Uma arquitetura para IoT direcionada à ciência do contexto baseada em eventos distribuídos

Rodrigo Souza[1,4], João Lopes[1,4], Anderson Cardozo[3], Tainã Carvalho[2], Patrícia Davet[2], Alexandre Wolf[5], Adenauer Yamin[2,3], Jorge Barbosa[5], Cláudio Geyer[1]

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

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

[3]Universidade Católica de Pelotas (UCPel) – Pelotas – RS – Brasil

[4]Instituto Federal Sul-rio-grandense (IFSUL) – Pelotas – RS – Brasil

[5]Universidade do Vale do Rio dos Sinos (Unisinos) – São Leopoldo – RS – Brasil

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

{ptdavet, trcarvalho}@inf.ufpel.edu.br

{anderson.cardozo, adenauer}@ucpel.edu.br

{awolf, jbarbosa}@unisinos.br

Abstract. *The recent advances in the Internet of Things (IoT) area, which has provided an increasing availability of networked sensors and actuators, has given a new perspective to research in the context awareness in UbiComp. In this sense, the main contribution of this paper is the proposition of* COIOT*, a distributed architecture for IoT designed with the aim at providing, through rules, the proactive management of the EXEHDA Middleware interactions with the physical environment. To evaluate the functionalities of the proposed architecture we implemented a case study in the agricultural area. The achieved results are promising.*

Resumo. *Os avanços recentes no campo de Internet das Coisas (IoT - Internet of Things), que têm promovido uma disponibilidade cada vez maior de sensores e atuadores conectados em rede, trouxe uma nova perspectiva às pesquisa em ciência de contexto na Computação Ubíqua (Ubicomp). Neste sentido, a principal contribuição deste artigo é a proposta do* COIOT*, uma arquitetura distribuída e orientada a eventos para a IoT construída com o objetivo de prover, através de regras, o gerenciamento proativo das interações do meio físico com Middleware EXEHDA. Para validar as funcionalidades da arquitetura proposta foi implementado um estudo de caso na área da agricultura. Os resultados obtidos foram promissores.*

1. Introdução

Na Computação Ubíqua os sistemas computacionais devem ser capazes de reagir às mudanças de estado das diferentes variáveis contextuais de seu interesse. Essas variáveis contextuais devem ser coletadas em ambientes altamente distribuídos [Knappmeyer et al. 2013]. Por sua vez, os avanços científicos e tecnológicos

recentes no campo de IoT têm viabilizado a utilização de sensores em larga escala, os quais constituem fontes geradoras de informações contextuais para as aplicações ubíquas cientes de contexto [Perera et al. 2014].

Diversos desafios de pesquisa relacionados ao uso da IoT na obtenção de informações contextuais são associados com as diferenças entre os requisitos de alto-nível das aplicações ubíquas e as tarefas de gerenciamento dos dispositivos da IoT, as quais são relacionadas com as características eletrônicas envolvidas [Perera et al. 2014].

A principal contribuição deste artigo consiste em preencher essa lacuna através da proposição do COIOT (COntext + IOT), uma arquitetura integrada ao *Middleware* EXEHDA capaz de prover suporte ao tratamento de sensores e atuadores. O EXEHDA *(*Execution Environment for Highly Distributed Applications) [Lopes et al. 2014a] provê uma arquitetura de software baseada em serviços que visa a criação e gerenciamento de um ambiente ubíquo, bem como a execução de aplicações sobre este ambiente.

O COIOT [Souza et al. 2015] é uma arquitetura baseada em eventos, gerenciada por regras, com processamento distribuído de contexto, capaz de agir proativamente na coleta das informações contextuais do ambiente físico, bem como atuar remotamente sobre o mesmo.

Este artigo está organizado da seguinte forma: A seção 2 apresenta a modelagem do COIOT, detalhando a sua arquitetura e funcionalidades. Na seção 3, é apresentado o protótipo e os testes realizados na área da agricultura. Os trabalhos relacionados são apresentados na seção 4. Por fim, na seção 5, são realizadas as considerações finais deste artigo.

2. COIOT: concepção e modelagem

A abordagem de tratamento de contexto proposta para o COIOT tem suas funcionalidades distribuídas entre dois tipos de servidores: Servidor de Contexto e Servidor de Borda. O Servidor de Borda foi concebido para atuar, principalmente, no gerenciamento das interações com o ambiente físico. Por sua vez, o Servidor de Contexto atua no armazenamento e processamento de informação contextuais [Lopes et al. 2014b].

