UNIVERSIDADE ESTADUAL DE MARINGÁ
CENTRO DE TECNOLOGIA
DEPARTAMENTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

GUILHERME SIMÃO COUTO

Manna-X: Um framework para Ambientes Inteligentes

Maringá
2012

GUILHERME SIMÃO COUTO

Manna-X: Um framework para Ambientes Inteligentes

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação do Departamento de Informática, Centro de Tecnologia da Universidade Estadual de Maringá, como requisito parcial para obtenção do título de Mestre em Ciência da Computação.

Orientadora: Profª. Drª. Linnyer Beatrys Ruiz

Maringá
2012

Dados Internacionais de Catalogação-na-Publicação (CIP)
(Biblioteca Central - UEM, Maringá, PR, Brasil)

C871m

Couto, Guilherme Simão
Manna-X: Um framework para Ambientes Inteligentes / Guilherme Simão Couto. -- Maringá, 2012.
142 f. : il. color., figs., tabs., retrs.

Orientador : Prof. Dr. Linnyer Beatrys Ruiz Aylon.
Dissertação (mestrado em Ciência da Computação) - Universidade Estadual de Maringá, Centro de Tecnologia, Departamento de Informática, Programa de Pós-Graduação em Ciência da Computação, 2012.

1. Ambientes Inteligentes. 2. Engenharia de Computação Invisível. 3. Automação Residencial. 4. Domótica. 5. Internet das Coisas. I. Aylon, Linnyer Beatrys Ruiz, orient. II. Universidade Estadual de Maringá. Centro de Tecnologia. Departamento de Informática. Programa de Pós-Graduação em Ciência da Computação. III. Título.

CDD 21.ed. 004.21

ZSS-00877

FOLHA DE APROVAÇÃO

GULHERME SIMÃO COUTO

Manna-X: Um framework para ambientes inteligentes

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação do Departamento de Informática, Centro de Tecnologia da Universidade Estadual de Maringá, como requisito parcial para obtenção do título de Mestre em Ciência da Computação pela Banca Examinadora composta pelos membros:

BANCA EXAMINADORA

Profa. Dra. Linnyer Beatrys Ruiz Aylon
DIN/UEM – Orientadora – Presidente

Prof. Dr. Wagner Igarashi
DIN/UEM – Membro Interno

Prof. Dr. Luiz Filipe Menezes Vieira
PPGCC/UFMG – Membro Externo

Aprovada em: 10 de dezembro de 2012.
Local da defesa: Sala 101, Bloco C56, *campus* da Universidade Estadual de Maringá.

AGRADECIMENTOS

Agradeço àqueles que contribuíram de forma direta para a realização deste trabalho, em especial:

À Deus, pela saúde e oportunidades concedidas.

Aos meus pais pela educação, apoio e amor.

À minha namorada Ana Paula Rolim Borghi por todo apoio, carinho e compreensão.

À minha orientadora Professora Doutora Linnyer Beatrys Ruiz Aylon, pelo apoio e a confiança depositada em mim para o desenvolvimento deste projeto.

À todos os pesquisadores do grupo de pesquisa Manna.

Ao CNPq (Conselho Nacional de Desenvolvimento Científico e Tecnológico) e a CAPES por parte do apoio financeiro concedido a este trabalho.

Manna-X: Um framework para Ambientes Inteligentes

RESUMO

No desenvolvimento de soluções de Ambientes Inteligentes (AI) diferentes tecnologias podem ser usadas desde que ofereçam conforto e transparência aos usuários. A diversidade de dispositivos e tecnologias deve atender a um requisito principal, a invisibilidade, isto é, os habitantes do ambiente não devem ter ciência de que a tecnologia está embutida e realizando tarefas com diferentes objetivos, por exemplo: proporcionar bem estar aos habitantes, cooperar com a sustentabilidade, monitorar entidades (pessoas, animais e coisas), entre outras. Os elementos do AI estarão trocando informações, colaborando e cooperando entre si e com os próprios usuários no cumprimento dos objetivos definidos para o ambiente. Isto leva a um ambiente heterogêneo que requer uma forma de se conseguir disponibilizar trocas entre elementos fabricados com diferentes tecnologias. A tarefa de integrar a diversidade não é trivial. Este trabalho lida com este desafio: estudar os conceitos relacionados com Ambientes Inteligentes, identificar oportunidades de pesquisa e propor uma solução para a integração de diferentes tecnologias. Em particular, o trabalho propõe um arcabouço chamado de Manna-X e desenvolve protótipos como prova de conceitos das boas práticas definidas para se auxiliar desenvolvedores e fabricantes na implementação de ambientes inteligentes. Estão envolvidas as seguintes áreas e tecnologias: Rede de Sensores Sem Fio, Internet das Coisas, Kinect, Zigbee, dentre outras.

Palavras-chave: Ambientes Inteligentes, Engenharia de Computação Invisível, Automação Residencial, Domótica, Internet das Coisas.

Manna-X: A framework for Intelligent Environments

ABSTRACT

In the development of Intelligent Environments (IE) solutions different technologies can be used since they offer comfort and transparency to the users. The diversity of devices and technologies must follow a major requirement, invisibility, i.e. the inhabitants of the environment should not be aware that the technology is embedded and performing tasks with different objectives, e.g. to provide welfare to the inhabitants, cooperate to sustainability, monitoring entities (people, animals and things), among others. The elements of IE will be exchanging information, collaborating and cooperating with each other and with the user in achieving the goals defined for the environment. This leads to a heterogeneous environment which requires a way to provide exchanges between elements from different technologies. The task of integrating diversity is not trivial. This thesis deals with this challenge: to study the concepts related to Intelligent Environments, identify research opportunities and propose a solution for the integration of different technologies. In particular, the thesis we propose a framework called Manna-X and develop prototypes as proof of concepts of good practice set to assist developers and manufacturers in the intelligent environments implementation. In this work the following areas and technologies are involved: Wireless Sensor Networks, Internet of Things, Kinect, Zigbee, among others.

Keywords: Intelligent Environments, Invisible Computer Engineering, Home Automation, Domotic, Internet of Things.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS E ABREVIATURAS

AES: Advanced Encryption Standard
AI: Ambiente Inteligente
AmI: Ambient Inteligence
CEBUS: Consumer Electronics Bus
ECI: Engenharia de Computação Invisível
EIB: European Installation Bus
EOA: Arquitetura Orientada a Eventos
IA: Inteligência Artificial
IEEE: Institute of Electrical and Electronics Engineers
IHC: Interface Humano-Computador
IoT: Internet of Things
JVM: Java Virtual Machine
M2M: Machine-to-Machine
NFC: Near Field Communication
OSGi: Open Services Gateway Initiative
p2p: Peer-to-Peer
REST: Representational State Transfer
RFID: Radio-Frequency IDentification
ROA: Arquitetura Orientada a Recursos
RSSF: Redes de Sensores Sem Fio
SOA: Arquitetura Orientada a Serviços
T3: Things That Think
UH: Ubiquitous Healthcare
UPnP: Universal Plug and Play
WoS: Web of Sensors
WoT: Web of Things

SUMÁRIO

Capítulo

1

INTRODUÇÃO

A cada dia a computação tem sido inserida no cotidiano sem que a maioria das pessoas tenha ciência disto. A cada dia a computação tem se tornado transparente a ponto de seus usuários nem perceberem que já estão utilizando mais de um processador, software de diferentes propósitos e mais de um tipo de conexão com o mundo digital.

Muitas pessoas já utilizam dois celulares, máquinas de lavar, micro-ondas, televisores, e sistemas de alarmes micro processados. No trabalho elas também estão usando diferentes tecnologias que as habilitam para as mais diferentes tarefas. O mais comum é observar pessoas usando computadores em seus ambientes profissionais. Elas têm a impressão que apenas o computador é o ponto de contato com a tecnologia. Contudo, as novidades do mundo tecnológico vão além dos tradicionais computadores, Tablets, PDAs, smartphones, e a própria Internet (Ruiz et al., 2011).

Uma das tendências é a instrumentação do mundo físico com sensores de dimensões micro e até nanométricas. A novidade não está no aspecto funcional dos sensores que desde o início dos tempos tem a função de coletar dados, mas no fato de que eles estão deixando de usar cabos, passando a usar conexões sem fios e ainda estão sendo produzidos com processadores e baterias. Assim, o sensor de tamanho minúsculo passa a ter inteligência e pode processar os dados coletados e raciocinar sobre estes. Os sensores também estão aptos a enviar os dados por conexões sem fio e tomar decisões individuais ou coletivas, em cooperação com outros sistemas. Como resultado das pequenas dimensões e da liberdade advinda da ausência de fios, estes sensores podem ser utilizados em diferentes aplicações, tais como, monitoração da força de trabalho, computação vestível, *Health Care*,

agricultura de precisão, monitoração de infra estrutura, monitoração de edifícios e obras civis e militares, ecologia de paisagens, ambientes inteligentes, etc.

Este trabalho lida com o desafio de instrumentar o dia a dia das pessoas propondo o desenvolvimento de um ambiente inteligente onde casas, escolas, hospitais, ginásios, centros de treinamento, vestiários, instalações, hotéis, praças, lojas dentre tantos outros ambientes possam usufruir de um espaço inteligente que promoverá a excelência no bem estar, bem como também na comodidade por não precisar executar tarefas diárias como regar o jardim, acender as luzes a noite, ligar o ar condicionado quando está calor e outras cuja o funcionamento pode incluir funções de sustentabilidade.

A computação disponível 24 horas por dia, 07 dias por semana e disponível em qualquer lugar já é o desejo de muitos usuários e uma alternativa interessante para idosos, deficientes, crianças, treinadores e atletas empenhados em alcançar a performance máxima.

O foco do trabalho está no desenvolvimento de uma arquitetura consolidada em um framework chamada de Manna-X que permite integrar diferentes dispositivos com fio e sem fio (embutidos em artefatos, roupas, móveis, imóveis, construções, edificações de maneira imperceptível aos usuários) na formação de uma rede virtual que permite coletar e processar dados e informações sobre pessoas e ambientes promovendo a monitoração e controle. Neste último caso, o controle pode estar relacionado a aspectos como controle de parâmetros ambientais tais como temperatura, umidade, ventilação, por meio de atuadores.

O Manna-X foi pensado como uma ferramenta de auxílio ao mundo dos desenvolvedores de ambientes inteligentes e um exemplo de como o mundo computacional pode passar da transparência à invisibilidade ao se ter diferentes dispositivos espalhados como um spray computacional, funcionando em rede para provisionar diferentes serviços. Com o Manna-X, a computação embutida em diversos itens do mundo das pessoas e usando diferentes tecnologias de comunicação poderá cooperar e trocar informação de forma transparente, permitindo que os ambientes acolham os seus usuários utilizando de tecnologia que gera calma e que mantém a informação disponível a todo momento e em qualquer lugar.

Este capítulo está organizado da seguinte maneira. As próximas seções tratam sobre a Problematização e Motivação sobre o tema de Ambientes Inteligentes. Em seguida, os objetivos a serem alcançados pelo trabalho de mestrado serão apresentados bem como sua justificativa científica. Por fim, a Organização Completa do Texto será explicada.

1.1 PROBLEMATIZAÇÃO E MOTIVAÇÃO

As aplicações em potencial para a engenharia de Computação Invisível têm estimulado o desenvolvimento dessa tecnologia e atraído a atenção da comunidade acadêmica. No Brasil, o INCT NAMITEC[1] (Instituto Nacional de Ciência e Tecnologia de Sistemas Micro e Nanoeletrônicos) é um dos 123 institutos criados pelo Programa de Institutos Nacionais de Ciência e Tecnologia do Ministério de Ciência e Tecnologia que tem como tema central de pesquisa as Redes de Sensores Sem Fio e seus componentes.

O desenvolvimento de elementos de hardware e software de Rede de Sensores Sem Fio (em escalas micro e nano), tecnologia que habilita a engenharia de computação invisível, proposto como objetivos do INCT NAMITEC tem caráter estratégico para tornar possível a implantação no Brasil de indústrias de base eletrônica inovadoras ou de fornecedoras de tecnologias para essas no domínio da micro e nano-eletrônica. No exterior, as RSSF foram incluídas como uma das seis áreas de grande desafio de pesquisa no relatório do Workshop da National Science Foundation em 2003 (NSF, 2003). Toda esta tecnologia demonstrará a formação de uma nova geração de engenheiros e o surgimento de uma nova classe computacional diferenciada do passado, fascinante no presente e bastante útil no futuro.

Toda esta novidade tecnológica faz surgir alguns contrapontos que dizem respeito ao empenho financeiro e custo amortizado de se produzir, implantar e manter tais sistemas. Também aspectos relacionados com a legislação, políticas públicas, aspectos éticos e morais, competição e colaboração com o trabalho humano, propriedades intelectuais e simplicidade da vida.

O crescente estudo sobre Ambientes Inteligentes, Computação Invisível e Ubíqua gera grandes expectativas sobre o futuro da humanidade. Os avanços tecnológicos nunca evoluíram tanto como nos últimos anos a ponto de pensarmos em como será a casa, escola e carro do futuro ou como será a interação com os aparelhos eletrônicos.

Nos dias de hoje, alguns dispositivos já estão conectados na Internet, mas poucos possuem uma interface por comandos de voz ou gestos. Os dispositivos encontrados ao redor das pessoas não possuem alguma inteligência para ajudar as pessoas a tomar decisões, ou funções diferenciadas para melhorar a vida delas. A criação de ambientes inteligentes não é algo trivial de se desenvolver, tais soluções enfrentam diversos problemas como os encontrados na literatura (Römer et al., 2002; Rui et al., 2009; Sampaio et al., 2012), e esses podem ser assim apontados:

[1] `http://namitec.cti.gov.br/`

- Como integrar dispositivos fabricados por fabricantes diferentes e criar um ambiente com a possibilidade de acrescentar outros dispositivos heterogêneos?
- Quais são os dados e parâmetros a serem coletados dos diferentes ambientes com objetivo de fornecer dados úteis a serem utilizados pelos usuários do ambiente?
- Como será a interação do usuário com os dispositivos encontrados no ambiente de maneira que seja de fácil acesso, ubíquo e útil?
- Como criar ambientes inteligentes, ubíquos, sensíveis ao contexto, embora não exista muitos padrões de integração de dispositivos?

Este trabalho tem a expectativa de contribuir para o avanço tecnológico dos dispositivos que cercam as pessoas, tanto no dia a dia em casas, quanto na prática dos esportes, apresentações, aulas, teatros, estádios, laboratórios, hotéis, ambientes de lazer como bares, boates e etc. Mais do que isso, ele visa aumentar a experiência do usuário no que se refere ao acesso à informação sobre os dispositivos que o cercam, melhorar a Interface Humano-Computador para que o acesso aos dispositivos seja feita de forma intuitiva e fácil, bem como também dar suporte para os desenvolvedores de dispositivos criarem dispositivos novos com inteligência, mais fáceis de usar, com mais funcionalidades, podendo até serem compartilhados pela Internet por meio das redes sociais.

1.2 OBJETIVO GERAL

O objetivo geral deste trabalho é estudar, projetar e desenvolver um arcabouço tecnológico (framework) a ser utilizado por desenvolvedores de ambientes inteligentes e fabricantes de dispositivos que sirva como uma plataforma que promova a integração de diversos dispositivos heterogêneos de modo a propor uma solução para o problema clássico de interoperabilidade dos ambientes inteligentes.

1.3 OBJETIVOS ESPECÍFICOS

Os objetivos específicos para este projeto envolvem:

1. Estudar os problemas relacionados com a área de Ambientes Inteligentes, considerando os temas abordados em diferentes tecnologias, quais sejam: Engenharia de Computação Invisível, Domótica, Automação e Internet das Coisas;

2. Desenvolver uma arquitetura flexível que permita integrar as diferentes tecnologias aplicadas à instrumentação considerando elementos de hardware e software bem como disponibilizar os dados, coletados e processados, em diferentes ambientes tais como Internet e Redes Sociais. Esta arquitetura provê interfaces que permitirão a diferentes fabricantes desenvolver e construir dispositivos que poderão ser integrados ao ambiente inteligente, proposto como prova de conceitos;

3. Elaborar meios para que dispositivos feitos por fabricantes diferentes possam ser utilizados na mesma solução para ambientes inteligentes de modo que o desenvolvedor da solução consiga utilizá-los em cooperação;

4. Desenvolver um protótipo que sirva como prova de conceitos do desenvolvimento de uma solução de ambiente inteligente usando o Manna-X.

5. Como prova de conceitos a implementação do framework e sua utilização em um ambiente genérico será realizada. Em particular o trabalho tem como base tópicos relacionados com Internet das Coisas (Kortuem et al., 2010), Mashups físicas (Guinard, 2010), Ambientes Inteligentes (Rui et al., 2009), Computação Ubíqua (Weiser, 1991), Redes de Sensores Sem Fio (Ruiz et al., 2003) e Sensoriamento Social (social sensing) (Rosi et al., 2011).

1.4 ORGANIZAÇÃO DO TEXTO

O restante da dissertação está organizado conforme a descrição a seguir. O Capítulo 2 intitulado de Fundamentos Teóricos tem como objetivo esclarecer os conceitos pesquisados, bem como discutir os principais temas envolvidos para que o leitor tenha um melhor entendimento sobre o trabalho proposto, neste capítulo algumas áreas relacionadas e tecnologias utilizadas serão explicadas.

No Capítulo 3, uma nova proposta de desenvolvimento em ambientes inteligentes é mostrada para o leitor. Essa metodologia de desenvolvimento vem a tentar mudar o paradigma de como são feitos ambientes inteligentes até o momento, explicando problemas encontrados na área. Além disso, o capítulo mostra para o leitor outros trabalhos relacionados encontrados na literatura que são semelhantes com a solução proposta por essa dissertação. Os trabalhos encontrados além de servirem como comparação, tiveram grande contribuição para a solução proposta, evidenciando novas ideias, técnicas, tecnologias e tendências.

O Capítulo 4 tem por objetivo apresentar a solução proposta para resolver os problemas de desenvolvimento na área de ambientes inteligentes. Utilizando a nova proposta de desenvolvimento, neste capítulo o framework Manna-X que foi projetado e desenvolvido como objetivo desta dissertação será apresentado. A divisão desse capítulo evidencia uma visão geral do arcabouço desenvolvido, descreve as partes, e interfaces do framework e por fim detalha seu funcionamento e interfaces.

No Capítulo 5, a fim de provar a viabilidade do framework proposto, são apresentados detalhes da implementação do framework. Um experimento realizado na Universidade Estadual de Maringá será mostrado provando a viabilidade, performance e utilização do framework proposto. O ambiente utilizado foi o próprio laboratório de pesquisa do grupo Manna, localizado na Universidade Estadual de Maringá. Por fim, o trabalho é comparado com outros encontrados na literatura e suas diferenças evidenciadas.

As conclusões, dificuldades e os trabalhos futuros observados são mostrados no Capítulo 6.

Capítulo

2

FUNDAMENTOS TEÓRICOS

No desenvolvimento de soluções de Ambientes Inteligentes diferentes tecnologias podem ser usadas para oferer conforto e transparência aos usuários. Este é um dos paradigmas de uma disciplina emergente definida como Engenharia de Computação Invisível. Segundo Ruiz (Ruiz et al., 2011), a ECI está relacionada com a forma transparente e intensa de interação entre pessoas e elementos computacionais, e a interação entre os próprios elementos computacionais na instrumentação de ambientes e provisionamento de diferentes serviços. As diferentes formas de se entender o universo das coisas que são dotadas de capacidade computacional e comunicação sem fio e que estão interagindo com entidades (pessoas, plantas, edificações, etc) vem recebendo diferentes nomenclaturas e focos de atenção.

Na literatura estão relacionados alguns termos e áreas afins com a Engenharia de Computação Invisível, quais sejam: *Ubiquitous Healthcare* (UH), *Pervasive Computing, Calm Technology, Things That Think* (T3), *Everyware, Ambient Intelligence (AmI), Context Based Computing, Sentient Computing, Autonomous Computing, Amorphous Computing, Universal Computing, Tangible Computing*, Computação Sensível a Posição, *Palpable Computing, Universal Computing, Tangible Computing*, Computação Impregnante, Computação Desagregada, Computação Nomádica, Computação Espontânea, *Augmented Reality, Life Computing, Anima Computing* (senso comum), *Internet of Nano-Things, Post PC Computing* (senso comum), *Everyday Computing*. Em particular, alguns conceitos relacionados com estas áreas foram selecionados para o desenvolvimento da solução proposta incluindo tecnologias emergentes, tais como: Rede de Sensores Sem Fio (RSSF), *Web of*

Sensors (WoS), Internet das Coisas (IoT), *Assisted Living*, Interação Humano-Computador, *Mashups* físicas e os fundamentos específicos da área de Ambientes Inteligentes.

As tecnologias citadas acima habilitam ou têm em comum características de computação ubíqua, termo usado para descrever a computação que está disponível em todo lugar a todo momento (Weiser, 1991). Então, os dispositivos envolvidos com a computação ubíqua devem ser dotados de interface de comunicação sem fio. Algumas das tecnologias de comunicação mais comuns têm sido ZigBee[1], Z-Wave[2] e NFC[3]. Além destas, algumas aplicações têm usado X10[4], Bluetooth[5], KNX[6], WiFi, UPnP[7], e etc.

