servidores: Servidor de Contexto e Ser-

vidor de Borda. O Servidor de Borda se destina a gerenciar a interação com o meio físico através de sensores e atuadores. O Servidor de Contexto, por sua vez, atua no armazenamento e processamento das informações contextuais [Lopes et al. 2014].

2.1. Modelo Arquitetural do CoIoT

O CoIoT tem sua arquitetura apresentada na Figura 2 com destaque para o Servidor de Borda. Nessa figura, além dos diversos módulos que constituem o Servidor de Borda está identificada a relação dos mesmos com o Servidor de Contexto e com os dispositivos que interagem com o ambiente físico. O Servidor de Borda é elemento central nos procedimentos de coleta e atuação do CoIoT, tendo seu software instanciado em um equipamento do tipo EXEHDAborda. Sua arquitetura permite gerenciar diferentes dispositivos da IoT, como nodos sensores heterogêneos (sensores programáveis), sensores não programáveis e atuadores.

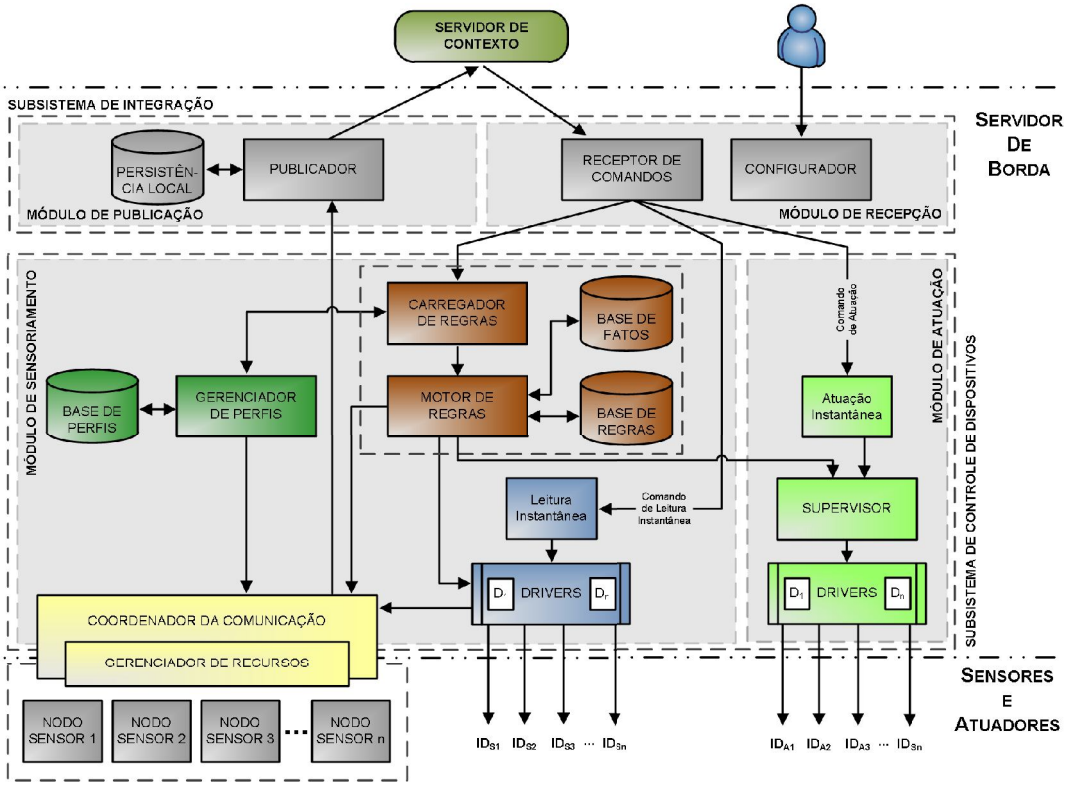


Figura 2. Arquitetura do CoIoT

A arquitetura proposta tem por premissa atuar de forma autonômica uma vez que os dados contextuais continuam a ser coletados e processados mesmo nos períodos nos quais as aplicações interessadas no seu uso estejam inoperantes.