2.1. Modelo arquitetural proposto

A arquitetura proposta para o COIOT, apresentada na Figura 1, foi concebida com o objetivo de gerenciar diferentes dispositivos da IoT, como sensores e atuadores heterogêneos. Esta arquitetura tem por premissa atuar de forma autônoma, tanto na coleta e processamento das informações contextuais, quanto na atuação sobre o ambiente, uma vez que essas atividades continuam a serem executadas mesmo nos períodos nos quais as aplicações interessadas no seu uso estejam inoperantes.

O processamento do contexto no COIOT se dá de forma distribuída entre os Servidores de Borda e Contexto. O módulo *Motor de Regras* (Servidor de Borda) constitui o primeiro nível de processamento, enquanto o *Processador de Contexto* (Servidor de Contexto) o segundo nível. As regras submetidas ao Motor de Regras devem ser elaboradas de forma a atender, prioritariamente, os eventos críticos, cujo tratamento deve ser realizado no menor tempo possível e com mínimo de falhas. Isso se deve ao fato de que o Servidor de Borda é geralmente alocado fisicamente próximo ao ambiente monitorado,

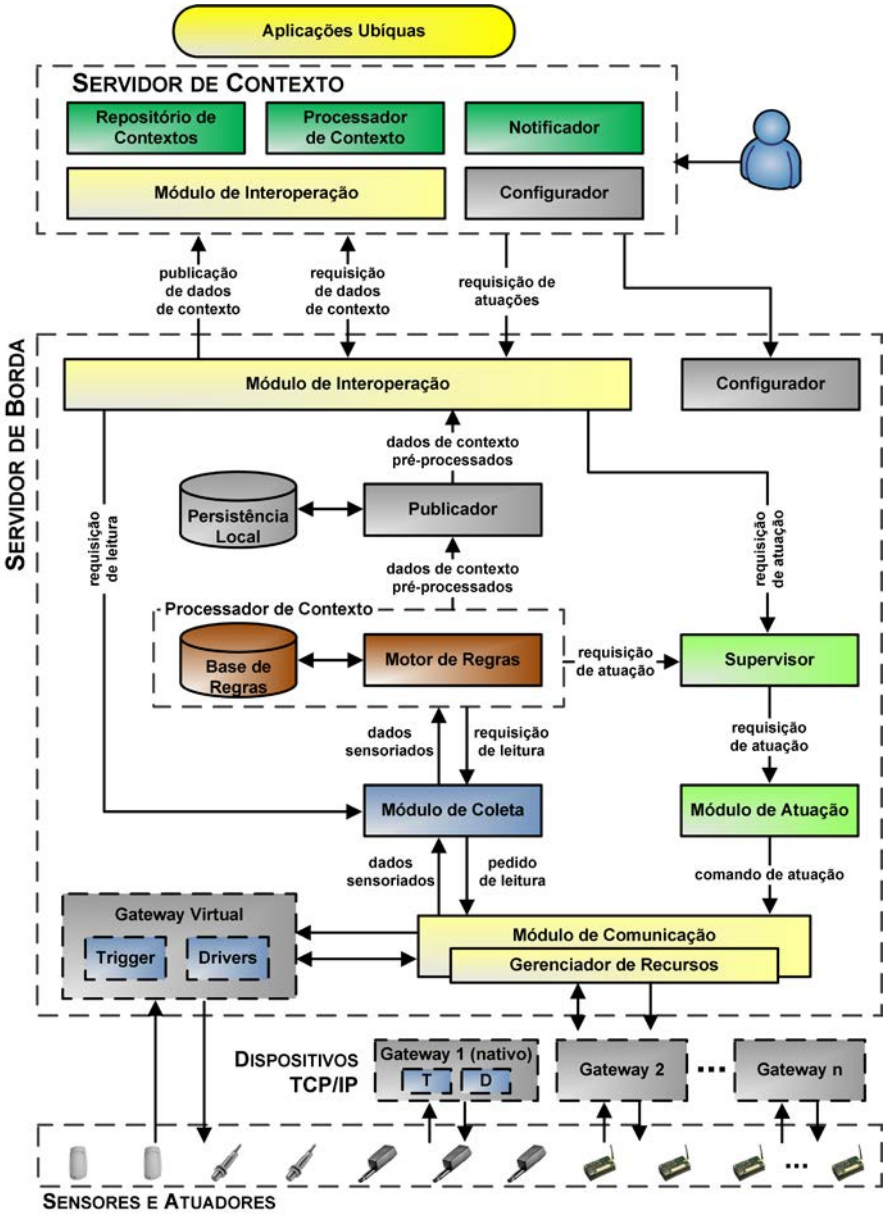

Figura 1. Arquitetura do CoIoT

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 coletados de outros Servidores de Borda, 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 podem ser disparadas a partir de eventos produzidos pelo ambiente. 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.