Este capítulo apresenta tópicos relevantes relacionados com as ciências e tecnologias que foram utilizadas no desenvolvimento e implementação do arcabouço proposto para ambientes inteligentes. As tecnologias são usadas como materiais de desenvolvimento. A seção 2.1 apresenta uma visão geral de Ambientes Inteligentes. A seção 2.2 trata sobre a computação ciente de contexto aplicada em ambientes inteligentes. A seção 2.3 apresenta a tecnologia disponível para instrumentar ambientes, as Redes de Sensores Sem Fio. Na seção 2.4 é apresentado o nó sensor Waspmote que foi utilizado como estudo de caso. A seção 2.5 evidencia a tecnologia de nós sensores MICAz. A seção 2.6 mostra detalhes e o funcionamento do dispositivo Microsoft Kinect utilizado no protótipo implementado. A seção 2.7 tem por objetivo esclarecer detalhes sobre a tecnologia OSGi que foi utilizada para a implementação do arcabouço. A seção 2.8 trata sobre um tema atual chamado Internet das Coisas e Web das Coisas. A seção 2.9 relata sobre a tecnologia de comunicação ZigBee e IEEE 802.15.4, que foi utilizada na implementação do framework. Por fim, na 2.11 apresenta as conclusões finais sobre este capítulo.

2.1 AMBIENTES INTELIGENTES

A base dos ambientes inteligentes (AI) é feita pela Automação Residencial (AR). Segundo (Gill et al., 2009) a Automação Residencial é a introdução da tecnologia dentro de casa para melhorar a qualidade de vida de seus ocupantes, através da prestação de serviços diferentes, tais como entretenimento multimídia, tele saúde e conservação de energia.

[1] http://www.zigbee.org/
[2] http://www.z-wave.com/modules/ZwaveStart/
[3] http://www.nfc-forum.org/home/
[4] http://www.x10.com/homepage.htm
[5] http://www.bluetooth.com/Pages/Bluetooth-Home.aspx
[6] http://www.knx.org/
[7] http://www.upnp.org/

Em (Muratori e Bó, 2011) tem-se a seguinte definição para AR: é o conjunto de serviços proporcionados por sistemas tecnológicos integrados como o melhor meio de satisfazer as necessidades básicas de segurança, comunicação, gestão energética e conforto de uma habitação. Nesse contexto, a maior parte das pessoas acha mais adequado o termo "domótica", largamente empregado na Europa, pois é mais abrangente. No entanto, no Brasil, foi optado pela tradução literal de *home automation*, denominação americana mais restrita, uma vez que, conceitualmente, o termo "automação" não englobaria, por exemplo, sistemas de comunicação ou sonorização.

A definição do termo Domótica, junção das palavras "Domus" e "Robótica", é a ciência que estuda a gestão de todos os recursos habitacionais. Segundo Dias (Dias e Pizzolato, 2004), o sistema domótico contém uma rede de comunicação que integra vários modelos de dispositivos, equipamentos e outros sistemas, com o objetivo de obter informações sobre o ambiente residencial e o meio em que ele se insere, efetuando determinadas ações a fim de supervisioná-lo ou gerenciá-lo. Houve uma pesquisa significativa no campo da automação residencial nas últimas décadas. O padrão da indústria X10, desenvolvida em 1975 para a comunicação entre dispositivos eletrônicos, é o mais antigo padrão identificado a partir de (Gill et al., 2009), proporcionando um controle limitado sobre aparelhos domésticos por meio de linhas de energia da casa. Depois dele surgiram diversos padrões tais como: BACNet[8], BatiBUS(Allen e Dillon, 1997), CEBUS(Markwalter e Russell, 1988), EIB(EIBA, 1999), HomePnP(O'Driscoll, 2000), LonTalk[9], LonWorks[10], UPnP[11], ZigBee, Z-wave, Bluetooth, OBiX[12], CABA[13], DLNA[14], KNX, Ethernet, e etc.

Foi com base na ideia da automação residencial, que vários dispositivos de uma casa poderiam ter alguma inteligência, que Weiser (Weiser, 1991) pensou num mundo diferente do vivido até hoje, onde a computação move-se para fora das estações de trabalho, computadores pessoais (PCs) e torna-se pervasiva em nossa vida cotidiana. No mundo idealizado por ele, deve-se aprender a conviver com computadores, e não apenas interagir com eles. Esses sistemas então passaram a ser definidos por ele de Ubíquos. De acordo com sua visão, os dispositivos ubíquos seriam elementos computacionais capazes de desaparecer, visto estarem acoplados aos objetos cotidianos de tal maneira a se tornarem indistinguíveis dos mesmos. Dessa maneira, as aplicações ubíquas podem prover serviços ou tarefas aos seus usuários de maneira transparente, personalizada e sob demanda.

[8] http://www.bacnet.org/
[9] http://www.hvacc.net/pdf/lonworks/Echelon\%20-\%20LonTalk\%20Protocol.pdf
[10] https://www.echelon.com/technology/lonworks/
[11] http://www.upnp.org/
[12] http://www.obix.org/
[13] http://www.caba.org/
[14] http://www.dlna.org/

Os avanços na miniaturização dos eletrônicos estão permitindo que dispositivos com computação, de várias capacidades e interfaces se tornem parte de nossa vida diária. Sensores, atuadores e unidades de processamento já podem ser comprados a preços mais acessíveis. Esta tecnologia pode ser usada em rede e com a coordenação de software altamente inteligente para compreender os eventos e contextos relevantes de um ambiente específico e tomar decisões sensatas em tempo real ou posteriormente (Augusto et al., 2010).

A descrição acima é uma das várias encontradas na literatura sobre ambientes inteligentes. O termo *Smart Spaces* também é usado como um sinônimo para ambientes inteligentes, que segundo (Poslad, 2009) tem a seguinte definição: são ambientes constituídos de dispositivos, sensores e atuadores, embarcados que juntos proporcionam informações ao usuário sobre o ambiente. Além disso, esses dispositivos devem ser sensíveis ao contexto, pois devem perceber o meio que estão inseridos e atuar dinamicamente conforme as mudanças em um ambiente de acordo com a estratégia adotada.

O trabalho de (Chandrasekhar et al., 2011) divide o sistema de inteligência ambiental em *front-end* e *back-end*. No *front-end* de um sistema de AmI estão uma variedade de dispositivos minúsculos que podem ouvir, ver ou sentir a presença/ausência de um usuário final. No *back-end*, uma rede de sensores sem fio dá o sentido a esses dados, identificando o usuário final, compreendo as necessidades dele/dela e atendendo as solicitações presentes. É com base nessa definição que o arcabouço Manna-X foi especificado.

Uma das principais características do ambiente inteligente é a sua inteligência propriamente dita, isto é, a maneira como o sistema vai aprender com as preferências dos habitantes, analisar o comportamento dos habitantes, perceber quais são as intenções dos habitantes, e vir a controlar o ambiente de forma pró-ativa para atender as suas necessidades de acordo com os parâmetros ambientais, tais como temperatura e umidade. O Ambiente Inteligente (AI) pode estar ciente da presença e contexto de seus habitantes, e sensível, adaptável e compreensível às necessidades dos habitantes. O AI também pode criar as condições ambientais desejadas e funcionalidades por meio de sistemas inteligentes interligados e seus serviços para os habitantes (yong Duan e Li, 2010).

Outro termo encontrado na literatura é o Ambient Intelligence (AmI), ou inteligência ambiente em português. Ele foi cunhado pela Comissão Européia, quando em 2001 um dos seus grupos de programa consultivos, European Community's Information Society Technology (ISTAG), lançou o desafio AmI (Ducatel et al., 2001), mais tarde atualizada em 2003 (Ducatel et al., 2003). Inteligência Ambiente (AmI) é a visão de um futuro em que os ambientes apoiam as pessoas que nelas habitam. Este ambiente é discreto, previsto, interligado, adaptável, dinâmico, integrado e inteligente (Sadri, 2011).

Ambientes com AmI visam produzir uma mudança significativa na maneira como as pessoas vivem, graças ao crescimento da tecnologia digital, e a grande disponibilidade de diferentes tipos de dispositivos. No entanto, a presença generalizada de dispositivos eletrônicos não cria ambientes inteligentes, uma vez que estes dispositivos devem coordenar e cooperar entre si a fim de apoiar as atividades humanas.

Em (Chandrasekhar et al., 2011) é dito que para que a AmI fique atualizada, o usuário deve perceber que o ambiente que ele/ela está vivendo é inteligente, mesmo se o ambiente, individualmente, não é inteligente em tudo. Na verdade, a AmI não implica que o ambiente (ou seja, os componentes ambiente e ou dispositivos) deve ser dotado de uma espécie de inteligência artificial, mas que deve ser percebida como inteligente pelo usuário, por exemplo, reagir e antecipar suas ações com base em eventos passados ou suas preferências por trazer algum conceito de probabilidade ou de ações de comportamento seletivo.

As características-chaves para AmI são a inteligência e a incorporação. Pela inteligência, ela significa que o sistema é sensível ao contexto, é adaptativa, aprende com o comportamento dos usuários e, finalmente, reconhece e expressa a emoção. Incorporando ele quer dizer pequeno, possivelmente pequenos dispositivos que se fundem com o fundo de atividades das pessoas e ambientes. Ou seja, aplicações em ambientes inteligentes visam dar assistência ao usuário durante o dia a dia por meio da incorporação da inteligência ambiental no meio ambiente (Lehmann et al., 2010).

Outra característica importante encontrada em (Kameas, 2010) para AmI, é a fusão do espaço físico e digital, isso significa o mapeamento dos dispositivos. Sem esse mapeamento o sistema nunca poderia saber quais dispositivos ele está gerenciando, e nem o usuário saberia quais dispositivos os sistema está de fato gerenciando, pois em uma casa dos dias de hoje nem todos os dispositivos são inteligentes e passíveis de serem integrados.

2.2 COMPUTAÇÃO CIENTE DO CONTEXTO APLICADA A AMBIENTES INTELIGENTES

Aplicações em ambientes inteligentes necessitam monitorar o usuário, a si mesmos e seu meio ambiente e proporcionar reações apropriadas às mudanças monitoradas antes de levar a uma interrupção de operação, e para isso a informação de contexto (usuários, dispositivos e o ambiente) requerida deve ser bem definida (Lehmann et al., 2010). Estes dados poderão ser disponibilizados para as aplicações e através de modelos computacionais as interfaces poderão “entender” o que está acontecendo ao redor do usuário e adaptar-se

ou tomar decisões, alterando e facilitando a comunicação com o humano. Esta definição é conhecida como Consciência de Contexto.

Contexto foi especificado por (Dey e Abowd, 1999) como "qualquer informação que possa ser usada para caracterizar a situação de entidades (ex. uma pessoa, lugar ou objeto) que são consideradas relevantes para a interação entre um usuário e a aplicação, incluindo o usuário e a aplicação". A lógica básica para análise de contexto, não leva em conta o tipo de armazenamento, desde que bem representado, porém o processamento segue geralmente no mesmo padrão como listado abaixo:

1. Percepção da informação de contexto;
2. Interpretar as informações sensoriadas (detectar e compreender as mudanças de contexto);
3. Selecionar ou definir uma estratégia de adaptação adequada;
4. Executar / Aplicar a estratégia de adaptação.

Esse processamento existe em todo sistema sensor/atuador, porém se visto num ambiente inteligente, a análise do contexto pode ser vista como na Figura 2.1. Nela pode se encontrar dois tipos de sensoriamento um relacionado à percepção de fatores tangíveis como temperatura, luminosidade, estado dos dispositivos, posicionamento das pessoas, ou seja, informações fáceis de mensurar que podem ser obtidos por uma rede de sensores e dispositivos do AmI. Já a parte de abstração tem relação com sensoriamento de fatores intangíveis como sentimentos, satisfação dentre outros fatores difíceis de mensurar. Esses fatores podem ser abstraídos de redes sociais por exemplo. O usuário pode escrever em seu Facebook que está triste, então o sistema AmI por ser sensível ao contexto atua de forma a proporcionar ao usuário um melhor ambiente para determinado contexto, no caso a tristeza.

Após a coleta dos dados dos sensores, o contexto é interpretado, com a ajuda de dados passados e através de alguma inteligência o sistema deve atuar para modificar o ambiente. Essa atuação no ambiente pode alterar a percepção, porém não a abstração, já que não se pode mudar dados intangíveis do usuário. Com isso forma-se um ciclo de processamento(Silva et al., 2010).

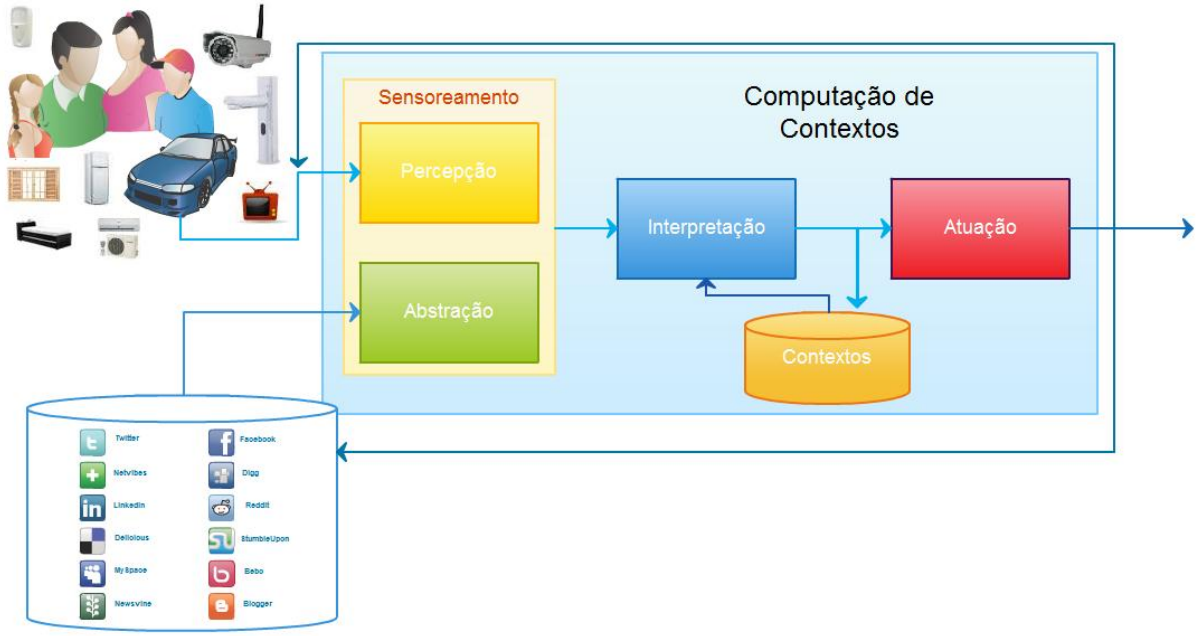

Figura 2.1: Computação de Contextos (Ruiz et al., 2011)

2.3 REDE DE SENSORES SEM FIO

Uma RSSF pode ser formada por centenas a milhares de elementos de rede com capacidade computacional e de comunicação sem fio, além da capacidade de sensoriamento e autonomia de energia. Trata-se de uma ferramenta de sensoriamento distribuído de fenômenos, processamento e disseminação de dados coletados e informações processadas para um ou mais observadores(Ruiz et al., 2003).

Os elementos de rede, chamados de nós sensores, são depositados em um ambiente (podem ser lançados sobre áreas remotas tais como reservas ambientais, oceanos, rios, florestas, áreas de desastre, etc. ou colocados em áreas internas tais como hospitais ou fábricas) e sem intervenção de técnicos ou operadores formam uma rede sem fio ad hoc que coleta dados sobre os fenômenos de interesse, realiza processamento local, e dissemina as informações para um ponto de acesso (gateway), através do qual a rede comunica-se com outras redes ou com usuários. O conceito de redes ad hoc é usado para caracterizar uma rede temporária, formada espontaneamente, sem administração centralizada e sem infraestrutura fixa.

Os componentes básicos de um nó sensor são: dispositivo de comunicação sem fio (ex.: rádio), processador, bateria e conexão para a placa de sensores que pode ser acoplada ao nó. A placa de sensoriamento pode ser composta por diferentes tipos

de dispositivos sensores, tais como magnetômetros, acelerômetros, sensores químicos, temperatura, pressão, radiação, luminosidade, umidade, estresse mecânico.

Em geral, os nós sensores são projetados com pequenas dimensões e esta limitação de tamanho acaba impondo limitações nos recursos dos nós, tais como capacidade da fonte de energia, processador e transceptor. Em alguns casos, uma RSSF também pode ser composta de dispositivos atuadores que permitem ao sistema controlar parâmetros do ambiente monitorado(Akyildiz et al., 2002).

A Figura 2.2 apresenta alguns tipos de nós sensores sem fio resultantes de pesquisas em diversas instituições, como o COTS Dust, o Smart Dust e Mica Dot da Universidade da Califórnia, Berkeley, Wireless Integrated Network Sensors da Universidade da Califórnia, Los Angeles e Sensor Web da NASA.

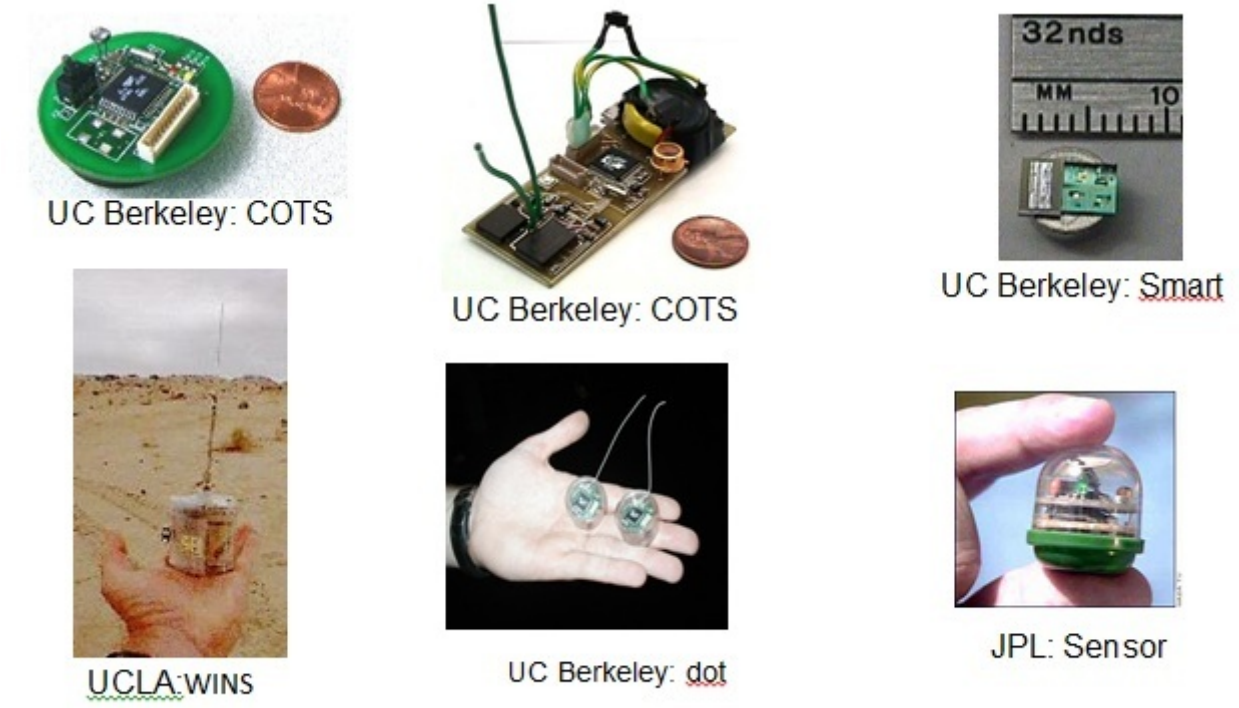

Figura 2.2: Projetos acadêmicos de nós sensores (Ruiz, 2003)

A expectativa é que os nós sensores venham a ter um baixo custo comercial e que as RSSFs se tornem disponíveis em todos os lugares executando as mais diferentes tarefas.

2.4 NÓ SENSOR WASPMOTE

Uma das tecnologias utilizadas para a realização desse projeto foram os nós WaspMote da empresa Libelium[15]. Esses nós, cuja estrutura foi baseada em Arduino[16], utilizam a rede ZigBee, Bluetooth e GPRS como forma de comunicação. A Libelium projeta e fabrica tecnologia de hardware para a implementação de redes de sensores sem fio para que as empresas integradoras de sistemas, engenharia e consultoria possam implementar soluções confiáveis e cidades inteligentes para usuários finais dentro do tempo mínimo para o mercado (Libelum., 2012).

Todos os produtos são modulares, horizontais e de rápido aprendizado e incluem uma extensa documentação e suporte através de uma comunidade de desenvolvedores. Suas principais linhas de pesquisa e desenvolvimento são:

Waspmote: Dispositivo sensor de baixo consumo para a criação de redes de sensores sem fio que integra mais de 50 sensores diferentes;

Meshlium: O único router multi-tecnologia que integra a malha Wi-Fi (2,4 GHz - 5GHz), ZigBee, GPRS, GPS e Bluetooth em uma única unidade.

Como dito, os nós utilizados foram os Waspmotes, que podem ser vistos na Figura 2.3. Nela pode-se ver o nó Waspmote no centro e ao seu redor exemplos do que pode ser acoclapdo a ela como a placa de expansão de radiação, uma antena GPS, outra GPRS, e sua bateria.