2.2. Operação Distribuída

Com o intuito de atender o perfil inerentemente distribuído das aplicações ubíquas, o Subsistema de Integração da arquitetura do CoIoT foi concebido para promover a interoperação entre o Servidor de Borda e os demais serviços do Middleware em execução

em outros equipamentos. Além disso, através desse subsistema são oferecidos mecanismos para a configuração dos componentes da arquitetura de forma remota. Suas tarefas são operacionalizadas pelos módulos de Publicação e de Recepção. O Módulo de Publicação é formado pelos componentes Publicador e Persistência Local, enquanto o Módulo de Recepção pelo Receptor de Comandos e pelo Configurador.

O *Publicador* consiste no elemento que realiza o envio dos dados para as demais camadas do Middleware, interoperando com a interface de aquisição do Servidor de Contexto. Considerando as possíveis falhas de comunicação entre o Servidor de Borda e o Servidor de Contexto se faz necessário um mecanismo de persistência para garantir o armazenamento temporário dessas informações até que as mesmas sejam publicadas. O elemento da arquitetura desenvolvido para prover esse armazenamento é o componente *Persistência Local*. O componente *Receptor de Comandos*, por sua vez, é responsável pelo recebimento dos comandos e das regras provenientes do Servidor de Contexto bem como pelo encaminhamento dos mesmos aos respectivos componentes da arquitetura.

Todas as configurações necessárias para o funcionamento do Servidor de Borda são operacionalizadas através do componente *Configurador*. Através deste componente é disponibilizada uma interface Web pela qual é possível gerenciar a remoção e inclusão de sensores e atuadores, configurar *drivers* de dispositivos, gerenciar as regras de controle, configurar o endereço do Servidor de Contexto para a publicação dos dados sensoreados, entre outros. Esta interface é utilizada tanto pelo desenvolvedor das aplicações quanto pelo administrador do Middleware considerando as demandas do usuário final e o perfil operacional das aplicações alvo.

2.3. Gerenciamento de Dispositivos Através de Regras

Na perspectiva de atender a elevada heterogeneidade que caracteriza os dispositivos na Ubicomp bem como, os diferentes perfis de interesse das aplicações quanto às interações com o meio, o CoIoT provê suporte à customização no tratamento a nível de sensor e atuador. Para tanto, o Subsistema de Controle de Dispositivos (vide Figura 2) viabiliza a operação dos sensores e atuadores considerando as especificidades de cada dispositivo, através de componentes organizados nos Módulos de Sensoriamento e de Atuação.

A estratégia de gerenciamento adotada no CoIoT teve como base o modelo de regras ECA (*Event-Condition-Action*) [Terfloth 2009], o qual foi concebido para tratar eventos considerando uma especificação do comportamento autonômico esperado por parte da rede de sensores. Estes eventos podem ser produzidos por alterações no tráfego de dados, frequência de aquisição e valores contextuais aquisitados. Todas estas informações, de forma combinada ou não, são tratadas através de regras, a quais são elaboradas considerando as necessidades da aplicação destinada ao usuário final.

O *Motor de Regras* é o componente arquitetural responsável pelo processamento das regras, as quais são armazenadas na Base de Regras. A arquitetura do Servidor de Borda não se restringe a um conjunto fixo de regras. Conforme a necessidade das aplicações ubíquas o conjunto de regras pode ser expandido através da inclusão de novas elementos na Base de Regras e na Base de Perfis. Para servir de suporte ao processamento das regras, o componente *Base de Fatos* da arquitetura tem por função registrar os aspectos necessários às execuções das mesmas.

Para adequar-se ao dinamismo do ambiente que se reflete no comportamento

das aplicações da Ubicomp, na concepção do CoIoT foram utilizados mecanismos de reprogramação dinâmica, os quais realizam a substituição dos processos em execução nos nodos sensores conforme as demandas das aplicações. Na arquitetura proposta, esse processo é realizado de forma coordenada entre os componentes Carregador de Regras, Gerenciador de Perfis e Base de Perfis. Toda regra submetida ao Servidor de Borda é avaliada pelo componente *Carregador de Regras* o qual tem a função de ativar na arquitetura os elementos necessários ao seu processamento. No caso da regra envolver nodos sensores, o Carregador de Regras informa o componente *Gerenciamento de Perfis* quanto aos códigos disponíveis na *Base de Perfis* que devem ser distribuídos aos nodos sensores para dar suporte a tal regra. A partir do momento em que todos os códigos estão em execução nos nodos sensores, o Carregador de Regras submete a nova regra ao Motor de Regras para que este possa processá-la.