O módulo *Repositório de Contextos* utiliza um modelo relacional para a representação das informações contextuais, o qual proporciona um registro histórico desses dados. A estrutura do Repositório de Contexto reflete a organização da arquitetura do *Middleware* EXEHDA, contemplando assim o relacionamento entre aplicações, componentes, sensores, ambientes e contextos de interesse. O repositório também armazena a tabela de dados de configuração da arquitetura e as publicações dos sensores existentes no ambiente ubíquo. Estes dados são utilizados pelo módulo Processador de Contexto para disparar as ações apropriadas dependendo das informações contextuais.

Considerando a característica inerentemente distribuída das aplicações ubíquas, os *Módulos de Interoperação* do CoIoT foram concebidos para promover a interoperabilidade entre os Servidores de Borda e de Contexto, assim como com outros serviços do *Middleware*. A concepção deste módulo teve como referência o estilo arquitetural REST [Fielding 2000].

O *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*, em que recebe subscrições de todos os serviços e/ou aplicações que requerem notificações a respeito das mudanças nos estados dos contexto.

Todas as configurações necessárias para o funcionamento do CoIoT são gerenciadas através de uma interface web disponibilizada pelo módulo *Configurador*. Entre as funcionalidades oferecidas tem-se: a configuração de sensores e atuadores (inclusão, remoção e alteração), o gerenciamento de *drivers* de dispositivos, gerenciamento das regras de processamento de contexto, configuração de acesso aos Servidores de Contexto e Borda, entre outros.

O módulo *Publicador* 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.

Com o intuito de garantir a interoperabilidade com tecnologias de mercado, e também potencializar a distribuição das iniciativas de coleta e/ou atuação, foram utilizados *gateways* de dois tipos: (i) Gateway proprietário, que tem funcionalidades heterogêneas variando de acordo com seus fabricantes, e o; (ii) Gateway nativo cujas funcionalidades operam de maneira integradas à arquitetura do CoIoT. O módulo *Gateway Virtual* age como uma virtualização do Gateway nativo e implementa dois tipos de módulos básicos: *Drivers* e *Triggers*. Drivers são módulos arquiteturais responsáveis pelo acesso aos valores das grandezas físicas capturadas pelos sensores, bem como pelas execuções de comandos enviados para os atuadores. Os Drivers encapsulam e controlam os sensores e atuadores de uma maneira individualizada, evitando que diferenças operacionais destes dispositivos se propaguem a outros componentes da arquitetura. O Trigger gerencia a leitura de sensores através de eventos, tendo sido concebido para lidar com os

dois principais tipos de eventos a serem tratados na IoT: eventos temporais e eventos do ambiente [Perera et al. 2014].

O *Módulo de Comunicação* e o *Gerenciador de Recursos* foram concebidos para gerenciar os aspectos associadas à comunicação entre os *Gateways* e o Servidor de Borda. O Módulo de Comunicação administra as comunicações através de API REST enquanto o Gerenciador de Recursos provê um mecanismo de descoberta que gerencia a entrada e saída de dispositivos na rede, ocorrências típicas da IoT.

O *Módulo de Coleta* tem a função de direcionar as solicitações de coleta aos respectivos *gateways* sob demanda do Motor de Regras, do Servidor de Contexto, ou das aplicações. O módulo *Supervisor* aglutina os comandos de atuação, recebendo os parâmetros de controle e resolvendo eventuais conflitos entre as requisições de oriundas de diferentes fontes. O *Módulo de Atuação* tem uma função semelhante a do Módulo de Coleta, recebendo os comandos de atuação e seus parâmetros operacionais (duração, energia de ativação, etc.) e os encaminhando aos *gateways* correspondentes para processamento adicional.

2.2. Suporte ao tratamento de eventos proposto

Na Internet das Coisas, eventos do ambiente ocorrem 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. Esses eventos 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].