Além desses adendos, outros podem ser adicionados como a antena Bluetooth, placas de sensores de gás, sensores agrícolas, bem como relês e outros dispositivos eletrônicos que podem ser conectados.

[15] `http://www.libelium.com/`
[16] `http://arduino.cc/`

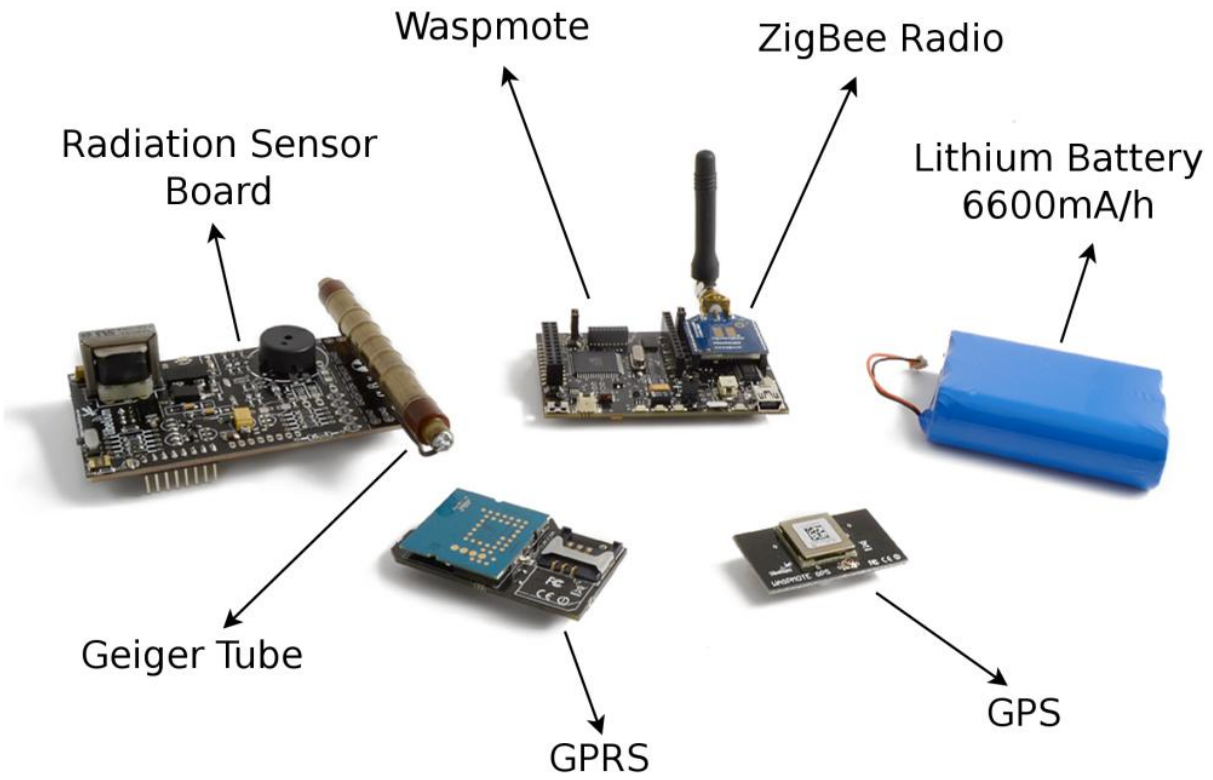

Figura 2.3: Nó Waspmote e seus adicionais

2.5 NÓ SENSOR MICAz

Desenvolvido pelos pesquisadores da Universidade de Berkeley e comercializado pela Crossbow, os modelos MICA2 Mote, MICA2 Dot e MICAz, são constituídos por uma série de sensores (temperatura, luminosidade, etc), microcontrolador RISC, memórias RAM, ROM, FLASH, conversores analógicos-digitais e transceptores.

O nó sensor MICAz, lançamento da família MOTE, permitiu a expansão de conexão para sensores de luminosidade, pressão barométrica, aceleração, magnético entre outros. Baseado na plataforma MPR 2400, com processador ATmega 128L, processador de 8 bits, este de baixo consumo, memória flash interna 128 Kbyte, interface de 51 pinos de expansão, conversores analógico/digital de 10 bits, comunicação I2C, SPI e interface serial. A frequência 2400/2483.5 MHz multi-canal com taxa de transmissão de 250 kbps. Este modelo de nó sensor roda o sistema operacional TinyOS e é utilizado em aplicações de monitoramento *indoor* de edifícios, acústico, vídeo, vibração, sensoriamento de alta

velocidade e RSSF de grande escala. Um exemplo de sensor MICAz é ilustrado a seguir na Figura 2.4:

Figura 2.4: Nó Sensor MICAz (Ruiz, 2003)

2.6 KINECT

O Kinect (anteriormente chamado de "Project Natal") é o nome de um projeto encabeçado pela Microsoft para seu console de videogame de última geração Xbox 360, que tem ainda como colaboradora a empresa Prime Sense[17]. O projeto visa criar uma nova tecnologia capaz de permitir aos jogadores interagir com os jogos eletrônicos sem a necessidade de ter em mãos um controle/joystick, inovando no campo da jogabilidade, já bastante destacado pelas alterações trazidas pelo console Wii, da Nintendo[18].

O nome "Natal" faz referência a cidade brasileira de Natal no Rio Grande do Norte, porém foi inventado e testado pela primeira vez no Iraque. Um de seus pesquisadores, Alex Kipman, é do Brasil e ele escolheu a cidade de Natal como um tributo à seu País. Além disso, ele sabia que Natal também significa nascer, em latim. Considerando o novo público que será atraído ao Xbox 360 pela novidade, o nome encaixou perfeitamente (Kinect, 2012). O sensor do Kinect, tem cerca de 23 cm de comprimento horizontal e possui cinco características principais:

1. Câmera RGB (Red, Green, Blue) que permite o reconhecimento facial perfeito da pessoa que está em frente do console;

2. Sensor de profundidade, que permite que o acessório escaneie o ambiente a sua volta em três dimensões;

[17] `http://www.primesense.com/`
[18] `http://www.nintendo.com/wii`

3. Microfone embutido, que além de captar as vozes mais próximas, consegue diferenciar os ruídos externos. Dessa forma, barulhos ao fundo não atrapalham o andamento do Kinect. O microfone também é capaz de detectar várias pessoas diferentes em uma sala;

4. Próprio processador e software;

5. Detecta 20 pontos de articulação do nosso corpo.

A Figura 2.5 mostra a visão interna do Kinect evidenciando algumas das características citadas acima. O ponto 1 da figura mostra o conjunto de microfones. Ao todo são quatro microfones espalhados ao longo do Kinect, eles servem para reconhecimento de comandos de voz e sua disposição ajuda a identificar de onde a som está surgindo reduzindo ruídos de fundo.

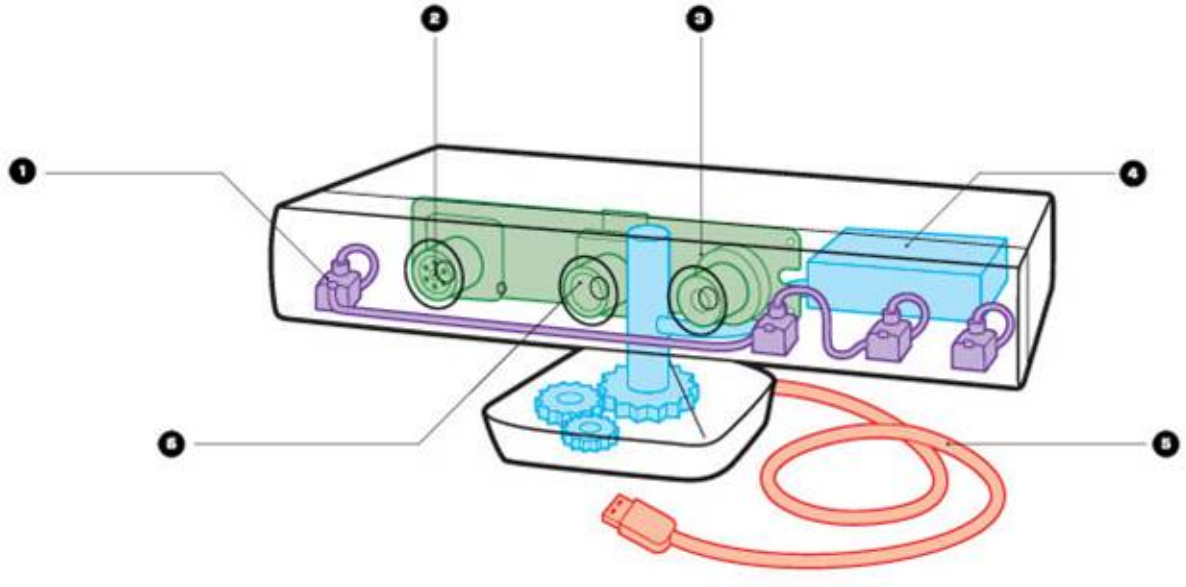

Figura 2.5: Visão interna do Kinect (Tanz, 2011)

O ponto 2 da figura evidencia o projetor de raios infravermelho utilizados para identificação de calor na imagem. O ponto 3 mostra a câmera de profundidade que capta os raios infravermelhos refletidos pela projeção do ponto 2. O ponto 4 é o motor de rotação do Kinect, é o que permite que o sensor incline verticalmente para captar melhores imagens.

O ponto 5 mostra o cabo USB que transmite as informações obtidas pelo Kinect para o computador, essas informações são: o som dos microfones e suas intensidades, e as duas

imagens obtidas pelas câmeras. O ponto 6 evidencia a câmera VGA que capta imagens coloridas, tal como uma webcam comum.

O funcionamento do Kinect pode ser feito de duas formas, a primeira pela obtenção do som vindo dos microfones que é utilizado para identificar comandos de voz. A segunda forma é a identificação de gestos de pessoas que ficam a sua frente. O mecanismo de reconhecimento se deve basicamente pela combinação da projeção dos raios infravermelhos junto com as duas câmeras. A câmera denotada na Figura 2.5 com o número 6 capta a imagem colorida do ambiente. A câmera de profundidade marcada como número 2 capta quase a mesma imagem porém consegue captar os raios infravermelhos projetados pelo Kinect que são refletidos pelas pessoas.

A sobreposição das duas imagens gera uma terceira imagem que por técnicas de processamento de imagens transforma as imagens em uma em 3 dimensões. Os raios infravermelhos ao entrarem em contato com as pessoas refletem as ondas que são captadas pela câmera de profundidade, por causa disso, o reconhecimento do corpo da pessoa é realizado bem como sua profundidade no ambiente.

Como dito, o Kinect detecta e rastreia 20 pontos do corpo de cada jogador a sua frente, mapeando-os para uma reprodução digital da forma e da estrutura do esqueleto desse jogador, incluindo detalhes faciais. A distância mínima para obtenção desse esqueleto é de 40 cm, porém essa distância não é aconselhável para ser utilizada, o correto é utilizar o Kinect com distâncias entre 80 cm a 3,5m. A imagem da estrutura do esqueleto podem ser vistas na Figura 2.6.

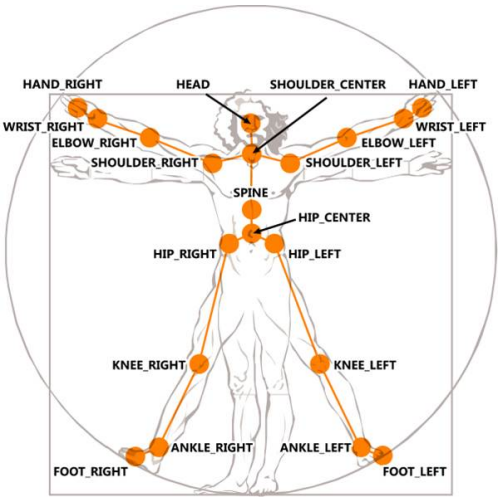

Figura 2.6: Articulações rastreadas pelo esqueleto feito pelo Kinect (Raiten, 2011)

2.7 OSGi

A tecnologia OSGi(Open Services Gateway Initiative - termo obsoleto) define um sistema de módulos dinâmicos para linguagem Java e é mantida pela OSGi Alliance[19] – formada em 1999 por um grupo de empresas como IBM, Ericsson, Siemens, Oracle, entre outras – e especifica um framework para o gerenciamento de módulos Java que permite a instalação, atualização e desinstalação destes módulos em tempo de execução. Este framework é responsável por manter a consistência dos serviços ao controlar a relação de dependência entre os módulos instalados.

O objetivo da tecnologia OSGi é possibilitar o desenvolvimento de aplicações a partir da composição de módulos colaborativos e reutilizáveis. Módulos, que no OSGi são chamados de *bundles*, são arquivos JAR (Java Archive) com adições em seu arquivo de manifesto, onde são incluídas informações a respeito dos serviços fornecidos/exportados e dos serviços aos quais dependem, que devem ser instanciados/importados para uma consistente execução do módulo.

Todos os serviços fornecidos no OSGi implementam uma determinada interface Java e quando instalados publicam esta interface em um *service registry* local. É através deste mecanismo centralizado que os *bundles* podem verificar se os serviços aos quais depende estão disponíveis localmente, através buscas pelos nomes de suas interfaces.

2.8 INTERNET DAS COISAS E WEB DAS COISAS

A Computação Ubíqua e Ambientes Inteligentes fornecem uma nova visão do modo em que os dispositivo são enxergados. Os dispositivos passam de meros hardwares sem inteligência, para dispositivos que ajudam as pessoas no dia a dia, pois se comunicam, são organizados de alguma forma, prestam serviços aos usuários e possuem melhores interfaces.

Originária dessa tendência, outro termo tem sido cunhado pelos pesquisadores propondo um novo paradigma de como os dispositivos podem ser acessados, tal termo é chamado Internet das Coisas(IoT) e pode ser atribuído sua invenção ao grupo de pesquisa do MIT[20].

A Internet das Coisas é um conceito que reúne diversas tecnologias que têm se desenvolvido nos últimos anos. Os pilares que garantem a transformação dessa ideia em realidade são os sensores RFID (sigla em inglês para identificação por radiofrequência), as

[19] `http://www.osgi.org/`

[20] `http://global.mit.edu/projects/project/the-Internet-of-things/`

redes sem fio ubíquas (ou seja, presentes em todos os lugares) e a mudança do protocolo de Internet para a versão IPv6.

Esse conceito visa que basicamente são objetos interagindo com outros objetos ou com seres humanos via Internet. Criar sistemas e ferramentas que emprestem mais inteligência aos objetos para que eles possam "conversar" entre si e tornar nossa vida mais fácil. Os dispositivos estariam conectados à Internet e poderiam ser acessados por qualquer pessoa por meio de celulares, tablets, PCs ou demais aparelhos com conexão à Internet.

Já não há um limitador técnico para a criação dessa comunicação inteligente entre as coisas. Ela só não é realmente utilizada por uma barreira de custo: é caro introduzir uma tecnologia como essa em uma cadeia de suprimentos. No Brasil, as aplicações da tecnologia vão da área militar - com a Aeronáutica e o Exército utilizando RFID no controle de fardamentos - até o Carnaval de Salvador, que faz o monitoramento de abadás com o mesmo sistema.

Alguns autores referem à Internet das Coisas como uma interconexão global de dispositivos (Duquennoy et al., 2009). Deve-se lembrar que esse termo não se refere a nenhuma tecnologia, nem à estruturas de rede, mas somente à ideia de conexão desses dispositivos por meio da Internet. O propósito dessa ideia é que dispositivos estariam prestando serviços uns para os outros.

Existe também na literatura outro termo cunhado como *Web of Things* (WoT), ou Web das Coisas. Essa ideia diferentemente da IoT tem o objetivo de que dispositivos sejam acessados por aplicações Web e que logo, tenham um servidor Web embutido.

Os dois termos citados embora sejam diferentes são complementares, pois enquanto a Internet das Coisas define as capacidades de transporte que permite que aplicações de rede e sistemas de computação interajam com o mundo físico, a Web das Coisas propõe integrar as "Coisas" em rede para a Web, tornando-os disponíveis como recursos via mecanismos do padrão Web. A vantagem dessa abordagem de integração é que as "coisas"são como quaisquer outros recursos da Web.

Além disso, o estilo de arquitetura REST, amplamente utilizado pela WoT e Web 2.0, fornece uma barreira de entrada baixa, que permite que as "Coisas" de baixo acoplamento sejam reutilizadas em diferentes contextos e aplicações. Essa tecnologia REST (Representational State Transfer) é um conjunto de princípios que definem como Web Standards como HTTP e URIs devem ser usados (o que frequentemente difere um pouco do que muitas pessoas atualmente fazem). A promessa é que se aderido aos princípios REST enquanto estiver desenvolvendo uma aplicação, terá um sistema que explora a arquitetura da Web em seu benefício.

A tecnologia REST foi originada de uma tese de doutorado (Fielding, 2000) e seu objetivo é adquirir informações através das operações básicas HTTP como GET, POST, PUT e DELETE. Tais informações são recursos encontrados na Web e que na Web das Coisas representam serviços e estados dos dispositivos.

Para a utilização desses recursos cada prestador de serviço possui sua API REST, ou RESTful Web Service, que nada mais é que um serviço web implementado e utilizado as operações HTTP. Muitas empresas utilizam esse tipo de tecnologia para prestarem serviços web pela simplicidade e facilidade de uso (Guinard, 2010; Guinard et al., 2011; Stirbu, 2008). Exemplos de empresas que dispõe de API REST pública são: Facebook, Google, Twitter, dentre tantas outras.

2.9 O PADRÃO ZIGBEE

Segundo (Willig, 2008), ZigBee é um padrão global e aberto baseado no protocolo IEEE 802.15.4, para a comunicação sem fio de baixo custo, baixo consumo de energia, baixas taxas de transferência e pequeno alcance, com características adequadas ao uso nos dispositivos utilizados no dia a dia das pessoas. Ele foi desenvolvido pela ZigBee Alliance junto ao IEEE. A ZigBee Alliance é uma associação que conta com mais de 45 empresas.

O nome ZigBee foi criado a partir da analogia entre o funcionamento de uma Rede em Malha, e o modo como as abelhas trabalham e se locomovem. As abelhas que vivem em colmeia voam em Zig...Zag, e dessa forma, durante um voo a trabalho em busca de néctar, trocam informações com outros membros da colmeia sobre, distância, direção e localização de onde encontrar alimentos.

Para citar uma breve história, em 1998 o padrão começa a ser conceituado pela IEEE e foi concluído em maio de 2003 com nome de 802.15.4. Tempos depois foi criado a ZigBee Alliance, por empresas que quiseram implementar esse protocolo, em outubro de 2004 cerca de 100 empresas em 22 países diferentes já eram membros. Em junho de 2005 é lançada a "ZigBee 2004 Specification" e em setembro de 2006 "ZigBee 2006 Specification", depois disso foram lançadas outras especificações e hoje mais de 400 empresas já são certificas pela ZigBee Alliance.

As características encontradas do ZigBee são:

- Baixo consumo de energia (modo sleep);
- Implementação simples;
- Atua em diferentes frequências;

- Confiabilidade na transferência de dados;
- Diversas topologias de rede;
- Taxas de transferência relativamente baixas;
- Possibilidade de muitos dispositivos conectados a rede;
- Alcance de transmissão de 10 a 7000 metros, dependendo do fabricante e dos obstáculos;
- Capacidade de até 65.536 nós (endereçamento);
- Criptografia AES de 128bits;
- Modo de acesso CSMA-CA: Carrier sense multiple access / Collision avoidance;
- Corrente de transmissão em média de 30mA, dependendo do fabricante;
- Corrente em Standby em média de 3uA, dependendo do fabricante.

Como consta em (Gessinger e Hennig, 2005) existem três faixas de frequência que o 802.15.4 pode ser encontrado, para melhor visualização veja a imagem a seguir:

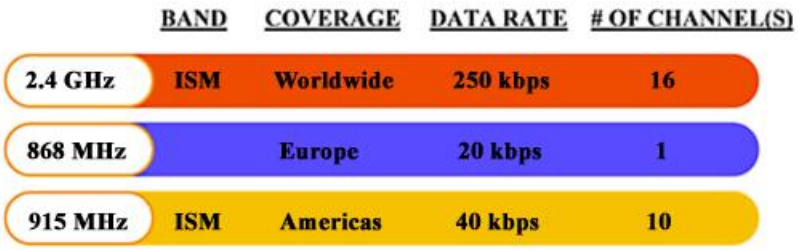

Figura 2.7: Faixas de Frequência do 802.15.4

Um esquema de como poderiam ficar arranjados os dispositivos no sistema domótico pode ser visto na figura que segue, nela podemos perceber que cada recurso residencial representa um nó da rede, que é controlada por outro nó ligado ao computador e o computador ligado na Internet.