2.4. Tratamento da Heterogeneidade

Os *Drivers*, enquanto componentes arquiteturais, são responsáveis pelo acesso aos valores das grandezas físicas medidas pelos sensores bem como pelos comandos de atuação enviados aos atuadores. Eles encapsulam e controlam os sensores e atuadores de maneira individualizada, o que evita que as diferenças operacionais desses dispositivos se projetem nos demais componentes da arquitetura.

O componente *Leitura Instantânea* tem o objetivo de permitir a leitura de um determinado sensor sob demanda das aplicações a qualquer momento. O componente faz a recepção assíncrona das solicitações e, a partir do ID do sensor, dispara o *driver* correspondente.

Pela natureza dos nodos sensores típicos da IoT, é esperado que sejam executadas atividades colaborativas entre os processos em execução nestes nodos [Mottola and Picco 2011][Perera et al. 2014]. Por isso torna-se importante a existência de um mecanismo que simplifique a comunicação entre os nodos sensores e destes com o Servidor de Borda. Na arquitetura do CoIoT, essa funcionalidade é operacionalizada através do componente *Coordenador da Comunicação*. Este componente implementa um modelo de coordenação baseado na abstração de Espaço de Tuplas [Souza et al. 2012] que simplifica a programação das aplicações, abstraindo aspectos de baixo nível associados com a troca de informações entre os dispositivos envolvidos.

Em redes de sensores 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 [Mottola and Picco 2011]. Neste sentido, o componente *Gerenciador de Recursos* tem a função de administrar esses eventos, mantendo a consistência da infraestrutura da rede como um todo.

O Módulo de Atuação do Servidor de Borda, cuja estrutura é apresentada na Figura 3, agrupa os componentes que fazem o controle dos atuadores. O componente *Atuação Instantânea* tem um funcionamento análogo ao componente de Leitura Instantânea. Ele recebe comandos com o ID do atuador e os correspondentes padrões de operação (tempo de duração, potência de ativação, etc.), os quais são repassados ao componente Supervisor para tratamento.

O componente *Supervisor* aglutina os comandos de atuação. Uma vez recebidos

os parâmetros para controle da atuação, o componente Supervisor, após avaliar eventuais conflitos entre as regras oriundas das diferentes fontes, ativa o respectivo *driver* do atuador envolvido.

3. Estudo de Caso

Esta seção resume os principais aspectos do estudo de caso realizado para avaliar as funcionalidades do CoIoT. Segundo [Knappmeyer et al. 2013], a transparência introduzida através uso de middlewares em aplicações conscientes do contexto introduz uma dificuldade quanto à avaliação das funcionalidades propostas. Para o autor, uma das principais estratégias utilizadas para avaliar a capacidade do middleware é através de aplicações experimenteis, nas quais as experiências dos usuários podem ser exploradas de maneira explicita ou implícita. Nesse sentido, o estudo de caso realizado para avaliar a proposta do CoIoT procurou atender as demandas do Projeto AMPLUS (*Automatic Monitoring and Programable Logging Ubiquitous System*). O AMPLUS foi desenvolvido com o objetivo de promover soluções de Ubicomp para o Laboratório Didático de Análise de Sementes (LDAS) da Faculdade de Agronomia da Universidade Federal de Pelotas. O LDAS integra o Programa de Pós Graduação em Ciência e Tecnologia de Sementes do Departamento de Fitotecnia, sendo utilizado em pesquisas e atividades de pós-graduação.

Para que as análises realizadas no LDAS sejam válidas, são necessários equipamentos adequados e a aplicação de métodos padronizados que especificam procedimentos uniformes que devem ser seguidos [Brasil 2009]. Porém, grande parte dos equipamentos do Laboratório possuem volume interno relativamente pequeno. Os germinadores BOD utilizados neste estudo de caso, por exemplo, têm, em média, 340 litros, portanto se existir uma grande diferença entre o ambiente interno e externo, as condições internas variam rapidamente devido a pouca inércia térmica. Isso exige, entre outros aspectos, o monitoramento e controle da temperatura e umidade das sementes ao longo de todo período de análise e ação rápida caso algum dos valores saia das faixas especificadas. Nessa perspectiva, o Projeto AMPLUS mostrou-se oportuno para avaliar características importantes do CoIoT. Os aspectos avaliados nesse cenário foram: (*i*) coleta de dados do ambiente; (*ii*) atuação pró-ativa sobre o meio; e (*iii*) publicação dos dados sensoriados no Servidor de Contexto.