Ambientes da IoT geram potencialmente uma grande quantidade de eventos que devem ser gerenciados pela arquitetura de suporte. O gerenciamento desses eventos por parte da arquitetura possibilita que os mesmos possam ser tratados no momento em que eles acontecem, permitindo uma reação rápida quando necessário [Razzaque et al. 2016].

Eventos são frequentemente identificados como primitivos (discretos) ou complexos (compostos). Um evento primitivo refere-se a uma ocorrência instantânea, atômica de um acontecimento de interesse em um determinado instante de tempo. Enquanto que um evento complexo (também chamado de evento composto) consiste na combinação de eventos primitivos ocorridos em um determinado intervalo de tempo [Terfloth 2009]. O CoIoT provê suporte tanto a eventos primitivos quanto a eventos complexos, os quais podem ser utilizados para disparar regras ECA.

O modelo de tratamento de eventos proposto para o CoIoT considera um conjunto de eventos primitivos gerados a partir de: (i) mudanças de estado dos contextos de interesse coletados através de sensores; (ii) ativação/desativação de atuadores; e (iii) alterações na infraestrutura do ambiente computacional. Estes eventos são apresentados na Tabela 1. O suporte a eventos complexos é provido pelo CoIoT a partir da composição de eventos através lógicas condicionais tratadas pelas regras ECA e processadas pelos Processadores de Contexto.

3. CoIoT: prototipação e testes

Esta seção resume os principais aspectos de prototipação e testes realizados através do projeto AMPLUS (*Automatic Monitoring and Programmable Logging Ubiquitous Sys-*

Tabela 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

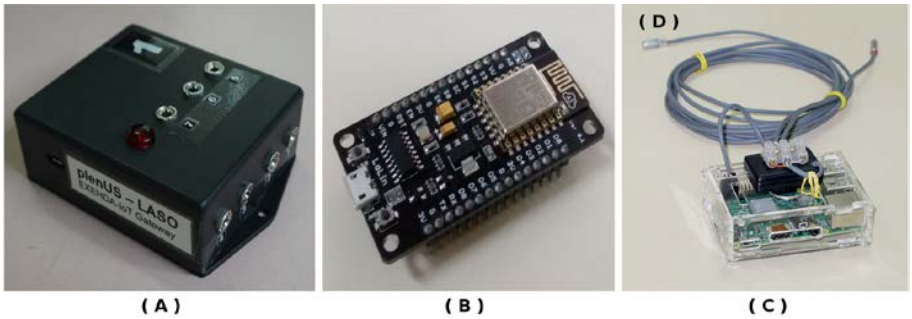

Figura 2. (A) Gateway Nativo; (B) NodeMCU; (C) Raspberry PI; (D) Sensor de Temperatura DS18B20

tem) usados para avaliar as funcionalidades do CoIoT. O estudo de caso inclui tarefas relacionadas com o sensoriamento, coleta, processamento e notificação de contexto. Neste estudo de caso foi desenvolvida uma ferramenta para avaliar as principais funcionalidades da arquitetura proposta.

O projeto AMPLUS foi concebido para fornecer serviços móveis e cientes do contexto que permitem o armazenamento dos estados contextuais que caracterizam os equipamentos do Laboratório Didático de Análise de Sementes (LDAS - http://amplus.ufpel.edu.br/ldas), a geração de notificações e atuações quando necessário.

3.1. Infraestrutura de hardware

Para avaliar as funcionalidades do CoIoT optou-se por utilizar no LDAS um conjunto de dispositivos que consistem de um Gateway nativo, 15 sensores e um atuador. Os sensores selecionados para este estudo de caso são baseados na tecnologia de 1-Wire (http://www.maximintegrated.com). Esta tecnologia é caracterizada como uma rede de transmissão de dados com base em dispositivos eletrônicos endereçáveis, e se destaca pela sua versatilidade e a facilidade de implementação. Os sensores de temperatura utilizados podem ser vistos na Figura 2(D). Este sensor é envolto em um invólucro de alumínio para dar maior resistência mecânica e isolar da umidade. O Gateway Nativo (Figura 2 (A)) foi desenvolvido utilizando NodeMCU o qual permite a conexão de até sete dispositivos 1-wire. NodeMCU (http://nodemcu.com/) é uma plataforma de código aberto para desenvolvimento de dispositivos IoT (vide Figura 2 (B)). Para explorar a característica reativa da arquitetura, também foi utilizado um atuador (alerta luminoso) com base na tecnologia de 1-Wire. Este atuador é acionado quando é necessária a atenção dos trabalhadores do laboratório com alguns equipamentos.