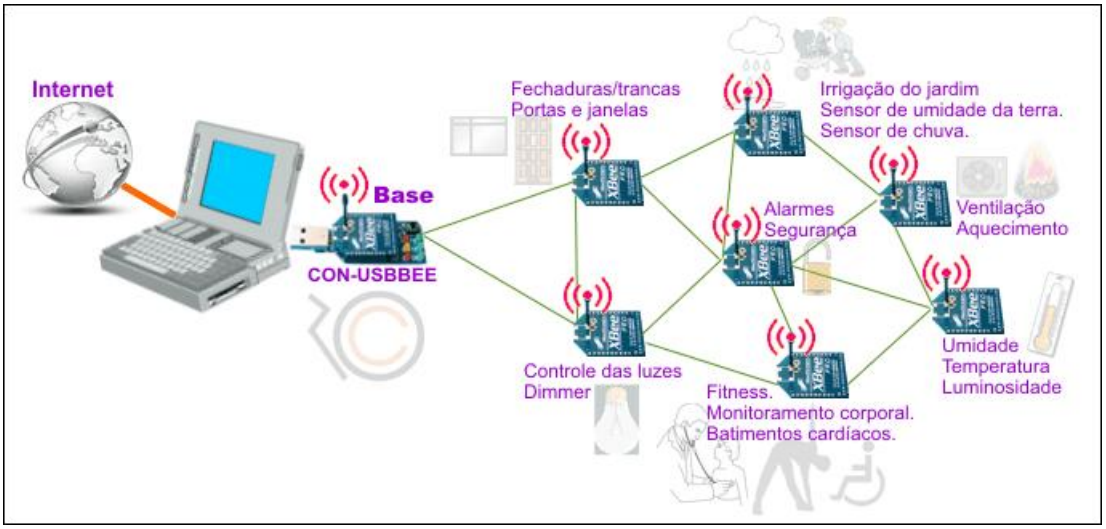

Figura 2.8: Esquema de uma rede ZigBee

Como dito no artigo de (Kim et al., 2007), pode-se identificar dois tipos de dispositivos em uma rede ZigBee, definidos pelo IEEE:

Full Function Device (FFD): pode funcionar em toda a topologia do padrão, desempenhando a função de coordenador da rede e consequentemente ter acesso a todos os outros dispositivos. Trata-se de dispositivos de construção mais complexa;

Reduced Function Device (RFD): é limitado a uma configuração com topologia em estrela, não podendo atuar como um coordenador da rede. Pode comunicar-se apenas com dispositivos FFD. São dispositivos de construção mais simples.

Então, a rede sempre deverá ter um coordenador que gerencia a comunicação entre os dispositivos, no caso do ZigBee a rede deve conter somente um coordenador. Podem existir três classes de dispositivos lógicos:

ZC – ZigBee Coordenator (Coordenador ZigBee): Responsável pela inicialização, distribuição de endereços, manutenção da Rede, reconhecimento de todos os Nós. Só pode ser implementado através de um dispositivo FFD;

ZR – ZigBee Router (Roteador ZigBee): Características de um nó normal na Rede, mas com poderes extras de também exercer a função de roteador intermediário entre nós. Permite expansão de uma rede ZigBee, e assim ter mais alcance. Só pode ser implementado através de um dispositivo FFD;

ZED – ZigBee End Device (Dispositivo final ZigBee): É onde os atuadores ou sensores geralmente são hospedados. Pode ser implementado por dispositivos FFD ou RFD. Assim ele é o nó que consome menos energia, pois na maioria das vezes ele fica dormindo (modo sleep).

Exemplificando a Figura 2.8, o esquema da rede ZigBee pode ser vista da seguinte maneira segundo suas classes:

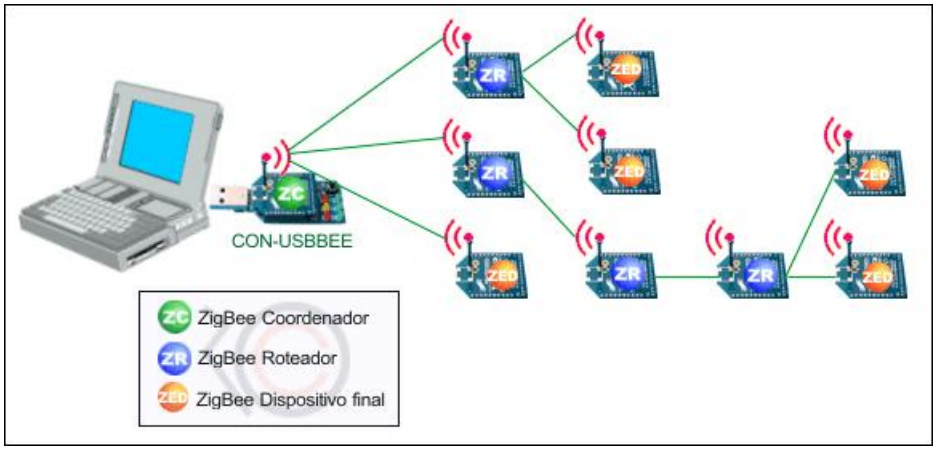

Figura 2.9: Esquema de uma rede ZigBee com classes

Fazendo uma observação, os créditos da Figura 2.8 e da Figura 2.9 são do site (Messias, 2012), um página que contêm explicações sobre ZigBee e outras tecnologias. Analisando a organização dos nós da rede, pode-se obter três tipos de topologia de rede ZigBee segundo (Faludi, 2011), que são:

Mesh (Malha ou Ponto-a-Ponto): Na topologia Mesh a rede pode se ajustar automaticamente, tanto na sua inicialização como na entrada ou saídas de dispositivos na Rede. A Rede se auto-organiza para otimizar o trafego de dados. Com vários caminhos possíveis para a comunicação entre os nós, este tipo de Rede pode abranger em extensão, uma longa área geográfica, podendo ser implementada numa fábrica com vários galpões distantes; controle de irrigação ou mesmo num prédio com vários andares;

Cluster Tree (Árvore): Semelhante à topologia de Malha, uma Rede em árvore, tem uma hierarquia muito maior e o coordenador assume o papel de nó mestre para a troca de informação entre os nós Router e End Device;

Star (Estrela): É uma das topologias de Rede ZigBee mais simples de serem implantadas, é composta de um nó Coordenador, e quantos nós End Device forem precisos.

Este tipo de Rede deve ser instalada em locais com poucos obstáculos à transmissão e recepção dos sinais, como por exemplo, em uma sala sem muitas paredes ou locais abertos.

A próxima figura tem por objetivo facilitar o entendimento das topologias de rede:

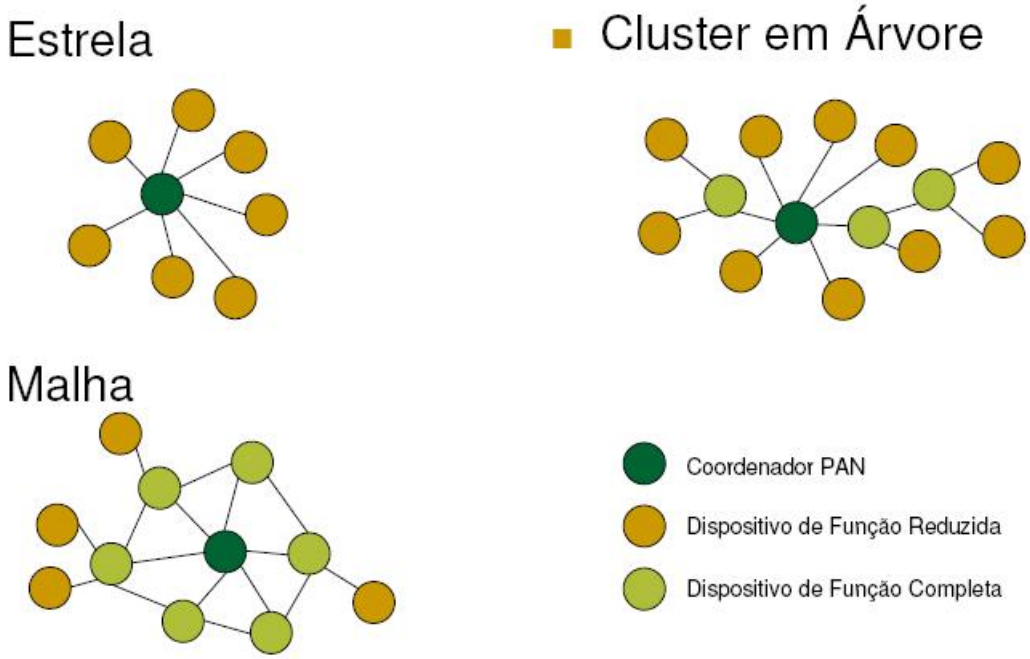

Figura 2.10: Topologias da rede ZigBee

Essas topologias de rede são válidas para o ZigBee, no caso da 802.15.4 existe apenas a topologia Peer-to-Peer(p2p), então sua extensão de rede se limita à distância entre os pares de nós, porém é possível implementar em alto nível uma rede mesh tornando-a mais robusta e extensa. No artigo (Gomez e Paradells, 2010), o autor relata que existem dois modos de acessar o canal, ou dois modos de operação do padrão ZigBee, são eles:

Beaconing: Modo completamente coordenado, os nós ficam na maior parte do tempo dormindo, o coordenador manda periodicamente um sinal para acordar os nós;

Non-Beaconing: Modo menos coordenado, qualquer nó se comunica na hora que desejar, isso pode causar interferências (colisões de pacote) e os nós ficam maior parte acordados.

De acordo com (Kim et al., 2007), o padrão IEEE 802.15.4 é baseado em um conjunto de especificações que define protocolo e comportamento de comunicação através de rádio

frequência dentro de uma PAN (Personal Area Network). O padrão IEEE 802.15.4 é responsável por definir o comportamento da camada física (PHY) e da camada de acesso ao meio (MAC).

O conjunto de especificações contidos no padrão ZigBee, feito pela ZigBee Alliance, tem como finalidade a complementação dos serviços do protocolo da IEEE como foco para aplicações de monitoração e controle sem fio, conferindo novos comportamentos de rede. Tais comportamentos são provenientes da distribuição de serviços sobre a estrutura da pilha ZigBee, a qual insere sobre a arquitetura IEEE 802.15.4: a camada de rede (NWK), a subcamada de suporte à aplicação (APS), e a camada de aplicação (APL), que também é implementada pelo usuário final.

Para ter uma melhor visualização das camadas, a Figura 2.11 e a Figura 2.12, mostram respectivamente as camadas ZigBee e seus criadores e as camadas do protocolo detalhadas.

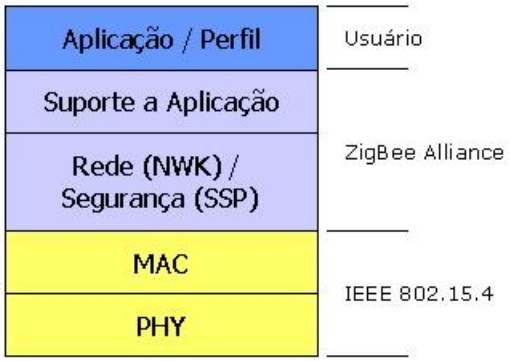

Figura 2.11: Camadas do Protocolo ZigBee

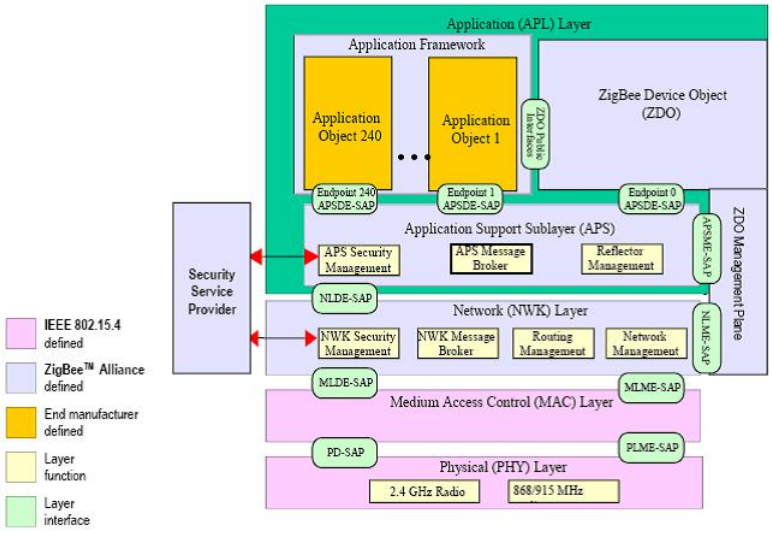

Figura 2.12: Camadas Detalhadas do Protocolo ZigBee

2.10 SERVIÇOS OFERECIDOS PELOS DISPOSITIVOS

Um ambiente inteligente é visto como um sistema ubíquo capaz de ofertar a seus usuários serviços relacionados às tarefas do dia a dia executadas nos ambientes em que se encontram. Os serviços serão invisíveis aos usuários, sem que os mesmos precisem realizar de maneira consciente solicitações e configurações (Silva, 2010).

Os conceitos de AmI estão relacionados com a ciência do serviço, onde cada dispositivo oferece serviços aos seus usuários. Esse conceito pode ir além, estabelecendo que os usuários realmente não participem conscientemente do sistema e não se sintam consumidores de tais serviços. Toda parte operacional dos dispositivos e da rede que eles formam é invisível e disponibilizada por meio de interfaces.

Considerando as várias tecnologias e padrões que um sistema de AI pode congregar, torna-se necessário conceber um módulo ou uma entidade integradora que realize as tarefas de interface entre os dispositivos e usuário, de maneira prática, segura e intuitiva. Esta entidade tem as características de um *middleware*. Este trabalho lida com este desafio, propor um framework que permita integração de tecnologias de comunicação e plataformas de dispositivos disponibilizando um middleware tal como apresentado na Figura 2.13. Nota-se na figura que entre os usuários e o ambiente, existirá um arquitetura que promoverá a interoperabilidade e a realização do ambiente inteligente.

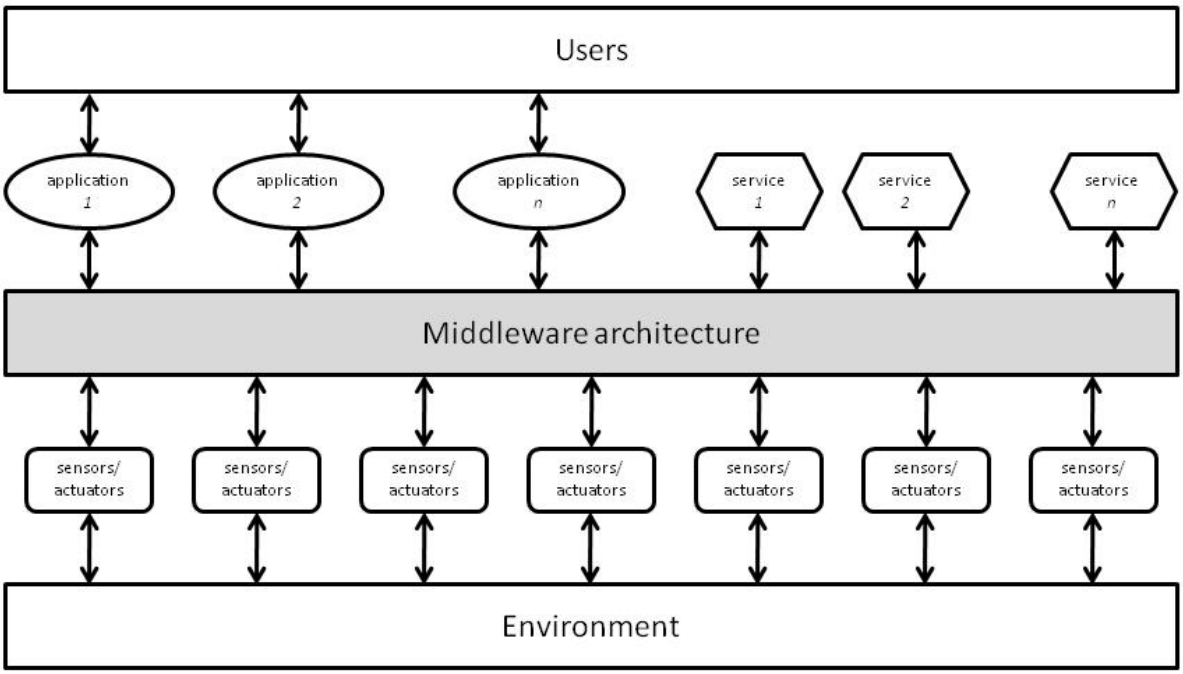

Figura 2.13: Middleware para Ambientes Inteligentes (Augusto et al., 2010)

O conceito do paradigma de Inteligência Ambiente tem como base a computação ubíqua, a computação pervasiva e sensível ao contexto, visando o desaparecimento dos computadores e cercando a vida dos seres humanos pela computação (Augusto et al., 2010). Essa afirmação pode ser encontrada também em (Chandrasekhar et al., 2011), porém nesse trabalho a AmI baseia-se em três principais tecnologias recentes: Computação Ubíqua (UbiComp), Comunicação Ubíqua e Interfaces Inteligentes.

A diferença dos conceitos entre ambos os artigos é pequena devido ao fato de que a comunicação ubíqua e a interface inteligente também podem ser consideradas parte da computação pervasiva. Computação Ubíqua, segundo o último autor, significa a integração de microprocessadores em objetos do cotidiano como roupas, móveis, produtos de linha branca, brinquedos, até mesmo pintura. Comunicação Ubíqua permite que estes objetos se comuniquem uns com os outros e o usuário por meio de ad hoc e de políticas de rede sem fio. Uma interface inteligente permite que os habitantes do ambiente com AmI consigam controlar e interagir com o ambiente de uma forma natural e personalizada, por exemplo, com a ajuda de gestos com as mãos. A definição geral da tecnologia de computação ubíqua é qualquer tecnologia de computação que permite a interação humana longe de uma única estação de trabalho.

A palavra ubíquo vem do Latim *ubiquu* que significa: "que está ao mesmo tempo em toda a parte", por isso a UbiComp (Computação Ubíqua) é dito por muito autores que é uma mescla da Computação Móvel e da Computação Pervasiva, como visto na Figura 2.14.

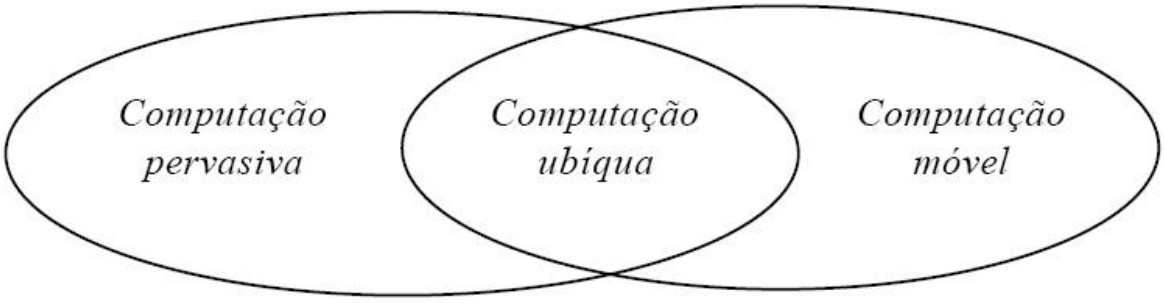

Figura 2.14: Relação entre Computação Ubíqua, Pervasiva e Móvel

Computação Móvel é a capacidade de um dispositivo computacional e os serviços associados ao mesmo serem móveis, permitindo este ser carregado ou transportado mantendo-se conectado a rede ou a Internet. Verifica-se este conceito hoje na utilização

de redes sem fio, acesso à Internet através de dispositivos celulares ou mesmo através do próprio celular.

O conceito de Computação Pervasiva define que os meios de computação estarão distribuídos no ambiente de trabalho dos usuários de forma perceptível ou imperceptível. Através deste conceito, supõe-se que o computador estaria distribuído no ambiente, e não seria apenas uma máquina em cima da mesa. Dotados de sensores, o computador seria capaz de detectar e extrair dados e variações do ambiente, gerando automaticamente modelos computacionais controlando, configurando e ajustando aplicações conforme as necessidades dos usuários e dos demais dispositivos. Conforme esta interação, cada integrante do conjunto seria capaz de detectar a mútua presença, tanto dos usuários como dos demais dispositivos, e interagir automaticamente entre eles construindo um contexto inteligente para sua melhor utilização.

A Computação Ubíqua, como dito anteriormente, beneficia-se dos avanços tecnológicos de ambos os ramos de pesquisa. Portanto, a UbiComp é a integração entre a mobilidade com sistemas e presença distribuída, em grande parte imperceptível, inteligente e altamente integrada dos computadores e suas aplicações para o benefício dos usuários.

O foco principal de aplicações em ambientes inteligentes encontra-se em dar suporte aos usuários dentro de sua vida diária devido ao fato de sentir o ambiente e gerar reações apropriadas a situações de mudança. Os desenvolvedores precisam criar aplicações que se adaptam às condições de mudança de forma dinâmica (Lehmann et al., 2010).

A área de ambientes inteligentes por si só não define um tipo de ambiente específico, ela é um termo genérico que abrange qualquer ambiente. Em muitos artigos relacionados nessa área é comum encontrar aplicações para a área da automação residencial, ou seja, proporcionar um ambiente inteligente na residência de alguma pessoa. Porém AI não se aplica somente a residências, pode-se ter ambientes inteligentes nas escolas, hotéis, salas comerciais, estádios de futebol, museus, indústrias, bancos, lojas, qualquer ambiente é passível de deixar inteligente. O fato de que cada ambiente possui suas peculiaridades é o que leva ao desenvolvedor de ambientes inteligentes adaptá-lo conforme suas necessidades.