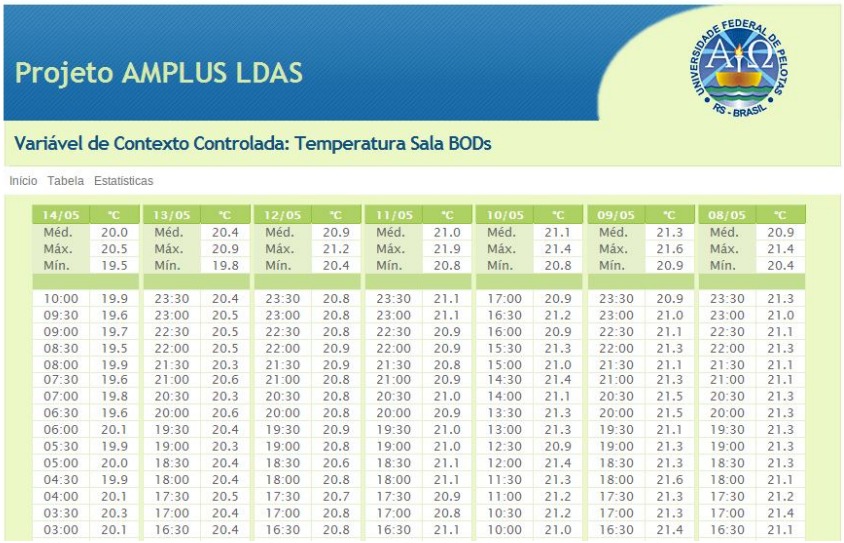
A fim de avaliar as funcionalidades da arquitetura proposta optou-se por utilizar no LDAS um conjunto de dispositivos constituído por nodos sensores e por sensores não programáveis. Os sensores não programáveis selecionados para o estudo de caso são baseados na tecnologia 1-Wire[1]. Essa tecnologia caracteriza-se como uma rede de transmissão de dados, baseada em dispositivos eletrônicos endereçáveis, e tem se destacado por sua versatilidade e facilidade de implementação. Por sua vez, os nodos sensores adotados nos testes são do tipo Telos, revisão B com sistema operacional Contiki [2], os quais têm sido amplamente utilizados em pesquisas envolvendo redes de sensores. O Telos faz parte da linha de *motes* desenvolvida pela Universidade da Califórnia, Berkeley, e vem sendo fabricado pela empresa Memsic [3]. Assim, foram utilizados no LDAS 15 sensores 1-Wire e 3 nodos sensores Telos B. Para explorar a característica reativa da arquitetura,

[1]http://www.maximintegrated.com
[2]http://www.contiki-os.org/
[3]http://www.memsic.com/

também foi utilizado um atuador baseado na tecnologia 1-wire, na forma de um alerta luminoso, que é acionado quando há a necessidade da atenção dos laboratoristas para com algum dos equipamentos.

O protótipo do CoIoT foi escrito em Phyton sobre o Sistema Operacional Raspbian, sendo usado como hardware a Raspberry PI [4]. A tecnologia central utilizada na implementação do Tratador de Regras e do Publicador foi XML-RPC[5]. O Motor de Regras executa sequencialmente as regras especificadas. A leitura de um dado sensor é realizada sempre que uma das condições definidas nas regras é satisfeita. Esta leitura é efetivada por *drivers* específicos que tratam cada sensor individualmente, segundo as particularidades tecnológicas de cada um. As regras de gerenciamento do Servidor de Borda, que controlam a leitura dos sensores, a publicação dos seus valores no Servidor de Contexto bem como a supervisão dos atuadores, também são escritas em Phyton.

Nesse cenário de testes foram utilizadas regras que tratam condições que contemplam dois critérios distintos: (*i*) critério de tempo, em que a ação é disparada em função da passagem de um tempo especificado, o qual é usado para publicar dados contextuais periodicamente para registro histórico; (*ii*) critério de valor, em que uma ação é disparada quando um contexto de interesse (temperatura) extrapola uma determinada faixa de valores, sendo que nesse caso a ação consiste no acionamento de um alerta luminoso seguido da publicação do valor sensoreado e o envio de e-mail ou SMS.