3.2. Infraestrutura de software: principais características

A maior parte do protótipo do CoIoT foi escrita em Python no sistema operacional Raspbian. O hardware utilizado no Servidor de Borda é o Raspberry Pi (http://www.raspberrypi.org) (vide Figura 2 (C)). O Servidor de Contexto foi instalado em um hardware com processador Intel E3400-2.6GHz dual-core com 4GB de memória, com o Linux Ubuntu Server como sistema operacional. A tecnologia usada na implementação do Motor de Regras e do Processador de Contexto é o Drools (http://www.drools.org/). A leitura dos sensores é feita através de drivers específicos que fazem um tratamento individualizado dos dispositivos de acordo com as características tecnológicas de cada um. O Módulo de Interoperação foi implementado através do Sails.js (http://sailsjs.org/), um framework MVC (MVC) para Node.js (https://nodejs.org). A API REST desenvolvida fornece recursos para lidar com os sensores e atuadores, bem como para realizar a publicação dos dados coletados. Os dados enviados através das operações REST são estruturados em JSON.

3.3. Infraestrutura de software: soluções desenvolvidas para o LDAS

O suporte do CoIoT à operação do cenário de avaliação é provido através de *triggers* de leitura dos sensores e de um conjunto de regras de processamento de contexto. Os *triggers* são utilizados para gerenciar as leituras dos sensores de temperatura das Incubadoras BODs (*Biochemical oxygen demand*) em duas situações: (i) em intervalos de tempo regulares; e (ii) quando o valor está fora da faixa especificada. As Incubadoras BODs são utilizadas no LDAS para realizar testes de germinação de sementes, os quais exigem precisão quanto aos limites específicos para variação da temperatura.

As regras de processamento de contexto utilizadas foram organizadas entre os Servidores de Borda e Servidor de 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. As regras utilizadas são apresentadas nas Tabelas 2 e 3.

A ferramenta desenvolvida possibilita a seleção do contexto de interesse a ser exibido, que pode ser apresentado na forma de um relatório textual ou através de um modo gráfico. Através da ferramenta o pesquisador do LDAS pode a ter acesso à visualização das variações dos valores de temperatura e umidade ocorridas nas Incubadoras BODs durante os períodos de análise, os quais influenciam diretamente nos resultados dos processos de germinação das sementes.

O relatório gráfico desenvolvido permite visualizar simultaneamente as curvas de variação dos valores de vários sensores utilizados no LDAS (vide Figura 3). 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 sensoriados.

De forma a promover a proatividade do Projeto AMPLUS com a comunidade de usuários, foram desenvolvidas interfaces para serviços de comunicação: e-mail e SMS para a rede celular. O Servidor de Contexto produz essas mensagens a partir do processamento de regras de contexto de forma autônoma.

Tabela 2. Regras do Servidor de Borda

Nome da Regra	Evento	Condição	Ação
Lê temp. BOD	Publication	Se valor fora do especificado	Aciona alerta luminoso
Publica temp. BOD	Publication	-	Publica temperatura no Servidor de Contexto

Tabela 3. Regras do Servidor de Contexto

Nome da Regra	Evento	Condição	Ação
Lê temp. BOD	Publication	Se valor fora do especificado	Notifica o usuário (e-mail/SMS)
Persiste temp. BOD	Publication	-	Persiste a temperatura no Repositório de Contexto

Figura 3. Relatório Gráfico

A rotina dos laboratoristas implica na mobilidade por diferentes ambientes físicos do LDAS. Para resolver esta situação, uma interface foi desenvolvida para alertas visuais, que são ativados sempre que um dispositivo está em um estado que requer atenção. Considerando-se esses alertas, os detalhes podem ser inferidos através da interface da ferramenta de visualização desenvolvida para o Projeto AMPLUS.

A avaliação da arquitetura do CoIoT ocorreu através da avaliação de aceitação da ferramenta desenvolvida. O estudo envolveu 10 voluntários, entre professores, alunos e técnicos, com atividades relacionadas ao LDAS. 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), usando uma escala de Likert [Yoon and Kim 2007], que varia entre 1 e 5. 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. A média dos resultados obtidos para facilidade de uso foi de 4,7 enquanto que para a percepção de utilidade foi obtida uma média de 4,4. A análise desses resultados mostrou que a aprovação foi elevada tanto para a facilidade de uso bem como para a percepção de utilidade.