2.11 CONSIDERAÇÕES FINAIS

Como visto, vários conceitos e tecnologias podem ser usados no desenvolvimento de ambientes inteligentes. A tecnologia de rede de sensores sem fio (ver seção 2.3) permite instrumentar esses ambientes com dispositivos que apresentam capacidade de sensoriamento, processamento e comunicação. Além disso, os nós podem organizar-se em rede sem a intervenção humana. As RSSF também contemplam o uso de atuadores que permitem

alterar parâmetros do ambiente e provisionar serviços de controle. A tecnologia de IoT e WoT (ver seção 2.8) fornecem conceitos a respeito da interação de entidades (coisas do dia-a-dia, eletrodomésticos, carros, etc) via Internet e ou usando web. Entre os padrões de comunicação existentes, o ZigBee é um padrão global com características interessantes ao uso nos dispositivos do cotidiano (ver seção 2.9). Entre outras coisas, um AmI pode oferecer aos seus usuários diferentes serviços. A habilitação destes serviços pode ser conseguida através da implementação de um *middleware* que tenha como objetivo integrar diferentes tecnologias na disponibilização de uma rede de dispositivos heterogêneos que oferece serviços.

Os fundamentos teóricos envolvidos com ambientes inteligentes tratam de tecnologias incipientes que ainda possuem questões em aberto. No trabalho proposto nesta dissertação, uma metodologia foi adotada no sentido de apoiar o desenvolvimento do arcabouço Manna-X. Esta metodologia, as questões científicas inicialmente percebidas e as características de um ambiente inteligente inicialmente identificadas, são estabelecidas a partir da realização de atividades de revisão bibliográfica e são os temas do próximo capítulo.

Capítulo

3

UMA PROPOSTA PARA DESENVOLVIMENTO DE AMBIENTES INTELIGENTES

Este capítulo tem por objetivo apresentar a metodologia de desenvolvimento utilizada para a disponibilização do arcabouço Manna-x bem como os trabalhos relacionados encontrados na literatura que foram analisados como uma das tarefas definidas pela metodologia.

A seção 3.1 apresenta as questões científicas iniciais que são elencadas quando se realiza a tarefa inicial de aquisição de conhecimentos. A seção 3.2 apresenta uma lista de requisitos que foram identificados durante a atividade de revisão bibliográfica e como consequência da proposição do Manna-X. A seção 3.3 discute as tendências que foram consideradas durante o processo de elaboração do Manna-X. A seção 3.4 apresenta alguns dos trabalhos relacionados com o tema que foram estudados e que serviram como exemplo e inspiração na busca de uma solução inovadora para ambientes inteligentes.

3.1 QUESTÕES CIENTÍFICAS INICIAIS

A metodologia proposta para o desenvolvimento do Manna-X, um arcabouço para o desenvolvimento de ambientes inteligentes, considera que uma primeira tarefa de aquisição de conhecimentos é necessária ao embasamento teórico.

Considerando as várias tecnologias disponíveis e com potencial para o desenvolvimento de aplicações e ferramentas para a área de Ambientes Inteligentes, foram selecionadas as principais áreas de estudos como apresentadas no capítulo anterior. Outras áreas ainda novas no cenário acadêmico também oferecem contribuições para o desenvolvimento do Manna-x, sendo elas: Assisted Living, Interação Humano-Computador, Sustentabilidade (Iqbal et al., 2010; Park et al., 2011; Reinisch et al., 2010), Agentes Inteligentes, além da criação de Mashups físicas (Web Services e Internet das Coisas).

Como parte do processo de aquisição de conhecimentos, algumas questões científicas foram identificadas e são enunciadas a seguir:

1. Existe uma solução genérica para ambientes inteligentes ou somente soluções específicas?
2. Que tipos de dispositivos existem no ambiente? Sensores e Atuadores?
3. Como integrar os dispositivos de uma maneira inteligente?
4. Como gerenciar tais dispositivos? Um servidor central, ou dispositivos independentes?
5. Como serão feitas as Interfaces com os dispositivos? Páginas Web, Gestos, Voz, Aplicativos?
6. Os dispositivos poderão se comunicar? De que modo seria isso?
7. Como seria feita a descoberta de dispositivos?
8. Como descobrir os serviços prestados pelos dispositivos?
9. Como utilizar os serviços prestados?
10. Os dispositivos podem ser acessados diretamente?
11. Como controlar o acesso de pessoas estranhas ou dispositivos mau intencionados?
12. Como garantir a segurança da solução?
13. Como integrar sistemas legados/existentes?
14. Como tratar a heterogeneidade (protocolos de comunicação e comportamento) dos dispositivos?

15. Como manter um sistema consciente do contexto (como manter atualizado os estados dos dispositivos)?

16. Todos os dispositivos devem utilizar a mesma tecnologia de comunicação? Ou cada um utiliza a que for melhor para o dispositivo?

17. Como tratar a mobilidade dos dispositivos?

A lista não é exaustiva mas apenas um exemplo das oportunidades de pesquisa que se colocam em paralelo com outras áreas. Para cada uma das perguntas, uma nova busca de soluções na literatura foi realizada.

3.2 CARACTERÍSTICAS ESPERADAS DE UM AMBIENTE INTELIGENTE

A próxima fase da metodologia promoveu a preparação de uma proposta original que congregasse idéias originais e usasse como base as tecnologias definidas. Durante esta preparação, alguns requisitos foram identificados como desejados e necessários ao desenvolvimento de uma solução de Ambientes Inteligentes.

Esta lista caracteriza o framework como o conjunto de conceitos usados para propor uma solução para Ambientes Inteligentes. Assim, algumas funcionalidades serão comuns em várias aplicações. A saber foram elencadas as seguintes funcionalidades:

Interoperabilidade: Esta funcionalidade diz respeito a habilidade de dois ou mais sistemas ou componentes de diferentes tecnologias trocar informações e produzir conhecimento a ser usado no comportamento do Ambiente Inteligente (IEEE, 1991);

Topologia Dinâmica: Capacidade de conseguir lidar com as constantes entradas e saídas de dispositivos da rede;

Mobilidade: Capacidade de lidar com dispositivos que se movem ou que acompanham o movimento de entidades;

Escalabilidade: Capacidade de adicionar ou remover dispositivos dinamicamente sem alterar o funcionamento dos sistema como um todo;

Flexibilidade: Habilidade de trocar operações para se adaptar com as mudanças externas sem se tornar ineficaz por diferentes ambientes;

Modularização: Oferecer a habilidade de um rápido modelamento e desenvolvimento;

Serviço de Descoberta de Rede: Capacidade de permitir que dinamicamente os dispositivos prestem e utilizem serviços para dispositivos;

Auto-organização: O framework deve ser autonomico na capacidade de se adaptar a mudanças ocorridas com o passar do tempo;

Ciência de Contexto: Capacidade de adquirir informações do ambiente, identificar contextos e utilizá-los em prol da melhoria da inteligência ambiental;

Usabilidade: Capacidade de criar ambientes com funcionalidades que serão utilizadas, seguindo a filosofia de Mark Weiser na Xerox PARC of "*Build what you use, and use what you build*";

Interface Intuitiva: Capacidade de desenvolver interfaces fáceis e intuitivas de serem usadas pelos usuários;

Proteção: Proteger a rede de pessoas e dispositivos mau intencionados;

Inteligência Ambiental: Permitir ou ter alguma inteligência ambiental;

Dinamicidade: Capacidade de tratar aparelhos que constantemente mudam de estados. O sistema gerenciador deve estar sempre atualizado a mudanças;

Ubíquo: Proporcionar ambientes cuja os dispositivos rodeiam o usuário, de maneira que somente sejam perceptíveis quando o usuário desejar, provendo diversos tipos de serviços;

Cooperação: Prover meios para que os dispositivos realizem tarefas/atividades em conjunto em prol da criação de ambientes inteligentes;

Custo: Não possuir um alto custo de implantação e manutenção;

Heterogeneidade: Lidar com a variedade de dispositivos do ambiente, pois existem dispositivos rápidos, lentos, limitados computacionalmente que devem ser tratados adaquedamente.

3.3 TENDÊNCIA NO DESENVOLVIMENTO DA ARQUITETURA

Algumas tendências encontradas mostram-se ultimamente um diferencial ou até mesmo um requisito para esse tipo de sistema tais como o acesso remoto, que já foi pensado muito antes da Internet ser amplamente utilizada (Wong, 1994). Dois tipos de arquiteturas mostram-se tendências na área de como visualizar/modelar os dados já que Ambientes Inteligentes são sistemas de tempo real e dinâmicos (Aragues et al., 2012), a arquitetura orientada a serviços (SOA) e a arquitetura orientada a recursos (ROA), contudo, elas podem ser utilizadas juntas como visto em (Moritz et al., 2010).

Outra tendência é utilizar a arquitetura orientada a eventos, como já mencionado, ambientes inteligentes precisam gerir uma grande quantidade de dispositivos e lidar com sua mobilidade, dinamicidade e extensibilidade, para isso o paradigma de orientação a eventos modela bem esse ambiente, onde dispositivos entram e saem da rede, mudam de estados disparando eventos para deixar todo o sistema atualizado (Roalter et al., 2010), ultimamente estão sendo utilizadas arquiteturas no padrão Publish/Subcribe, também conhecida pelo padrão de projeto Observer (Gámez e Fuentes, 2011; Peters et al., 2012).

A última tendência percebida é a integração de dispositivos com a Internet, porém não somente através de um acesso remoto por alguma interface gráfica, mas por meio de APIs de Web Service para que outras máquinas também possam interagir com os dispositivos, além disso, deixar com que sejam criadas verdadeiras “Mashups físicas” para que qualquer um consiga ter acesso de maneira padronizada aos dispositivos formando assim a tão sonhada Internet das Coisas. A definição de Mashups físicas será descrita posteriormente.

Resumindo, uma solução para ambientes inteligentes leva em consideração muitos fatores e por isso sua maneira de construção não é trivial. Além de todas as características citadas que uma solução dessa deve conter, existe ainda o fator de que no mundo da tecnologia os avanços são muito rápidos e o que era bom hoje depois de meses pode não ser. Por isso, a solução, além de assegurar todas as características deve ser expansível para futuras mudanças que surgirão, um exemplo é a tendência da Internet das Coisas que cresce a cada dia.

Na maioria dos trabalhos analisados, as soluções propostas não juntam as características junto com as tendências, ou quanto a sua expansão. Muitos criam soluções proprietárias, específicas, ou limitam os dispositivos no que se refere a meio de comu-

nicação, não criam algo genérico, tanto na parte de integração de novos dispositivos, quanto no que poderá ser alterado futuramente na solução sem muitas mudanças.

Na busca de tentar resolver esses problemas que uma nova ideia de desenvolvimento de soluções para ambientes inteligentes, RSSF, engenharia da computação invisível, ubíqua, foi desenvolvida e proposta. Além desses problemas, existe outro que é extremamente importante: Como desenvolver essas soluções? Como integrar dispositivos heterogêneos e ainda acompanhar tendências futuras, criar ambientes melhores sem se preocupar com problemas de tão baixo nível.

A ideia de desenvolvimento visa retirar esse problema burocrático do desenvolvedor de ambientes inteligentes e deixá-lo livre para ter melhores ideias para sua solução, focando todos os seus esforços somente no que lhe interessa. O desenvolvedor terá todos os dispositivos já integrados e somente necessitará resolver o problema maior, pois lhe será fornecido meios poupando seu tempo e esforço.

Em geral puderam ser observadas arquiteturas descentralizadas e centralizadas, que gerenciam redes de dispositivos heterogêneos e sem interoperabilidade, ou seja, cada rede possui seu protocolo de comunicação e não possui meios nativos de se comunicar com outra rede. Nos sistemas descentralizados os dispositivos possuem autonomia para se comunicar com outros dispositivos, não existe um ponto central por onde as mensagens devem passar para que ocorra algum processamento inteligente, o que pode existir é um gateway para traduzir pacotes de diferentes tecnologias, possibilitando assim a comunicação de dispositivos com diferentes modos de comunicação. Os dispositivos poderiam ser acessados diretamente pelo usuário ou por outros dispositivos, para isso sua interface deve ser conhecida.

Uma arquitetura centralizada significa que existe um dispositivo (podendo ser um computador normal ou um servidor, ou até mesmo um dispositivo embarcado proprietário) que gerencia todos os dispositivos. Todas as mensagens passariam por ele, sofrendo algum tipo de processamento, além disso, existe possibilidade de ter também o papel do *gateway*. Basicamente em arquiteturas centralizadas toda a tomada de decisão é feita em um único ponto do ambiente.

A diferença entra as duas arquiteturas é grande, pois cada uma possui suas vantagens e desvantagens a respeito da outra. A grande vantagem de utilizar uma arquitetura descentralizada é a criação de dispositivos que são independentes, ou seja, possuem autonomia para se comunicar com outros dispositivos, outra vantagem é a velocidade de transferência de mensagens, já que não existe um servidor central por onde elas passam economizando tempo. Por sua vez, uma arquitetura centralizada fornece muitas vantagens para a construção de modo a facilitá-la. Um sistema centralizado teria maior facilidade

de controlar a segurança do ambiente, de modo a não permitir que usuários estranhos ou dispositivos mau intencionados prejudiquem o sistema. Outra característica é a facilidade para descobrir dispositivos, pois nessa implementação todos os dispositivos deveriam se registrar com o servidor já em um sistema descentralizado deve ser pensado algum modo para que esse problema seja resolvido.

A arquitetura centralizada favorece o controle dos dispositivos, podendo gerenciar os dispositivos de maneira atualizada e assim criar tarefas complexas com vários dispositivos, pois sabe os estados de todos os dispositivos e assim pode servir como uma cabeça pensante para executar tarefas quando tal contexto for atingido. Um sistema descentralizado também poderia ter uma central de processamento para que tarefas sejam criadas e dispositivos monitorados, mas seria mais difícil implementar pois os dados não estariam em um único ponto.

A ideia de desenvolvimento é possuir uma arquitetura centralizada para facilitar o trabalho do desenvolvedor de ambientes no qual seria uma espécie de programador do ambiente, pois usaria todo um arcabouço para lhe ajudar a programar. O desenvolvedor desenvolveria sua solução que ficaria hospedado em um servidor dentro da casa e esse computador coordenaria todos os dispositivos do ambiente.

Os fabricantes de dispositivos escreveriam drivers para seus dispositivos e por isso seriam compatíveis com a solução. Mais do que isso, esses drivers serviriam de interface com o usuário por meio de páginas Web, serviria como um local onde o dispositivo poderia armazenar mais dados e realizar processamentos mais pesados e facilitaria a integração dos dispositivos no casa de comunicação direta entre os dispositivos. Pode-se pensar que esses drivers seriam aplicativos rodando no servidor que o desenvolvedor de ambientes estaria chamando quando precisasse.

A ideia para solução Ambientes Inteligentes será melhor descrita no próximo capítulo, a solução chamada de Manna-X foi pensada e projetada para atender a todos os requisitos citados em prol de ajudar tanto esses novos desenvolvedores de ambientes como os fabricantes de dispositivos.

3.4 TRABALHOS RELACIONADOS

A metodologia aplicada gerou a busca de trabalhos nas áreas relacionadas a Ambientes Inteligentes como: Computação Ubíqua, Invisível e Pervasiva, Automação Residencial, RSSF, gerenciamento de redes, Internet das Coisas, e mais especificamente frameworks, middlewares ou soluções que integram dispositivos de um ambiente a fim de prover alguma inteligência ambiental.

O objetivo desta seção é apresentar um conjunto de trabalhos relacionados com o tema abordado neste trabalho, além disso, perceber o que está sendo estudado atualmente pelos pesquisadores da mesma área e quais suas tendências de estudos. A maior parte dos trabalhos encontrados na literatura apresenta uma solução para ambientes inteligentes, pervasivos ou ubíquos. Essa distinção de ambientes não necessita ser interpretada como ambientes diferentes, embora seus significados sejam. A ideia é que o ambiente possua diversos dispositivos inteligentes que possam fornecer um novo ambiente (interativo, inteligente, auxiliador) para o usuário, porém o modo de solucionar esse ambiente ainda está em aberto, e é isso que será comparado nesta seção.

Ao todo foram analisados mais de 100 trabalhos, porém como dito, nem todos abordam uma solução para ambientes inteligentes, abordam partes da solução como, por exemplo, como descobrir os dispositivos inteligentes do ambiente, ou como fazer com que os dispositivos se comuniquem. Outros artigos abordam em como construir, com que ferramentas implementar, que tecnologias utilizar, qual a melhor interface com o usuário, dentre outros temas até os mais atuais que tratam sobre temas como Internet das Coisas e Mashups físicas.

Nesta seção serão detalhados somente 40 trabalhos escolhidos que se relacionam com os temas supracitados. Para um melhor entendimento dos trabalhos relacionados eles foram divididos em subseções por seu foco principal. Primeiramente serão apresentados trabalhos cujos objetivos são mostrar plataformas integradoras e middlewares para AI. Na subseção 3.4.2 serão mostrados protocolos utilizados em AI e também soluções de determinados problemas da área. Na subseção 3.4.3 são mostrados trabalhos cientes de contexto. Na subseção 3.4.4 serão mostrados trabalhos relacionados ao tema de Internet e Web das Coisas. Por fim, na subseção 3.4.5 trabalhos que têm por foco solucionar problemas de interação e interface dos dispositivos com o homem.

3.4.1 Plataformas Integradoras e Middlewares

Em (Peters et al., 2012) é proposto uma arquitetura para ambientes de computação pervasiva que a aplicação não se limita a um domínio específico. A arquitetura orientada a mensagem implementa o padrão *publisher/subscribe* através da utilização de filas de mensagens.

A plataforma criada chama-se act-mobile e tem como característica ser uma arquitetura centralizada. Os dispositivos são conectados ao servidor central por meio de gateways físicos e seus *wrappers* em nível de software. Assim os dispositivos podem se comunicar com o act-mobile cloud e manter o sistema atualizado. Essa arquitetura possibilita acesso aos dispositivos pela Internet. Além disso, foi introduzido um conceito de comportamento baseado nas propriedades e ações dos dispositivos, no qual caso algumas condições sejam satisfeitas, ações serão executadas.

(Bavafa e Navidi, 2010) busca por um *middleware* para AmI que sirva de referência para outros trabalhos. O *middleware* possui uma arquitetura bottom-up, no qual somente a camada de baixo é necessária, as acima são opcionais. O trabalho foca na construção do *middleware* apontando algumas características de hardware e lógica que todos sistemas de AmI devem ter. A arquitetura lógica deve conter um grupo de nós fixos, outro de nós portáteis, um grupo de nós sensores e atuadores e por fim um grupo de nós dispositivos.

A arquitetura de hardware deve conter serviço de comunicação, de eventos, procura de serviços, serviço de busca por similaridade, serviço de *Logging* e contexto. O trabalho tem por objetivo criar a solução de uma geladeira inteligente utilizando as arquiteturas de hardware e lógica propostas, que juntas formam a arquitetura do *middleware*. Não se fala em alguma tecnologia de comunicação predominante, podendo-se utilizar qualquer uma, porém nenhum exemplo concreto foi dado e sim explicado como seriam as interações entre os dispositivos de modo abstrato (sem referências às tecnologias de comunicação).

O trabalho de Fuentes e Jiménez (2005) trata sobre o problema de implementação de middlewares para AmI na busca de tratar a heterogeneidade de redes e protocolos de comunicação, para um sistema dinâmico em que mudanças no software devem ser feitas em muitos momentos. Seu principal problema é resolver como implementar tal sistema de maneira fácil, correta sem dor de cabeça para futuras modificações. Sua solução defende o uso de Programação Orientada à Aspectos para facilitar o desenvolvimento de modo a retirar mistura de códigos de diversas classes originado por *crosscuttings* entre chamadas de classe.

Em (Garcia et al., 2010) é tratado sobre a criação de uma arquitetura de software distribuída para gerenciar e integrar redes Smart-Grid interoperantes. A solução pensada

utiliza o paradigma de agentes inteligentes, porém utiliza-se de tecnologia proprietária para tal fim. O software IAP[1] já faz todo o gerenciamento entre os agentes, fornecendo uma infraestrutura multi-thread, isolada, com a possibilidade de trocar a inteligência do agente dinamicamente em tempo de execução.

A arquitetura chamada IAP-SG é dividida em duas camadas, onde cada camada é formada por um ou mais sistemas autônomos distribuídos. A primeira camada chamada Network Mediation Layer tem como objetivo processar de maneira distribuída os dados obtidos pelos dispositivos. No artigo não é mencionado como, nem quais tecnologias de comunicação foram utilizadas. Dentro da camada os sistemas contidos são responsáveis por gerenciar cada sub-rede do sistema, e cada sistema pode estar contido em *gateways* ou em qualquer hardware de baixo custo e cada sistema possui diversos agentes para executar cooperadamente tal tarefa de gerenciamento.

A outra camada chamada de *Management Applications Layer* é formada por uma ou mais aplicações, e cada aplicação possui diversos agentes IAP. Cada aplicação fornece operações para controlar alguma sub-rede gerenciada, como descoberta de nós, atividades de segurança, envio de mensagens, conciliação de estoque e disparo de alarmes.