Projeto AMPLUS LDAS

Variável de Contexto Controlada: Temperatura Sala BODs

Início Tabela Estatísticas

14/05	°C	13/05	°C	12/05	°C	11/05	°C	10/05	°C	09/05	°C	08/05	°C
Méd.	20.0	Méd.	20.4	Méd.	20.9	Méd.	21.0	Méd.	21.1	Méd.	21.3	Méd.	20.9
Máx.	20.5	Máx.	20.9	Máx.	21.2	Máx.	21.9	Máx.	21.4	Máx.	21.6	Máx.	21.4
Mín.	19.5	Mín.	19.8	Mín.	20.4	Mín.	20.8	Mín.	20.8	Mín.	20.9	Mín.	20.4
10:00	19.9	23:30	20.4	23:30	20.8	23:30	21.1	17:00	20.9	23:30	20.9	23:30	21.3
09:30	19.6	23:00	20.5	23:00	20.8	23:00	21.1	16:30	21.2	23:00	21.0	23:00	21.0
09:00	19.7	22:30	20.5	22:30	20.8	22:30	20.9	16:00	20.9	22:30	21.1	22:30	21.1
08:30	19.5	22:00	20.5	22:00	20.9	22:00	20.9	15:30	21.3	22:00	21.3	22:00	21.3
08:00	19.9	21:30	20.3	21:30	20.9	21:30	20.8	15:00	21.0	21:30	21.1	21:30	21.1
07:30	19.6	21:00	20.6	21:00	20.8	21:00	20.9	14:30	21.4	21:00	21.3	21:00	21.1
07:00	19.8	20:30	20.3	20:30	20.8	20:30	21.0	14:00	21.1	20:30	21.5	20:30	21.3
06:30	19.6	20:00	20.6	20:00	20.8	20:00	20.9	13:30	21.3	20:00	21.5	20:00	21.3
06:00	20.1	19:30	20.4	19:30	20.9	19:30	21.0	13:00	21.3	19:30	21.1	19:30	21.3
05:30	19.9	19:00	20.3	19:00	20.8	19:00	21.0	12:30	20.9	19:00	21.3	19:00	21.3
05:00	20.0	18:30	20.4	18:30	20.6	18:30	21.1	12:00	21.4	18:30	21.3	18:30	21.3
04:30	19.9	18:00	20.4	18:00	20.8	18:00	21.1	11:30	21.3	18:00	21.6	18:00	21.1
04:00	20.1	17:30	20.5	17:30	20.7	17:30	20.9	11:00	21.2	17:30	21.3	17:30	21.2
03:30	20.3	17:00	20.4	17:00	20.8	17:00	20.8	10:30	21.2	17:00	21.3	17:00	21.4
03:00	20.1	16:30	20.4	16:30	20.8	16:30	21.1	10:00	21.0	16:30	21.4	16:30	21.1

Figura 3. Projeto AMPLUS: Relatório Textual dos Dados Sensoriados

Para acompanhar o comportamento da arquitetura, foi desenvolvida uma ferramenta para a visualização dos valores das variáveis físicas coletados pelos sensores gerenciados pelo Servidor de Borda. As funcionalidades concebidas para a ferramenta foram definidas a partir das necessidades operacionais informadas pelos pesquisadores do LDAS. Através da ferramenta desenvolvida o pesquisador passou a ter acesso à visualização das variações dos valores de temperatura e umidade ocorridas durante os

[4]http://www.raspberrypi.org
[5]http://www.xmlrpc.com

períodos de análise, os quais influenciam diretamente nos resultados dos processos de germinação das sementes.

A interface da ferramenta possibilita a seleção do contexto de interesse a ser exibido, que pode ser apresentado na forma de um relatório textual (vide Figura 3) ou através de um modo gráfico (vide Figura 4).

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

A comparação dos contextos de interesse registrados, visualizados através da ferramenta desenvolvida, com os demais eventos produzidos pelo protótipo do Servidor de Borda (acionamento do alerta luminoso, envio de e-mail e SMS recebidos) possibilitou avaliar o comportamento da arquitetura. Durante o período analisado, considerando as definições estabelecidas nas regras utilizadas, foi possível observar que todos os eventos monitorados do ambiente tem sido identificados pela arquitetura.