4. Trabalhos relacionados

O estudo da literatura permitiu identificar alguns trabalhos relacionados, entre as quais foram selecionados os seguintes: EcoDiF [Pires et al. 2014], Xively [LogMeIn 2015], Carriots [Carriots 2015] e LinkSmart [Kostelník et al. 2011]. Os aspectos considerados na seleção destes trabalhos relacionados foram: (i) o suporte à heterogeneidade; (ii) suporte ao gerenciamento de eventos; (iii) ciência do contexto; e (iv) interoperabilidade.

Entre os trabalhos relacionados, a ciência de contexto é suportada apenas pelo LinkSmart, mas o suporte oferecido é limitado. Por sua vez, o CoIoT oferece um mecanismo para a coleta e processamento distribuído de contexto através de regras, bem como para a atuação sobre o ambiente.

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. 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.

A manipulação de eventos é suportado por todos os trabalhos selecionados, mas apenas o Carriots utiliza regras neste tratamento. Além disso, o gerenciamento distribuído das regras entre os Servidores de Contexto e Borda é um diferencial em relação aos outros projetos. Esta funcionalidade de processamento de contexto nos trabalhos relacionados, é normalmente restrita a um único dispositivo.

5. Conclusão

Este artigo resume os esforços de pesquisa associados com a concepção do CoIoT. CoIoT é uma arquitetura para a Internet das Coisas, integrado ao Middleware EXEHDA, que gerencia a coleta e pré-processamento das informações contextuais, oferecendo suporte à atuação no ambiente.

A principal contribuição deste trabalho é concepção de uma arquitetura para IoT direcionada à ciência de contexto. A arquitetura proposta é orientada a eventos, baseada em regras e gerencia a coleta e processamento das informações contextuais, bem como as atuações no ambiente físico de forma distribuída. A estratégia adotada para a CoIoT ampliou o escopo de utilização do *Middleware* EXEHDA, permitindo seu uso em diferentes cenários.

Na continuidade da pesquisa os seguintes aspectos devem ser considerados em trabalhos futuros: (i) expandir o uso do CoIoT no LDAS, possibilitando o monitoramento de outros equipamentos de laboratório e, consequentemente, incorporando outros tipos de sensores e atuadores; e (ii) continuar os procedimentos de integração de CoIoT com os diferentes serviços e recursos do *Middleware* EXEHDA.

Referências

Carriots (2015). Carriots. https://www.carriots.com/. Acessado em Maio de 2015.

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

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.

Kostelník, P., Sarnovský, M., and Furdík, K. (2011). The semantic middleware for networked embedded systems applied in the internet of things and services domain. *Scalable Computing*, 12(3):307–315.

LogMeIn, I. (2015). Xively. https://xively.com/. Acessado em Fevereiro de 2015.

Lopes, J., Souza, R., Geyer, C., Costa, C., Barbosa, J., Pernas, A., and Yamin, A. (2014a). A Middleware Architecture for Dynamic Adaptation in Ubiquitous Computing. *Journal of Universal Computer Science*, 20(9):1327–1351.

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

Perera, C., Zaslavsky, A., Christen, P., and Georgakopoulos, D. (2014). Context aware computing for the internet of things: A survey. *IEEE Communications Surveys and Tutorials*, 16(1):414–454.

Pires, P. F., Cavalcante, E., Barros, T., Delicato, F. C., Batista, T., and Costa, B. (2014). A Platform for Integrating Physical Devices in the Internet of Things. *2014 12th IEEE International Conference on Embedded and Ubiquitous Computing*, pages 234–241.

Razzaque, M. A., Milojevic-Jevric, M., Palade, A., and Clarke, S. (2016). Middleware for Internet of Things: A Survey. *IEEE INTERNET OF THINGS JOURNAL*, 3(1):70–95.

Souza, R., Lopes, J., Geyer, C., Garcia, C., Davet, P., and Yamin, A. (2015). Context awareness in UbiComp: An IoT oriented distributed architecture. In *2015 IEEE International Conference on Electronics, Circuits, and Systems (ICECS)*, pages 535–538.

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

Yoon, C. and Kim, S. (2007). Convenience and TAM in a ubiquitous computing environment: The case of wireless LAN. *Electronic Commerce Research and Applications*, 6(1):102–112.