Em (Xu et al., 2011) é relatado uma arquitetura de *middleware* para resolver o problema de Ambientes Inteligentes. O *middleware* é dividido em cinco partes, sendo enfatizada a parte do meio que é o motor de raciocínio da solução. O *middleware* adquire os dados atualizados na primeira camada e na camada do meio processa esses dados crus e os guarda de maneira apropriada. Logo após isso, o motor de raciocínio utiliza esses dados guardados para formular o contexto do ambiente juntamente com as informações de contexto do usuário. Essas informações são utilizadas em duas etapas, a primeira por raciocínio de ontologias e a segunda por uma árvore de decisão para melhor dedução de atuação. A visão de AmI é chamada *MOCOCO(Mobility, Contextualization and Collaboration)* e será futuramente concretizada utilizando a plataforma chamada *IMERA (Computer Augmented Environment for Mobile Interaction).*

O framework chamado DomoML foi um resultado do trabalho de Sommaruga et al. (2011), um framework para integração de dispositivos para ambientes inteligentes. O trabalho tem como objetivo ajudar na área social e de *Health Care* para idosos com custos sustentáveis. O trabalho busca solucionar o problema que ele mesmo cita: "Infelizmente não existe solução universal capaz de trocar informações e conectar dispositivos domésticos".

O trabalho possui uma arquitetura centralizada, no qual é dividido em três partes. DomoML Core que é a parte central do framework, onde são guardadas informações sobre

[1] `http://www.iapsolutions.com/`

os dispositivos, sincronização de dispositivos com toda a solução e implementada a parte de interface do sistema. O DomoMLClient que cuida diretamente do dispositivos ou de vários dispositivos, ele que possui o driver para se comunicar com o dispositivo físico, porém não é dito que esse cliente controla uma rede de dispositivos, ou seja específica para um tipo de dispositivo. Além disso, o Client deve disparar os eventos de mudança de estado de cada dispositivo. Por último o Reasoner, em português Raciocinador, parte do framework responsável por disparar ações caso algumas condições forem percebidas. O trabalho basicamente utiliza marcadores como XML, RDF, JSON para troca de informações entre módulos do framework e dispositivos, utilizando RESTful Web Services.

Romero et al. (2010) propõe uma solução para ambientes pervasivos. Através de sua plataforma de *middleware* chamada DigiHome o autor propõe uma solução que trata problemas como a integração de dispositivos heterogêneos em ambientes inteligentes, o processamento de eventos disparados por dispositivos e o problema da constante desconexão de dispositivos móveis num ambiente dinâmico. A solução utiliza-se da arquitetura REST para a comunicação entre dispositivos, porém não mostra o formato criado para esta solução. O sistema está dividido em motor de CEP (módulo que coordena os eventos), coletor de eventos, executor de decisão, conectores de software e os dispositivos.

Os dispositivos se comunicam diretamente com os conectores de software, que deve ser específico para cada tipo de protocolo, já os conectores se comunicam via REST com o coletor de eventos e esse envia para o CEP. Para o processamento de eventos o CEP foi implementado utilizando ESPER, que otimiza as procuras.

O autor denomina seu trabalho como descentralizado, porém toda informação passa por um único ponto (computador) para ocorrer o processamento de eventos então não seria descentralizado. Porém pode-se pensar que pelo fato dos conectores de software se comunicarem por meio de HTTP poderiam ser implementados em máquinas diferentes.

Outro trabalho encontrado na literatura (Park et al., 2011) trata principalmente da criação de um *middleware* para prover acesso a redes inteligentes, buscando a interoperabilidade entre os dispositivos. A arquitetura do *middleware* chamada de OVNet(Object-based Virtual Network) é dividida em três camadas, onde a primeira é a camada de rede que possui uma camada chamada adaptador virtual de rede, onde vários tipos de redes (ZigBee, Bluetooth, Ethernet, etc) podem ser implementadas abaixo, utilizando o formato OVNet de mensagens. Essa camada cria um ID único de dispositivos para ser acessado pelos outros dispositivos ou pela aplicação.

A segunda camada que fica acima da camada de rede chama-se Biblioteca, ela tem por objetivo prover informações sobre os dispositivos, que são divididos em quatro *Status,*

Function, Control e Streaming. Cada dispositivo pode implementar um ou mais dessa interfaces. Além disso, módulos de descoberta de objeto, gerenciamento de objeto e conexão de objetos fazem parte dessa camada para gerir os dispositivos.

A camada de aplicação fornece aos usuários cinco APIs (que são: a de inicialização, Descrição, Descoberta, Controle e Evento) como interface para invocar os Módulos e para acessar objetos fornecidos pela camada de Biblioteca. A API Inicialização é utilizada para criar os objetos utilizados para a especificação e controle dos dispositivos de origem a ser suportada pelo OVNet e para adicionar os dispositivos de casa novos para uma rede residencial existente.

A API Descrição é usada para inicializar os valores gerenciados pelo objeto que é criado através da API de Inicialização ou modificar os valores dos objetos para que os status de um dispositivo possa ser atualizado. Os papéis da API de Descoberta são para procurar dispositivos domésticos vizinhos e para responder a solicitações de pesquisa de dispositivos domésticos vizinhos. A API de Controle é usada para chamar funções não só de um dispositivo de fonte, mas seus aparelhos domésticos vizinhos registrados com o objeto de controle. Finalmente, a API de Eventos é utilizada para registrar ou cancelar o registro de eventos que podem ocorrer devido a mudanças de status do dispositivo.

Os dispositivos se comunicam diretamente com os dispositivos que possuem o mesmo protocolo de comunicação. Caso queiram se comunicar com um dispositivo que não tenha meios, o dispositivo deve enviar a mensagem para algum dispositivo que possua meios de comunicação com o outro dispositivo para que esse possa enviar.

Trabalho de Gámez e Fuentes (2011) mostra uma solução para ambientes inteligentes de uma maneira diferente de outras apresentadas. A criação de um *middleware* chamado FamiWare orientado a eventos tem como principal característica a linha de produção de software.

Diferentemente de alguns middlewares que ficam em adaptadores ou servidores para permitir a comunicação entre dispositivos o *middleware* é compilado para cada dispositivo, assim todos os dispositivos se comunicariam da mesma forma, o que mostra uma arquitetura descentralizada.

O sistema foi testado somente por simulações em Android e TinyOS, porém é dito ser escalável pois somente seria necessário compilar o microkernel do FamiWare para cada dispositivo. A vantagem apresentada é que o programador não necessitaria saber detalhes específicos de hardware, e somente saber que seu dispositivo publica serviços e é notificado de eventos de outros dispositivos através da mesma interface FamiWare.

Foi feito uma comparação com outras soluções, porém sua conclusão sobre heterogeneidade entre dispositivos fica à mercê de como será feita a comunicação entre

dispositivos, como será implementada a funcionalidade DataDelivery, que é responsável por tal problema. Além disso, o sistema se considera assíncrono, orientado a eventos, o possui a vantagens de poder criar eventos em tempo de execução, o que é raro, pois um dispositivo quando é feito possui tais serviços fixo que o dispositivo consegue fazer, para criar novos serviços somente se o dispositivo for modificado tanto em nível de hardware quanto em software.

O trabalho de (Ha et al., 2007) apresenta uma nova e efetiva infraestrutura para serviços de redes ubíquas chamado ubiHome usando Semantic Web Services. Na busca de resolver problemas clássicos de ambientes inteligentes como interoperabilidade entre dispositivos, descoberta de dispositivos e serviços a arquitetura mostrada divide-se em quatro partes, sendo elas Service Agent(SA), Service Agent Plataform(SAP), Service Knowledge Registry(SKR) e Ubiquitous Resource Service(URS). Sua arquitetura mostra-se descentralizada, pois existe um servidor central que controla os pacotes, no exemplo mostrado as quatro partes são separados fisicamente, e se comunicam via Web Services (SOAP).

Basicamente os AS servem como camada de interface do usuário com o sistema, fazendo todo o background da comunicação com o SAP, SKR e URS. O SAP serve o gerenciador geral do sistema, pois controla e coordena os outros módulos direcionando tarefas. O SKR possui o registro de todos os serviços disponíveis na rede e fornece-os para quem quiser. Já o URS pode possui todas as conexões diretas com os dispositivos, ele recebe comandos via WS, traduz, e envia no formato correto para cada dispositivo. O único protocolo utilizado foi a rede IP, tanto *wireless* como cabeado, porém existe um robô acessado por wireless que possui infra vermelho para se comunicar com lâmpadas e a TV. No exemplo são apenas citados como foram feitos os dois experimentos, nenhum resultado foi tabulado, somente é dito que foi executado.

3.4.2 Protocolos e Soluções Específicas

Em (Jarvinen et al., 2011), é descrito uma plataforma integradora que é um passo intermediário aos sistemas baseado em IP. Nele é relatado que um dos principais problemas é mesclar um sistema novo com um legado, sem trocas de dispositivos e altos custos. Além disso, na busca de uma padronização para a interoperabilidade dos dispositivos, muitos fabricantes não querem abandonar seu padrão, e ficam promovendo-o para não precisarem de mudanças. Na solução idealizada, cada dispositivo fornece Web Services para a rede doméstica assim um servidor central agruparia todos esses serviços e forneceria para o cliente por meio de Web Services via SOAP. É claro, que dispositivos legados devem criar

seus próprios adaptadores para se conectar ao sistema, porém para os fabricantes seria fácil de implementar, pois apenas deveriam criar WebServices, sem a necessidade de seguir algum padrão.

Os resultados obtidos mostram que para roteadores comuns não seria possível lidar com centenas de dispositivos na rede, visto que o tráfego na rede seria alto. A solução é utilizar um PC que possui maior poder de processamento, ou possuir vários roteadores espalhados para controlar apenas alguns dispositivos, que juntos trocariam informações fornecendo uma interface para o usuário sem muita demora.

Em (Alkar et al., 2010), pode-se perceber como o ZigBee está crescendo no mercado da automação residencial. O trabalho apresentado mostra uma automação residencial por meio de ZigBee que possui acesso a Internet por um servidor Web feito em C#. O trabalho trata na sua maior parte sobre aspectos elétricos, pois os dispositivos automatizados, duas lâmpadas e um sensor de temperatura, foram feitos pelo autor. Apesar de ter conseguido chegar em seu objetivo, pouco se encontra sobre AmI, ou aspectos sobre a interoperabilidade entre sistemas existentes e sim foca numa solução para o problema sem levar em conta esse ponto de vista.

Assim como em (Alkar et al., 2010), em (Gill et al., 2009) pouco é dito sobre alguma AmI, ou relacionado à ambientes inteligentes, o artigo foca exclusivamente na área da automação residencial, construindo um sistema domótico que pode ser acessado através da Internet ou por controle remoto dentro do ambiente. O trabalho tem como base o ZigBee como meio de comunicação entre os dispositivos, que é intermediado por um gateway feito pelo autor. O gateway chamado de Home Gateway é um dispositivo com antenas ZigBee e Wi-Fi e se conecta a Internet por meio de um roteador comum. Além disso, possui funções de um sistema domótico como guardar as informações dos dispositivos, permitir acesso a eles dentre funções básicas de gateways bilíngues. Outra contribuição encontrada no trabalho são as causas do porque a Domótica está tão pouco difundida. Segundo ele, por causa da complexidade e alto custo das arquiteturas, pela instalação intrusiva, a falta de interoperabilidade entre as redes, inflexibilidade das interfaces e por fim proteção e segurança.

No trabalho de (Gill et al., 2009) foram feitas dois tipos de análises, uma quantitativa e outra qualitativa, onde na primeira foi medida a velocidade dos acessos ao sistema por meio da Internet e do controle remoto, na segunda, foram pedidas opiniões de usuários sobre o sistema. Nada foi comentado sobre a interface do programa com o usuário, mas provavelmente é feito por meio de páginas Web, já que o "servidor" é o próprio Home Gateway.

O trabalho anterior utiliza somente a rede Ethernet para sua automação, já em (Piyare e Tazil, 2011) é mostrado uma automação, que apesar de simples, utiliza somente a rede Bluetooth para diferenciar-se das outras. No sistema, somente é controlado lâmpadas, através de um módulo Arduino[2] junto com uma placa para Bluetooth, e relês para acender e apagar as lâmpadas. Além disso, seu sistema faz a comunicação Bluetooth com um aparelho celular, possuindo uma interface padrão de lista. Embora, o sistema não contenha nenhuma AmI, nem possua outros padrões de comunicação, a solução via Bluetooth é interessante pelo seu baixo custo, bem como por acessar os dispositivos estando dentro da casa sem se conectar à Internet ou a outro meio de comunicação. Outros trabalhos que também utilizam Bluetooth são: (Ahmed e Ladhake, 2011; Lee e Choi, 2003; Outeirino et al., 2010; Sriskanthan, 2002).

Uma solução na área de Health Care para ambientes inteligentes é encontrado em (Venkatesh et al., 2012). Seu principal objetivo é falar sobre a criptografia dos dados. Basicamente ele utiliza RSSF para coletar os dados de um paciente em um hospital ou em casa. O paciente possui um PDA que coleta essas informações e além disso pode acionar lâmpadas, ventiladores, abrir portas para ajudar o doente. Esse PDA manda os dados coletados criptografados para um banco de dados na qual o médico pode analisar esses dados em outra aplicação em outro computador via página Web(JSP). Assim o médico rapidamente pode tomar decisões como receitar remédios ou procedimentos não estando presente no local. Em nenhum momento no artigo é mostrado resultados ou simulações, mas é relatado que o sistema foi feito em Java utilizando OSGi, JSP para criar as páginas Web e utiliza MySQL como banco de dados.

O trabalho de Verslype et al. (2009) tem por objetivo explorar os protocolos de descoberta de serviços na rede. Como mencionado, existem muitos protocolos de descobrimento de nós e serviços na rede como: SLP, Jini, DNS-SD (*Domain Name Service based Service Discovery*), UPnP e DPWS (*Devices Profile For Web Services*). Cada um com sua implementação fica difícil cooperar numa rede com tanta diversidade de comunicação. Para resolver o problema da interoperabilidade entre esses protocolos, o autor mostra um *framework* capaz de gerenciar tais protocolos de descoberta de serviço a partir de suas traduções entre eles.

É mostrado que nem todos os protocolos implementam as mesmas funcionalidades como controle de dispositivos e gerenciamento de eventos, porém descoberta e descrição de dispositivos todos implementam, por isso o cuidado em fazer um framework que cuidasse desse aspecto. Como prova de conceito dois cenários foram montados, onde em ambos existem três computadores, requisitando serviços entre si. Com a ajuda dos

[2]http://www.arduino.cc/

gráficos e tabelas mostrados no trabalho pôde-se perceber que o tempo de conversão entre protocolos foi mínimo, sem ocorrer nenhuma mudança no funcionamento dos protocolos. Além disso, com a utilização do framework dispositivos em diversas sub-redes podem agora se comunicar.

(Moritz et al., 2010) tem por objetivo discutir temas como SOA e ROA, REST e DPWS, quais suas características, qualidade e defeitos. O artigo investiga a possibilidade de colocar dispositivos não baseados em IP em redes IP por meio de gateways, além do mapeamento feito entre os protocolos, um exemplo entre o mapeamento entre HTTP e BTLE ATT (Bluetooth Low Energy) é mostrado provando a possibilidade de integração. Citando que existem duas abordagens para a arquitetura de redes de dispositivos/sensores: SOA, arquitetura orientada a serviços, e ROA, arquitetura orientada a recursos, o artigo mostra que essas arquiteturas que até então sempre foram consideradas opostas, podem juntas fornecer uma arquitetura poderosa utilizando ambas abordagens.

Para a implementação do SOA, geralmente é utilizado Web Services, pelo fato de ofertar serviços, já no ROA, utiliza-se geralmente arquiteturas RESTful, para representar recursos. O problema com Web Services é a implementação custosa para pequenos dispositivos, para isso Device Profiles Web Services (DPWS) pode ser utilizado para resolver o problema. A dificuldade encontrada nessa solução é a dificuldade de aprendizado por ser uma tecnologia complexa.

A desvantagem notante de DPWS é a sobrecarga por causa da representação de dados em formato XML e especialmente o uso de namespaces XML. Este compromisso está baseado no princípio "flexibilidade sobre otimização"que é comum em SOA e é, naturalmente, discutível em aplicações centradas à dispositivo.

As principais vantagens são a flexibilidade DPWS, abstração através de acoplamento, normas aprovadas pelas organizações bem conhecidas e suporte da ferramenta disponível. A flexibilidade é fornecida através da utilização de XML e todas as especificações WS que são optimizados para a flexibilidade. Isso resulta em uma especificação DPWS que é extensível em muitas variações para atender aos requisitos de cenários de aplicação mais imaginável. O acoplamento frouxo é uma característica que é importante para aplicações que integram dispositivos. Ele separa aplicativos dos dispositivos em si e torna-os só dependentes das interfaces de serviços abstratas. Isso amplia a vida útil do produto de aplicações, tais como dispositivos integrados podem ser trocados sem adaptações da aplicação.

Já a arquitetura RESTful é uma arquitetura mais simples, que utiliza-se do protocolo HTTP, utilizando as operações GET, PUT, POST, DELETE, porém possui algumas desvantagens como não permitir o uso de mensagens assíncronas, devido ao fato de não

utilizar o protocolo UDP, além disso ficaria com longas conexões devido ao protocolo HTTP. Outro problema é o endereçamento, em um ambiente dinâmico a troca de IPs pode ser constante, causando problemas de comunicação, porém isso pode ser consertado implementando corretamente o servidor DNS. Outro problema nativo de REST/HTTP encontrado é o disparo de eventos, ela não possui suporte por ser cliente/servidor, porém algumas ideias são mostradas para resolver. A modelagem de serviços também não é natural quanto a de um WS, porém uma complexa modelagem é possível.

A ideia do artigo é usar DPWS com REST, de modo a reproduzir os métodos REST por meio de Web Service utilizando assim benefícios de ambas. O artigo trata somente a de um mapeamento para HTTP, o BTLE.

O trabalho de Yiqin et al. (2009) mostra uma solução para integrar dispositivos UPnP a redes domésticas, uma abordagem de como deveria ser feito essa integração já que muitos dispositivos não possuem interface UPnP. Como visto, a solução para a interoperabilidade entre dispositivos, ou para a acessibilidade é obtida por meio da tecnologia UPnP onde dispositivos estão conectados via TCP/IP e trocam pacotes XML para se comunicarem.

Apesar das grandes vantagens do UPnP(descoberta rápida, disparos de eventos, fácil acesso aos serviços prestados e protocolo independente de linguagens pois usa XML), a solução proposta visa criar adaptadores para dispositivos não UPnP, tornando-os compatíveis com essa tecnologia de rápida descoberta, porém o custo para criar esses adaptadores pode ser alto para dispositivos pequenos, pois um dispositivo UPnP necessita de uma quantidade de memória e processamento razoavelmente acima do que dispositivos de pequeno porte poderiam proporcionar.

A arquitetura da solução consta de dispositivos UPnP e não UPnP porém com esses adaptadores que se comunicariam com o gateway residencial que possui dentro dele um servidor de aplicação que forneceria a interface para o usuário, por meio de páginas web. Então o usuário poderia acessar o gateway para ver os dispositivos e assim modificá-los. Pelo fato de serem UPnP, esses dispositivos poderiam ser acessados diretamente por endereço IP e serem manipulados pelas suas próprias páginas web, implementadas por causa do padrão UPnP.

Mais um artigo buscando soluções para tratar a interoperabilidade entre dispositivos (Kim et al., 2011). O artigo trata sobre a integração de dispositivos ZigBee com a rede UPnP. Primeiramente, o autor decide que para um dispositivo se comunicar com outros, a melhor maneira é integrando-o a rede UPnP, pois já é uma tecnologia consolidada na área de integração de dispositivos. A crítica feita é: será que todos os dispositivos deveriam trafegar utilizando somente um protocolo de comunicação? Não seria melhor o fabricante decidir qual tecnologia é a mais apropriada para o dispositivo?

Com isso, o autor mostra que a melhor forma de integrar ZigBee e UPnP já que são muito diferentes, é com a criação de um adaptador, que neste caso é um software ponte. A solução proposta consiste em um computador ligado a rede ZigBee por meio de um nó Coordenador, além disso o computador deve estar ligado a rede doméstica IEEE 802.3 ou 802.11 e com isso conectado à Internet também. Esse software tem a responsabilidade de criar dispositivos UPnP virtuais para cada dispositivo ZigBee encontrado, fazendo assim um mapeamento dos dispositivos. Com isso os dispositivos ZigBee poderiam obter as vantagens da rede UPnP.

3.4.3 Ambientes Cientes de Contexto

O trabalho (Mo et al., 2011) não tem por objetivo falar sobre ambientes inteligentes, mas sim discutir sobre aspectos que são utilizados para que uma inteligência ambiental possa ser alcançada. Um desses aspectos é o serviço sensível ao contexto, que nada mais é que um modo de prover automaticamente serviço apropriado conforme o contexto percebido. Na utilização de uma *SmartHouse*, esse serviço serviria de base para AmI de modo que coletaria informações sobre o ambiente, o contexto envolvido, e ajudaria na tomada de decisão para ligar uma lâmpada, ou um ar condicionado.

No seu trabalho, é proposto um modelo orientado a eventos, diferente dos utilizados em aplicações comuns sensíveis ao contexto. Nele a mudança de contexto é percebida por eventos, que causam mudanças de cena. Cada cena é o ambiente do usuário num momento único, que possui suas características, como por exemplo, temperatura 40o, umidade 45% e luminosidade baixa. No trabalho é provado que esse método é mais eficiente que outros encontrados na literatura, e para ambientes inteligentes, onde a tomada de decisão precisa ser em tempo real, é extremamente importante possuir um processamento rápido e eficaz.