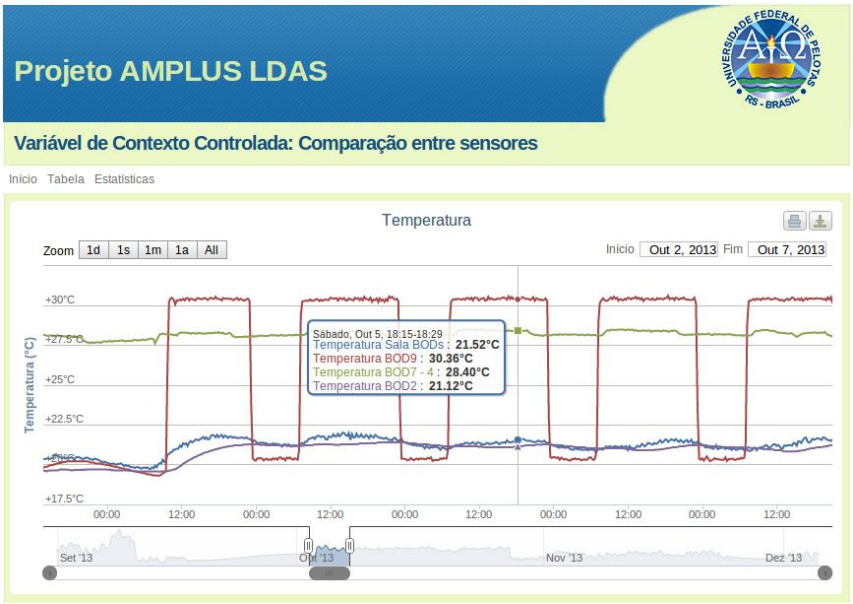


Figura 4. Projeto AMPLUS: Visualização Gráfica dos Dados Sensoriados

4. Trabalhos Relacionados

Tendo como base as premissas que motivaram o desenvolvimento da arquitetura proposta nesse artigo, o estudo da literatura da área proporcionou identificar alguns trabalhos relacionados, dentre os quais foram selecionados os seguintes: CARE [Agostini et al. 2009], CoCA [Ejigu et al. 2008], HiCon [Cho et al. 2008], Solar [Chen et al. 2008], WComp [Tigli et al. 2009]. Os aspectos considerados importantes na seleção de tais trabalhos foram: (*i*) suporte a redes de sensores e atuadores; (*ii*)

aquisição autonômica dos dados de contexto; (*iii*) suporte ao tratamento de regras; e (*iv*) suporte a atuação distribuída sobre o meio.

A arquitetura proposta para o CoIoT foi concebida de forma a gerenciar redes de sensores e atuadores. Com isso, pode ser otimizado o gerenciamento tanto da aquisição dos dados de contexto a partir de vários tipos de sensores, usual na IoT e nos ambientes computacionais para provimento de aplicações ubíquas, como na atuação distribuía sobre o meio físico. Tal característica é encontrada, em parte, nos projetos CoCA e HiCon, que têm suporte a redes de sensores. O projeto WComp, por sua vez, permite atuação sobre o meio, entretanto, não suporta o gerenciamento de redes de atuadores.

Com exceção dos projetos CARE e Solar, os demais preveem o emprego de mecanismos específicos para aquisição do contexto. Estes mecanismos adotam uma estratégia de separação entre a obtenção e o uso do contexto. Essa estratégia também é adotada no trabalho apresentado neste artigo, pois a arquitetura proposta para o CoIoT gerencia, através do Servidor de Borda, a coleta e atuação sobre o meio físico de maneira integrada ao Servidor de Contexto. Porém, no CoIoT, as tarefas de coleta e atuação são auto geridas. Desta maneira, proporciona outro diferencial em relação aos projetos relacionados, pois atua de forma autonômica na aquisição dos dados de contexto, ou seja, de forma independente das aplicações interessadas.

O suporte ao tratamento de regras é encontrado na maioria dos trabalhos identificados na literatura, porém a distribuição deste tratamento entre os Servidores de Contexto e de Borda é um diferencial em relação aos demais projetos. Enquanto no Servidor de Contexto são tratadas regras elaboradas que fazem o cruzamento de diferentes informações, incluindo dados históricos, os quais demandam maior poder computacional, o Servidor de Borda trata as regras de contingência. Esta funcionalidade de processamento de contexto, nos trabalhos relacionados, usualmente está restrita a um único equipamento.