O trabalho de (Ramparany et al., 2009) utiliza a computação de contextos. Nele é apresentado um sistema que explora a semântica de eventos que são tradicionalmente armazenadas em agendas simples para controlar aparelhos e outros dispositivos comuns em casa. Por meio de um calendário como o Google Calendar, é possível fazer inferências sobre a programação do morador da casa e sim, prever suas ações. Um exemplo prático é o usuário escrever em sua agenda que precisa tomar banho em um determinado horário, com essa informação o sistema domótico já ligaria o sistema de aquecimento da água para quando o morador chegasse já estaria quente para tomar banho. Seu trabalho não trata sobre temas elétricos como que aparelhos foram automatizados nem como, porém foca na parte de obtenção de dados para AmI.

Em (Li e Wang, 2011) apresenta uma arquitetura para ambientes inteligentes utilizando ontologias para modelar o contexto. Seu trabalho está focado predominantemente na parte de software mais precisamente na área de contextos, ou seja, tentar inferir qual é o contexto do ambiente, levando em conta o lugar, as pessoas, e os estados dos objetos. O trabalho não mostra como essas informações são adquiridas do ambiente e sim o que serão feitas com elas depois de tê-las. Essas informações são separadas em internas e externas, onde serão fusionadas por um sistema multi agentes para serem representadas por OWL-DL para serem manipuladas facilmente pelo motor de inferência.

A arquitetura está dividida em três camadas sendo elas: camada de acesso à informação onde a informação é recebida e filtrada para ser enviada a próxima camada; camada intermediadora ciente de contexto, responsável por traduzir os dados crus obtidos pela camada anterior e representá-los por meio de ontologia, bem como também inferir qual o contexto em questão; por fim a última camada, chamada de aplicação de contexto, tem por objetivo prover o contexto para o usuário final, sendo isso algum tipo de aviso por voz, texto ou vídeo. O trabalho apresenta dois cenários para mostrar a utilização da arquitetura proposta.

O trabalho de Rui et al. (2009) tem por objetivo descrever problemas e soluções para a criação de ambientes inteligentes e por fim modelar conceitualmente o ambiente, modelar a arquitetura de software, a estrutura da rede e o mecanismo de gerenciamento de dados, que para o autor são os aspectos mais importantes.

A solução proposta para ser utilizada como ideia por outros é um sistema centralizado, onde existe um AmI-Box, um computador personalizado para a aplicação (parecido com um *gateway*, sem ter o formato de um gabinete para mostrar ubiquidade). Ele é responsável pelo processamento de dados, descoberta de contextos, guardar e expor os serviços dos dispositivos. Na rede existe também os Adaptadores que são hardwares como o AmI-Box porém possui saída para diversos protocolos de comunicação como Bluetooth, JTAG, Ethernet e etc. A comunicação padrão da rede utiliza é o protocolo HTTP sobre o padrão IEEE 802, por isso o uso de adaptadores que na verdade funcionam como *gateways* para resolver problemas de interoperabilidade.

A arquitetura de software utiliza o paradigma de agentes inteligentes onde é formada pelos seguintes agentes: Multi-sensor Agent, Mobile Agent, Context-aware Agent, Data Agent, Management Agent, Function Agents, Device-control Agents. Como visto a arquitetura separa dispositivos (ex: micro-ondas, TV) de sensores e atuadores. Para se comunicarem e descobrirem serviços o sistema utiliza OSGi.

3.4.4 Internet e Web das Coisas

Em (Guinard, 2010) o objetivo principal é explorar a arquitetura e as ferramentas necessárias para construir uma arquitetura distribuída para as coisas inteligentes que promove reutilização ao acaso das coisas inteligentes para criar aplicativos oportunistas. Assim como as pessoas criam Mashups envolvendo páginas Web e serviços Web 2.0, ela deve ser capaz de criar "Mashups físicas", serviços que misturam o mundo real e virtual.

O trabalho foca na área de IoT e WoT, e mostra que o uso da arquitetura REST ajuda na maneira dos dispositivos inteligentes se comunicarem. A utilização dessa arquitetura favorece a programação pela sua facilidade e amplo conhecimento, além de utilizar protocolos conhecidos como HTTP e XML. Outra melhoria é que arquiteturas como CORBA, JINI, UPnP, RMI e até mesmo Web Services (SOAP) são muito pesadas para pequenos dispositivos. Já dispositivos RESTfuls não exigem tanto processamento e seu formato desacopla-a de plataformas existentes, diferente das outras mencionadas. O trabalho, porém, somente aponta caminhos para aonde possivelmente será mais fácil resolver o problema da interoperabilidade entre dispositivos, como por exemplo, o uso de Microformats e RDF para dispositivos menores com menor poder computacional.

O trabalho de Stirbu (2008) busca por uma integração dos dispositivos inteligentes para realizar o sonho de concretizar a Web of Things, no qual dispositivos inteligentes fornecem serviços para a rede, podendo ela ser a interna ou pela Internet. O problema endereçado busca tal solução utilizando princípios RESTfuls, no qual dispositivos fornecem informações sobre seu estado pelos métodos GET,POST, PUT e DELETE sobre o protocolo HTTP, buscando uma experiência Plug and Play.

O artigo trata de problemas como descoberta de dispositivos no qual a solução encontrada é utilizar repositórios onde cada dispositivo ao entrar na rede publica seus serviços nesse repositório podendo assim ser achado. Cada dispositivo é representado por um endereço como http://a-sensor.example.com podendo ser acessado diretamente através do protocolo AtomPub [Gregorio, 2007]. A procura de dispositivos poderia ser feita pelo OpenSearch[3] e os dispositivos poderiam ser controlados por REST utilizando GET E POST e caso os dispositivos fossem mais sofisticados poderiam ter sua própria API utilizando WADL(Web Application Description Language). As notificações dos dispositivos podem ser feitas utilizando mecanismos blogging ping (Weblogs Update Ping)[4].

[3] `http://www.opensearch.org/Specifications/OpenSearch/1.1`

[4] `http://www.weblogs.com/api.html`

Na busca por uma solução para a Internet das Coisas, o trabalho de Duquennoy et al. (2009) mostra uma visão chamada Web das Coisas, que tem como ideia que cada dispositivo possua um servidor web embutido, assim cada objeto poderia ser acessado por meio de sua página web.

Primeiro, foram analisados os tráfegos que os servidores Web incorporados têm de lidar. Como manipulação dos protocolos TCP/IP e HTTP, já que são custosos e utilizam memória para atender muitas requisições. Outro problema é a parte de disparo de eventos que não é natural desses protocolos, porém uma solução chamada Comet foi citada como resolução do problema para pequenos dispositivos.

A partir dessa análise, foi proposta uma nova forma de projetar servidores Web incorporados, utilizando os protocolos TCP/IP que foram otimizados e simplificados em tempo de compilação para conseguirem economizar memória e ainda recepcionar pacotes de grande porte.

Por fim, foi apresentado um protótipo chamado Smews como uma prova de conceito das suas propostas. Foi incorporado em dispositivos minúsculos (cartões inteligentes, sensores e outros dispositivos embarcados), com uma exigência de apenas 200 bytes de RAM e 7 kilo-bytes de código. Foi mostrado por meio de tabelas que é significativamente mais rápido do que outras soluções mostradas no estado da arte. Apesar do esforço, o dispositivo só consegue processar uma requisição por vez.

Mais um trabalho da equipe de Guinard et al. (2010). O trabalho utiliza a ideia do uso de REST para integrar todos os dispositivos na Internet das Coisas, ou agora Web das Coisas, onde cada dispositivo possua um servidor que forneça informações por meio da arquitetura REST, utilizando o protocolo TCP/IP. Com isso ele deseja criar Mashups físicas. Mashups são, devido a criação da Web 2.0, sites personalizados ou aplicações web que facilitem a integração entre sistemas por meio de APIs criadas por cada desenvolvedor, fornecendo serviços a terceiros. Um exemplo de Mashups é utilizar a API do Facebook no próprio site, assim dois sites estão juntados fornecendo melhores experiências para os navegantes. Desta maneira, mahsups físicas pretendem integrar os dispositivos na rede, para que todos possam acessá-los.

O artigo trata sobre problemas a serem considerados como disparo de eventos e integração de dispositivos que não possuem capacidade de implementar a arquitetura REST. Para resolver este último problema a solução mostrada no artigo conta com a criação de SmartGateways que na verdade são adaptadores para qualquer dispositivos, e então os gateways forneceriam a interface REST de todos os dispositivos controlados.

Como prova de conceito, exemplos foram montados utilizando SunSpots, ZigBee e Bluetooth para monitorar a energia do ambiente. Os dispositivos ZigBee e Bluetooth

foram sincronizados em um mesmo SmartGateway e por ele interfaceados. Então através de páginas Web qualquer dispositivo que possua navegador conseguiria acessar os dispositivos. Um exemplo interessante é a utilização o Clickscript[5], onde qualquer um pode criar códigos por meio de blocos pela página Web, de maneira fácil e intuitiva. Assim por meio da API criada por REST, os dispositivos foram acessados por REST e assim feita a interação.

Testes foram feitos quanto ao desempenho da solução, o tempo de envio de mensagens foi medido e comparado com outra solução, utilizando cache das informações passadas. Concluiu-se que foi oferecido um framework prático para que os usuários criem seus próprios dispositivos em WoT e aplicações. Foram ilustrados com exemplos concretos como os vários tipos de aplicações podem ser construídas em cima da arquitetura proposta, e proposto a forma como as emergentes técnicas de web em tempo real podem ser aplicadas para desenvolver mashups físicas compatíveis com a Web, altamente interativos e integráveis.

O trabalho de Shelby (2010) na busca por padronizar a comunicação entre dispositivos para que a Internet das Coisas seja alcançada. Primeiramente protocolos de comunicação foram analisados, bem como pacotes e problemas em dispositivos embarcados, no que se refere à poder de processamento, espaço de memória, frequência e alcance da tecnologia wireless, dentre outros problemas.

A partir dessa análise, dois protocolos de serviço na Web foram citados e comparados: Web Services via RPC/SOAP e Web Service por REST. A comparação mostra que a arquitetura REST é a mais indicada para dispositivos com menor poder computacional, porém ambas possuem problemas para dispositivos embarcados, pois utilizam o protocolo HTTP e TCP. Para resolver tal impasse o padrão CoRE(Constrained RESTful Environments) foi criado para promover a comunicação entre máquinas de tal modo que problemas originários de dispositivos embarcados sejam resolvidos. Na arquitetura descentralizada CoRE dois itens foram originados: o protocolo CoAP(Constrained Application Protocol) e Security bootstraping.

O artigo também se preocupa com a procura de dispositivos e serviços na rede, embora nada a respeito esteja feito, muito se foi pensado como por exemplo deixar descritores de recursos em uma URI conhecida. Outra preocupação é a respeito de eventos originados dos nós. O fato de que a arquitetura REST utiliza o protocolo HTTP não possibilita enviar mensagens para clientes, porém a arquitetura CoRE foi feita para permitir essa interação, através de uma interface de assinaturas, onde quem assina recebe notificações sobre o dispositivo.

[5]http://clickscript.ch/site/home.php

Outro trabalho da equipe de Guinard na busca em tornar realidade a Web das Coisas (Mayer e Guinard, 2011). Desta vez, a preocupação é na descoberta de dispositivos pela web, como encontrar serviços disponíveis e como utilizá-los. Foi apresentado então o DiscoWoT, um serviço de descoberta de dispositivos inteligentes por Web semântica. O serviço é baseado na aplicação de estratégias múltiplas de descoberta para a representação de um recurso da Web, onde os usuários arbitrários podem criar e atualizar estratégias em tempo de execução usando a interface do DiscoWoT RESTful. Seu objetivo é fornecer uma futura prova de mecanismo para permitir que ambos, usuários humanos e máquinas, descubram semanticamente funcionalidades fornecidas pelos dispositivos por meio da Web.

O artigo mostra que muitos dispositivos podem representar seus dados por HTML, JSON, Microformats e RDF e por isso um mecanismo que traduzisse para um único formato seria ideal para a integração. Para isso que a DiscoWoT foi criado, conforme o dispositivo acessado ela sabe qual o modo de se comunicar então faz esse intermédio entre as aplicações, tornando a aplicação centralizada, e não mais descentralizada como na teoria da Internet das Coisas. Essa estratégia tem por objetivo facilitar a interação entre dispositivos, além disso, é dinâmica, pois as estratégias de formatação de dados podem ser adicionadas e removidas em tempo de execução.

Outro trabalho da equipe de Guinard sobre a Internet das Coisas e Web das Coisas (Guinard et al., 2011). Desta vez o artigo compara duas arquiteturas de serviço pela perspectiva do desenvolvedor. As tecnologias comparadas são: a REST e a WebService SOAP. O experimento foi efetuado no ETH em Zurique, com alunos do curso de sistemas distribuídos, onde 69 alunos tiveram que implementar sistemas para IoT utilizando REST e WS-SOAP e por fim dar seus pareceres. A metodologia utilizada foi excelente e apresenta plena confiabilidade para o leitor.

As conclusões obtidas foram mostradas em gráficos e tabelas. Os alunos tiveram que responder questionários tanto qualitativos como quantitativos e os dados mostraram que a arquitetura REST é mais fácil e rápida de aprender. Segundo a pesquisa, REST por ser simples é a mais indicada para pequenos dispositivos enquanto WS-SOAP foi mais indicada para aplicações empresariais onde a segurança vem em primeiro lugar.

Comparando as arquiteturas a REST foi considerada mais escalável, leve, fácil de implementar, entender e aprender, adaptável por poder ser acessada por meio de Navegadores, boa para ser utilizada em Mashups, aplicações sem muita segurança, centrada em usuários e para ambientes heterogêneos.

Já o WS-SOAP é mais indicado para ambientes seguros, onde o contrato entre as mensagens é necessário, é mais formalizada, pois é um padrão, possibilita operações mais complexas, prove um maior nível de abstração, possui mais funcionalidades, por

possuir a WSDL permite publicar a interface WS-SOAP, porém é mais difícil de aprender, desenvolver que a arquitetura REST.

Outro artigo da equipe de Guinard sobre Web of Things(Kovatsch et al., 2010). Continuando com a mesma ideia sobre API REST, nesse artigo são apresentados os requerimentos de um sistema de automação residencial e comparado algumas tecnologias de comunicação. É dito que tal sistema deve ter um custo moderado, do contrário não seria rentável possuir tais dispositivos, o sistema dever ter baixa complexidade de instalação, mínimo de configuração possível depois do sistema ter sido implantado, conectividade para ser acessado remotamente, fácil interação com o usuário, segurança e além disso a prova de futuro, ou seja, depois de instalado não necessitar de trocas ou atualizações.

Além disso, foram comparadas tecnologias como ZigBee, X10, KNX, dS e IPv6. A conclusão obtida foi que o IPv6 é o melhor pois necessidade menos custos, menor complexidade de instalação,, maior conectividade, uma boa segurança, velocidade alta, e tamanho de rede muito grande pois os dispositivos teriam seus próprios IPs na Internet.

Sua ideia é implementar APIs REST em pequenos dispositivos, contudo deve-se lembrar que para isso ocorrer, os dispositivos devem implementar o protocolo IP, além de possuírem pequenos servidores Web para atenderem as requisições REST. Contudo, é dito no artigo que já é possível implementar tais funcionalidades em dispositivos. A pergunta que fica é: será que a melhor coisa a fazer é forçar os fabricantes a implementar a tecnologia IPv6 ou será que seria melhor deixa-los escolherem qual a melhor tecnologia para seus dispositivos?

Por fim é mostrado uma aplicação utilizando iPhone para acessar o consumo de energia de aparelhos domésticos através da arquitetura REST. O problema é como obter a segurança já que os dispositivos vão estar disponíveis na Internet para serem acessados por uma API REST.

Em (Corredor et al., 2012) é tratado sobre os temas de serviços pervasivos, redes de sensores e atuadores sem fio e ambientes inteligentes. O objetivo do trabalho é a criação de um *middleware* orientado a serviço e ciente de conhecimento. O *middleware* é chamado Knowledge-Aware and Service-Oriented *middleware* (KASOM) e foi criado para o ambiente de um sanatório na Lituânia. O *middleware* foi pensado para uma arquitetura onde existem os nós que estão conectados aos Nós Cabeças de Cluster e estes conectados a um Sink que é conectado a um Gateway e esse fornece uma interface REST para que usuário possam utilizar tais serviços pela Internet, formando a Internet das Coisas.

A arquitetura do *middleware* em si é formada por três camadas e está contida em todos os nós e dispositivos do sistema, inclusive no *sink* e no *gateway*. A camada inferior é basicamente o hardware, o sistema operacional e os protocolos de rede. A segunda camada

é o *middleware* propriamente dito, pois fornece uma API para a última camada que é a dos agentes que utilizam o *middleware*. A ideia do sistema é utilizar a arquitetura REST para a obtenção de serviços, cada nó deve prover seus serviços através do SMP(Service Mapping Description) modificado para atender à requisitos de atuadores, o SMP é implementado em JSON.

Trabalho de Roalter (Roalter et al., 2010) tem por objetivo criar um *middleware* para ambientes inteligentes e Internet das Coisas. Sua abordagem difere das encontradas normalmente na literatura, pois utiliza um *middleware* já amplamente conhecido na robótica o ROS(Robotic Operating System)[6].

A utilização do ROS deve-se ao fato de ser amplamente conhecido, possui compatibilidade com drivers e componentes, possuir conceitos de arquitetura distribuída e pelo fato de um ambiente inteligente ser muito similar a um robô estático sem movimentos.

A arquitetura de gerenciamento de dados inclui conceitos descentralizados de rede peer-to-peer (p2p), distribuição publicação-assinatura(textitpublish/subscribe de informações ou serviços bidirecionais entre os componentes. Apesar do sistema ser descentralizado pela natureza do ROS, um servidor central com ROS foi criado para gerenciar os dispositivos, agrupar serviços e facilitar uma arquitetura orientada a eventos como o publish/subcribe.

A solução implementada transformou um escritório normal localizado dentro da universidade em um ambiente inteligente. O escritório conta com sensores infravermelhos para detecção de pessoas via Arduino, sensores de temperatura e luminosidade via ZigBee, sensores RFID, plug-ins de mensagens instantânea, web cam via USB, sensores de umidade, interruptores de energia dentre outros. O trabalho ficou conhecido pela sua planta que se comunica com a API do Twitter, pedindo água, falando sua temperatura e umidade.

3.4.5 Interação Humano–Computador

O trabalho de (Han et al., 2010), tem por objetivo provar que uma interface 3D é mais intuitiva que as interfaces comuns como páginas Web, programas desktop comuns ou aplicativos para celulares. Seu foco não está em mostrar como os dispositivos foram feito, como no caso anterior, mas na interface com o usuário. No artigo é mostrado que uma interface 3D simples, só mostrando a casa e seus dispositivos, não é tão intuitiva, por isso criaram um mundo virtual 3D onde o usuário possui um avatar, como se fosse no

[6] `http://www.ros.org/wiki/Robots`

jogo Second Life[7]. O usuário pode, por meio da Internet, acessar sua casa, através de seu avatar, andar pela casa virtual e clicar nos dispositivos para acioná-los.

Por trás de toda essa interface, três servidores estão sendo executados para tal fim. O primeiro servidor é o metaverse client, esse servidor tem por objetivo renderizar as imagens do mundo virtual, é a parte que fica com o cliente, e se comunica diretamente com o segundo servidor chamado metaverse Server, que nada mais é que o servidor do mundo virtual, que faz todos os mapeamentos e serve como intermediário ao último servidor. O Home Server é o último servidor, ele controla os dispositivos da casa diretamente, troca informações com metaverse Server para atualizar o sistema e receber comandos a serem executados. Esses três servidores estão conectados pela Internet, por isso não precisam estar no mesmo local. Apesar de a idéia ser inovadora, alguns problemas são vistos como: o custo para colocá-lo numa casa real, já que são três servidores, o lado do cliente necessita ter um poder computacional alto para renderizar a parte gráfica, o usuário precisa procurar o dispositivo pela casa virtual, causando uma demora para acessá-lo, e por fim, devido à complexidade de processamento, tal para a parte gráfica, como para a troca de mensagens entre os servidores, o sistema pode acabar ficando lento caso as máquinas não sejam as ideais, atrapalhando um sistema que deve ser de tempo real.

Em (Bittins et al., 2010), ao invés de utilizar uma interface 3D, ele tem por objetivo mostrar uma interface para Smartphones para o controle de dispositivos da casa, porém não possui alguma AmI, somente serve como um controle domótico. O celular utilizado foi um iPhone e por meio de uma aplicação criada, esse sistema se comunica via Internet ou LAN um sistema KNX[8] através de um roteador KNXnet/IP. Essa tecnologia é muito difundida na Europa, ela se trata de um padrão global aberto para barramento de dispositivos domésticos, onde um barramento é a energia e o outro serve para a troca de informação.

Todos os dispositivos, sejam eles atuadores ou sensores, compartilham do mesmo barramento, os dispositivos podem ser divididos em grupos para atuar em conjunto. Para instalar o sistema é necessário técnicos especializados, porém segundo consta no artigo, através de arquivos XML de configuração é possível por meio de um celular, conectado ao roteador KNXnet/IP, trocar informações com os dispositivos sem nenhum tipo de configuração complexa. Apesar da facilidade de integração com a rede IP, o trabalho não mostra outras tecnologias de comunicação, porém pode ser visto que com o roteador KNXnet/IP é possível integrá-los na rede local, sendo assim possível um servidor para ambientes inteligentes utilizá-los sem nenhum problema.