5. Considerações Finais

Este artigo resume os esforços de pesquisa associados à concepção do CoIoT, uma arquitetura para IoT integrada ao Middleware EXEHDA que gerencia a coleta e pré-tratamento das informações contextuais com suporte à respectiva atuação sobre o meio. A arquitetura proposta é baseada em regras e atua de forma proativa em relação aos eventos dos contextos de interesse.

A principal contribuição deste trabalho é a proposição de uma arquitetura para a coleta e atuação, que possibilita gerenciar de maneira autonômica e através de regras, dispositivos da IoT de diferentes naturezas (sensores não programáveis, nodos sensores e atuadores) e tecnologias (de hardware, software básico e comunicação). A estratégia adotada para o CoIoT ampliou o escopo de uso do Middleware EXEHDA possibilitando sua utilização em diferentes cenários.

Dentre outros, na continuidade da pesquisa, os seguintes aspectos deverão ser considerados em trabalhos futuros: (*i*) ampliar o uso do CoIoT no LDAS através do monitoramento dos demais equipamentos do laboratório e consequentemente incorporando outros tipos de sensores e atuadores; (*ii*) explorar o emprego de regras de processamento contextual que utilizem outros mecanismos de inferência de mais alto nível, ampliando as opções de inferência sobre os dados coletados; e (*iii*) dar continuidade aos procedimen-

tos de integração do CoIoT com os diferentes serviços e funcionalidades do Middleware EXEHDA.

Referências

Agostini, A., Bettini, C., and Riboni, D. (2009). Hybrid reasoning in the CARE middleware for context awareness. *International Journal of Web Engineering and Technology*, 5(1):3–23.

Brasil (2009). *Regras para análise de sementes*. Ministério da Agricultura, Pecuária e Abastecimento. Secretaria de Defesa Agropecuária, Brasília.

Chen, G., Li, M., and Kotz, D. (2008). Data-centric middleware for context-aware pervasive computing. *Pervasive and Mobile Computing*, 4(2):216–253.

Cho, K., Hwang, I., Kang, S., Kim, B., Lee, J., Lee, S., Park, S., Song, J., and Rhee, Y. (2008). HiCon: a hierarchical context monitoring and composition framework for next-generation context-aware services. *IEEE Network*, 22(4):34–42.

Ejigu, D., Scuturici, M., and Brunie, L. (2008). Hybrid Approach to Collaborative Context-Aware Service Platform for Pervasive Computing. *Journal of Computers*, 3(1):40–50.

Knappmeyer, M., Kiani, S. L., Reetz, E. S., Baker, N., and Tonjes, R. (2013). Survey of Context Provisioning Middleware. *IEEE Communications Surveys & Tutorials*, 15(3):1492–1519.

Lopes, J. a. L., de Souza, R. S., Pernas, A. M., Yamim, A., and Geyer, C. (2014). A Distributed Architecture for Supporting Context-Aware Applications in UbiComp. In *IEEE International Conference on Advanced Information Networking and Applications (AINA)*, Victória, Canada.

Lopes, J. a. L., Souza, R. S., Geyer, C. R., Costa, C. A., Barbosa, J. V., Gusmão, M. Z., and Yamin, A. C. (2012). A model for context awareness in Ubicomp. In *Proceedings of the 18th Brazilian symposium on Multimedia and the web - WebMedia '12*, page 161, New York, New York, USA. ACM Press.

Mottola, L. and Picco, G. P. (2011). Programming Wireless Sensor Networks: Fundamental Concepts and State of the Art. *ACM Computing Surveys*, 43(3):1–51.

Perera, C., Zaslavsky, A., Christen, P., and Georgakopoulos, D. (2014). Context Aware Computing for The Internet of Things: A Survey. *IEEE Communications Surveys & Tutorials*, 16(1):414–454.

Souza, R. S. D., Lopes, J. a. L. B., Gadotti, G. I., Yamin, A., and Geyer, C. (2012). Um Modelo de Coordenação Escalável e Proativo para Aplicações Ubíquas. In *Simpósio Brasileiro de Computação Ubíqua e Pervasiva (SBCUP)*.

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

Tigli, J.-Y., Lavirotte, S., Rey, G., Hourdin, V., Cheung-Foo-Wo, D., Callegari, E., and Riveill, M. (2009). WComp middleware for ubiquitous computing: Aspects and composite event-based Web services. *annals of telecommunications - annales des télécommunications*, 64(3-4):197–214.