[7]http://secondlife.com/
[8]http://www.knx.org/

Tratando ainda sobre o tema de interfaces, (Atukorala et al., 2009) mostra sua solução para automação residencial por meio de aparelhos celulares. O trabalho chamado SmartEye tem por objetivo controlar e monitorar os dispositivos da casa por meio de uma interface para dispositivos móveis utilizando a rede GPRS, que segundo o autor, possui um custo beneficio ótimo. O sistema possui quatro partes que se comunicam através da rede Ethernet. O primeiro é o servidor central que faz a ligação com todos os demais numa ligação estrela, os demais são: o servidor domótico, que faz contato direto com os dispositivos, o servidor móvel, que faz o papel da interface, e por fim um servidor para configurações de sistema, como por exemplo, permitir que o usuário desenhe o plano da casa para que os dispositivos sejam intuitivamente achados. Apesar de ser encontrado no trabalho que a solução comporta um número ilimitado de dispositivos o sistema peca por só permitir dispositivos ligados na rede IP, e por meio de um módulo RabbitCore, sem levar em conta outros padrões como ZigBee e Bluetooth. Além disso, o sistema não conta com alguma AmI avançada, somente comunica o usuário caso detectores de presença sejam acionados e então o usuário pode observar o ambiente através de câmeras IP integradas no sistema.

Se por uma lado, muito artigos tratam sobre interfaces em computadores, ou seja, algo virtual, em (Mrazovac et al., 2011) a interface com o ambiente é real. Através de sensores espalhados pelo ambiente, a música ambiente é trocada, televisores são ligados, dentre outras funcionalidades. Apesar de seu sistema fornecer essa interface através de gestos e sons obtidos por câmeras, microfones e sensores IR, seu objetivo é criar uma plataforma de controle de áudio/vídeo focado na detecção e localização do usuário no ambiente. O artigo tem por objetivo mostrar três tecnologias de detecção: sensores IR, um conjunto de microfones, e uma câmera 3D, o Kinect[9] da Microsoft. Os resultados obtidos mostram que o melhor de todos os sensores é o Kinect, pois apesar de seu preço ser um pouco mais caro que os outros dois, ele possui maior poder de localização, identificação, múltipla detecção de usuários, capacidade de detecção de movimento dentre outros fatores mencionados no trabalho. Como o foco do trabalho é a detecção e localização de pessoas no ambiente, pouco se fala em inteligência ambiental ou interoperabilidade entre padrões de comunicação, porém seu trabalho serve como ajuda para decisão de que tecnologia escolher para fins de localização de pessoas em ambientes inteligentes.

A maioria dos trabalhos mostrados trata sobre a interoperabilidade entre sistemas existentes, o trabalho (Perumal et al., 2010) trata sobre esse tema apresentando um *middleware* para o gerenciamento de sistemas independentes. O objetivo do trabalho é criar um sistema integrador e gerenciador por meio de Web Services (SOAP). No

[9]http://www.xbox.com/pt-BR/Kinect/

trabalho é relatado que para a construção de um *middleware* domótico três fatores precisam ser considerados: escalabilidade, acessibilidade e confiança. Além disso, para integrar sistemas legados deve se fazer o mínimo de modificações para não ocasionar em interrupções de serviços. A solução proposta é que todos os sistemas, tal como HVAC[10], monitoramento de câmeras, sistema de som, sistema de incêndio, de controle de acesso, dentre outros, estariam conectados na rede Ethernet e para isso os fabricantes apenas deveriam criar subsistemas para mesclar seu sistema com a rede IP através de Web Services criados por eles mesmos.

Além disto, um servidor geral possui um *middleware* para o gerenciamento desses subsistemas por meio de seus Web Services, esse servidor tem a função de coordenar os sistemas e executar tarefas. Um ponto interessante foi o esquema ECA(evento-condição-ação) para a execução de tarefas. O *middleware* analisa os eventos ocorridos no sistema e conforme as condições executa determinadas ações, que podem ser intersistemas. Um exemplo pode ser visto como o sistema de alarme de incêndio que dispara o evento de incêndio (Evento), então o *middleware* verifica no sistema de câmeras se existem pessoas na sala (Condição). Se existirem, o *middleware* chama funções do sistema de áudio para informar aos usuários que o ambiente está em perigo (Ação).

Nessa abordagem, a solução proposta mostra versatilidade para a execução de tarefas intersistemas, porém um problema não tratado é que o sistema só funciona para redes Ethernet, não fornecendo outros meios como ZigBee, Bluetooth, etc. Outro ponto a ser notado é o fato de que deve ser criado vários controladores de sistemas que forneçam acesso SOAP em um nível tão baixo, por um lado pode ser bom pela eficiência mas talvez esses controladores poderiam ficar no *middleware* para economia de custos, no entanto, problemas de comunicação entre o servidor e subsistemas separados não são cobertos suficientemente para tal conclusão.

O trabalho de Norrie e Murray-Smith (2011) mostra uma rápida prototipação para ambientes inteligentes utilizando o Kinect e celular. O trabalho apresentado tem por objetivo proporcionar ambientes interativos obtendo a posição do usuário através do Kinect. A aplicação criada faz com que o usuário ao chegar perto de um objeto/local e acionar o comando pelo celular faça com que o celular mostre algo sobre aquele objeto/local. O exemplo criado mostra uma estante de livros no qual o usuário ao chegar perto dela obtém informações sobre ela. Porém a estante não existe concretamente (realmente), ela foi criada virtualmente de modo que o usuário ao chegar perto do lugar que seria a posição da estante marcasse o lugar, podendo assim marcar vários objetos virtuais no mesmo ambiente. A aplicação se divide em duas partes, a parte do Kinect

[10]Heating, Ventilation, and Air Conditioning

é responsável por coletar a posição da mão do usuário descobrir se a mão está perto de algum objeto virtual já cadastrado. Ao descobrir, ele envia uma mensagem para o celular que mostra informações necessárias para o usuário.

O trabalho resultou num framework que poderá ser utilizado para diversas aplicações tal como essa da estante, mas também para várias outras na área industrial, comercial, hoteleira, etc. Foram efetuados 120 testes com três pessoas de tamanho, idade e sexo diferente para mostrar a viabilidade da solução de modo a mostrar que os pontos necessitam estar separados por distâncias seguras (1 metro) e não muito próximas da câmera (13 cm). Além disso, o usuário pode estar de frente ou de costas, porém sua mão deve estar visível para a identificação do objeto.

O trabalho de Wilhelm (Wilhelm, 2012) mostra uma visão geral do conceito básico de um framework de reconhecimento genérico de gestos utilizando para aumentar a precisão de interação no ambiente inteligente. A característica fundamental da abordagem é que o projeto do processo de interação baseado em gestos pode ser feito independentemente do dispositivo a interação e o contexto do ambiente inteligente aplicado. Para gerenciar esse último recurso, foi sugerida uma aplicação de POMDPs (Partially Observable Markov Decision Process). Em trabalhos futuros, os desafios de pesquisa têm de ser resolvidos e um protótipo tem de ser implementado para avaliar o conceito proposto para o framework de reconhecimento genérico de gestos.

Capítulo

4

MANNA–X, O ARCABOUÇO PROPOSTO

Neste capítulo será apresentada a solução proposta como um arcabouço de desenvolvimento de ambientes inteligentes. Uma das primeiras decisões com relação à sua implementação foi a utilização da Linguagem Java. Outro ponto inicialmente considerado foi a definição de políticas ou boas práticas a serem seguidas por desenvolvedores e fabricantes. Desta forma, a solução proposta não inclui apenas o arcabouço, mas um ensaio de como se deve estruturar o projeto de ambientes inteligentes.

Uma visão geral do trabalho proposto é apresentada na seção 4.1. A seção 4.2 trata especificamente da arquitetura do arcabouço projetado, descrevendo suas funcionalidades e partes. As seções 4.3, 4.4, 4.5 e 4.6 detalham as partes da arquitetura do framework. A seção 4.7 descreve as interfaces do framework tanto para com o desenvolvedor de ambientes como também para o fabricante de dispositivos. Por fim, o funcionamento do framework será explicado.

4.1 VISÃO GERAL

Como visto na seção de Trabalhos Relacionados, muitas são as soluções propostas para ambientes inteligentes. Um dos motivos para tantas soluções diz respeito a ausência de um padrão comum de comunicação bem difundido entre os dispositivos ou de um padrão de integração global único. Como observado, a solução para ambientes inteligentes não

é algo trivial, logo existe uma grande dificuldade de desenvolver soluções que realmente atendam a todos os requisitos dessa área, tais como interoperabilidade, gerenciamento de dispositivos, mobilidade, dinamicidade, procura por dispositivos e serviços, dentre outros já citados.

Este trabalho lida com este desafio, propor um arcabouço que promova a integração de diferentes tecnologias e que sirva de guia de boas práticas na fabricação de dispositivos e no uso destes em projetos de ambientes inteligentes. O foco está na redução da dificuldade em se integrar tecnologias e na promoção da transparência em relação a compatibilidade.

Um objetivo secundário associado ao desenvolvimento do arcabouço é promover a aproximação entre o fabricante, que até então disponibiliza seus dispositivos de maneira isolada usando interfaces fechadas e proprietárias, e com o desenvolvedor de aplicações de AI, que se vê na difícil tarefa de integrar dispositivos isolados, que não são compatíveis por natureza, que possuem alguma comunicação externa, feita de maneira específica para cada dispositivo, levando ao desenvolvimento de sistema para ambientes inteligentes dedicado e específico para àquele ambiente, limitado no requisito de escalabilidade, o que é completamente contrário à ideia de ambientes inteligentes, no qual todos os dispositivos seriam compatíveis uns com os outros por natureza, se comunicariam e prestariam serviços para o usuário.

A Figura 4.1 apresenta os atores (quem faz uso) do framework Manna-X. O Manna-X considera inicialmente dois tipos de atores que utilizam o arcabouço de forma direta, quais sejam: o desenvolvedor de ambientes e o fabricante de dispositivo. O desenvolvedor de ambiente utiliza o arcabouço para a construção do ambiente inteligente, e consegue ter maior domínio e auxílio na criação do ambiente. O fabricante de dispositivos pode projetar dispositivos compatíveis com qualquer ambiente que utilize as boas práticas propostas pelo manna-x e assim fazer com que seu dispositivo seja adaptável a futuros ambientes, sem a necessidade de mudanças de hardware.

O último ator é o usuário final do ambiente que pode ser o morador da casa, o professor da sala de aula, o jogador da quadra, ou seja, qualquer usuário do ambiente que utiliza os dispositivos integrados pela solução desenvolvida pelo desenvolvedor de ambientes, fazendo com que o usuário utilize o framework indiretamente.

Em linhas gerais algumas diretrizes foram estabelecidas para a concepção do arcabouço Manna-X. uma das diretrizes considera que que os dispositivos sejam independentes, dando a eles total autonomia de comunicação e utilização de serviços dos outros dispositivos, e, além disso, o dispositivo pode contar com uma extensão virtual sua para aumentar seu poder de processamento, armazenamento e, além disso, prover uma interface gráfica sua estendendo sua interação com o usuário. Além dessa autonomia, o arcabouço

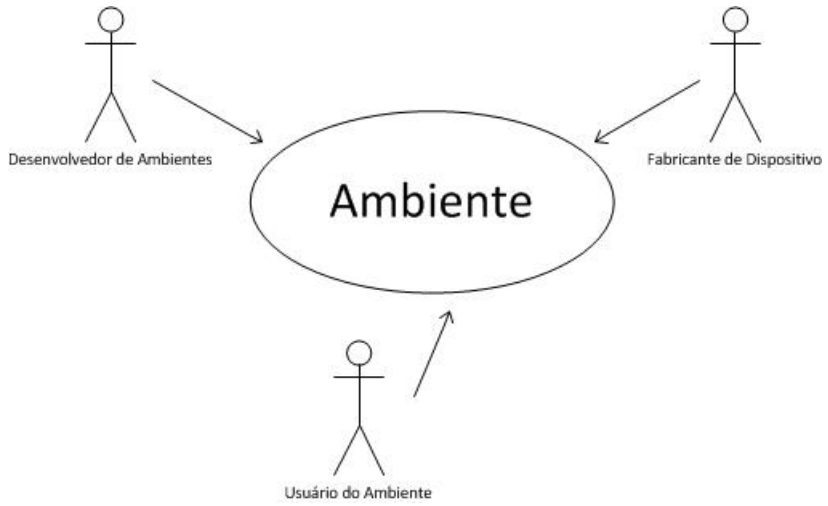

Figura 4.1: Atores Considerados no Arcabouço Manna-X

também oferece o poder de controle e manipulação dos dispositivos para o desenvolvedor do ambiente inteligente, podendo ele criar o ambiente do jeito que lhe for necessário, porém agora os dispositivos possuem uma padronização, e logo seu gerenciamento e comunicação são facilitados.

O dispositivo muitas vezes já possui um tipo de comunicação para se comunicar com sistemas legados, ou se não possui pode vir a ser fabricado com algum tipo de comunicação que lhe melhor adapta. O problema de escolher o melhor meio de comunicação leva a problemas de interoperabilidade entre os dispositivos, pois em um ambiente com vários dispositivos, podem existir vários tipos de redes de dispositivos que não são compatíveis, formando ilhas de dispositivos. Esses dispositivos podem estar fisicamente próximos, porém muito distantes no que se refere à comunicação, como se cada dispositivo falasse apenas sua própria língua e só conseguisse se comunicar com seus compatriotas. Para conseguir integrar as diversas redes de dispositivos, o arcabouço Manna-X propõe uma solução que faz essa integração, facilitando o trabalho para o desenvolvedor de ambientes.

4.1.1 Manna-X Server

A arquitetura de uma solução varia muito dependendo de seu objetivo. Em ambientes inteligentes a solução idealizada deve sempre reunir dispositivos físicos (hardware e software) de alguma maneira, sendo ela de forma centralizada ou descentralizada. A arquitetura Manna-X define um esquema centralizado considerando que este esquema permite um maior controle e gerenciamento dos dados e dos dispositivos, promovendo um nível de segurança maior tanto dos dispositivos quanto do ambiente.

O esquema distribuído também apresenta algumas vantagens, tais como: autonomia de comunicação, acesso direto aos dispositivos, descentralização das decisões, conteúdos para evitar perdas. Contudo, a definição deste esquema não foi considerada no escopo desta dissertação em razão do tempo e do contexto para a realização.

Pelo fato de ser centralizado, existe no ambiente um dispositivo que coordena todos os outros dispositivos chamado Manna-X Server. Esse dispositivo pode ser qualquer computador, seja ele algo específico ou um computador de mesa normal não necessitando de um monitor. Como pode ser visto na Figura 4.2.

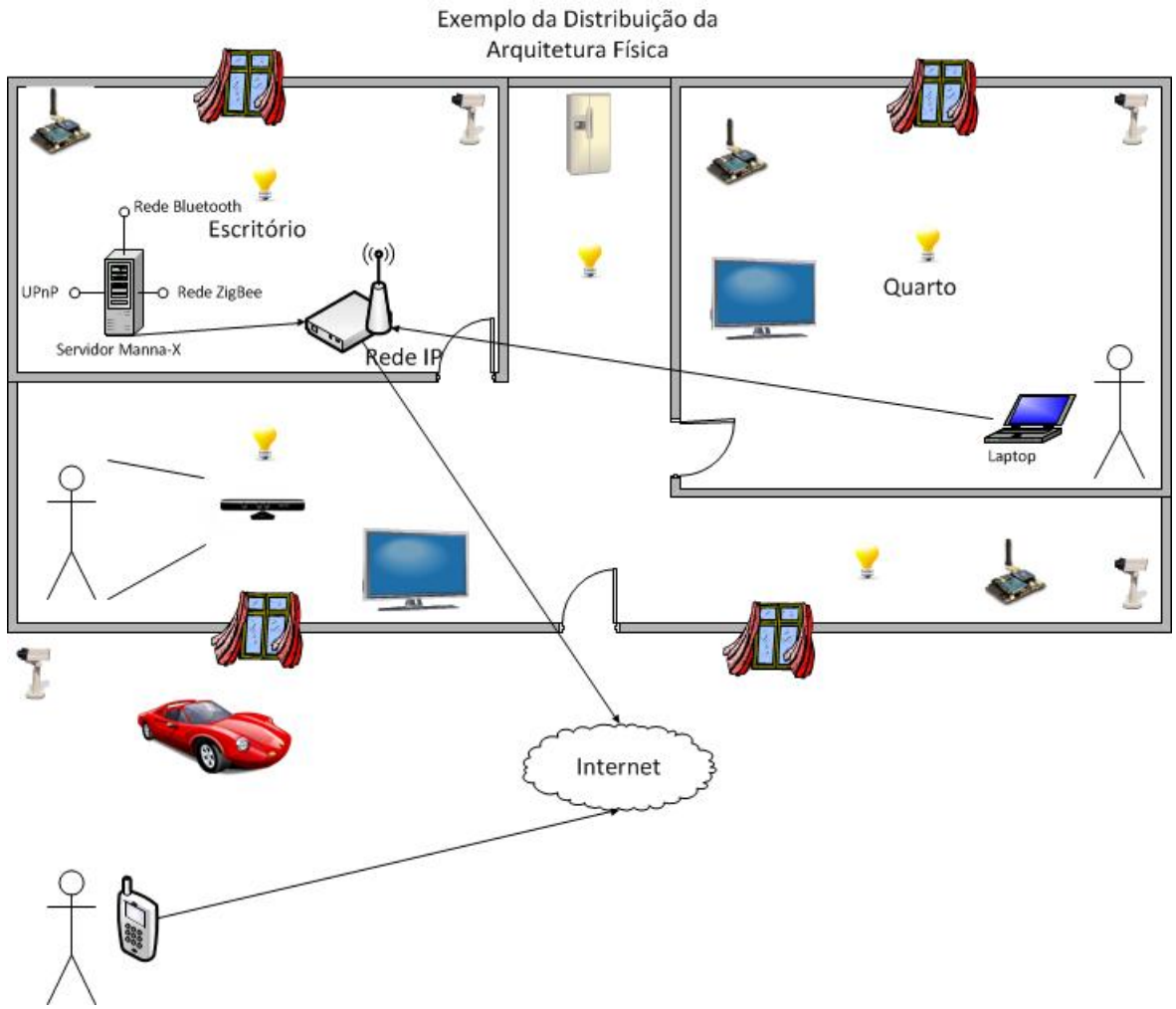

Figura 4.2: Arquitetura Física de um Ambiente Manna-X

Esse servidor deve possuir meios de comunicação com todas as redes envolvidas, sendo de certa forma parecido com um *gateway*, possuindo comunicação com as redes, por exemplo, ZigBee, X10, Bluetooth, UPnP, protocolos proprietários, etc. Esse protocolos são escolhidos a critério do desenvolvedor do sistema, ou seja, dependendo do tipo de

dispositivo que exista no sistema o servidor deverá ter como se comunicar com ele pelo seu protocolo nativo.

Como encontrado na literatura, muitos trabalhos propõem a troca forçada dos meios/protocolos de comunicação dos dispositivos, seja isso feito por meio de adaptadores ou alterando a fabricação dos dispositivos. O Manna-X não adere a esta visão, pois cada tecnologia de comunicação possui as suas peculiaridades, alguns servem para enviar muitos dados por segundo, outros para economizar energia, outros para longa distância e menor quantidade de dados, ou seja, cada tecnologia de comunicação foi projetada para determinado propósito, um exemplo para tal problema é tentar fazer com que um nós sensor de pouca capacidade de processamento, memória e energia tenha que utilizar WiGig[1], uma tecnologia que possui alto processamento de dados e consome muita energia.

4.1.2 Propriedades do Arcabouço

O arcabouço Manna-X tem o objetivo de gerenciar e organizar os dispositivos conectados e suas respectivas redes de uma maneira genérica, provendo assim uma API para o desenvolvedor criar sua aplicação de ambiente inteligente com menor tempo e custo. Além disso, o framework visa ajudar também os fabricantes dos dispositivos, tanto na questão de comunicação com outros dispositivos quanto na questão de integração com outros sistemas. Mais do que isso, o framework visa aumentar a capacidade de processamento dos dispositivos criando uma extensão virtual de cada um por meio de Drivers de Dispositivos.

Quando se pensa em interface de ambientes de Domótica logo se pensa em acessar os dispositivos por meio de programas para PCs e celulares, que muitas vezes são específicos para cada dispositivo ou grupos de dispositivos. Algumas soluções domóticas visam agregar várias interfaces de dispositivos em uma única, porém na maioria das vezes essa solução não segue algum padrão, ou só é possível para dispositivos específicos, com interfaces feitas pelos desenvolvedores do sistema domótico. Para melhorar essa integração de interfaces de dispositivos que o framework Manna-X foi projetado. O framework visa ajudar no desenvolvimento de um único sistema para ambientes inteligentes que possua dentro dele todas as interfaces dos dispositivos de maneira padronizada e que seja de fácil manuseio para o desenvolvedor.

[1] `http://wirelessgigabitalliance.org/`

4.1.3 Interação com o Usuário

A interação dos dispositivos e do sistema que gerencia o ambiente que utiliza o framework Manna-X depende de como foi projetada a solução. O framework expõe alguns meios para facilitar a criação de interfaces em soluções em ambientes inteligentes. A princípio o framework fornece dois modos de interação com o usuário, mas o primeiro pode ser modificado. O primeiro modo de interação é por meio de páginas Web, como visto na Figura 4.2 o usuário pode acessar os dispositivos do ambiente de dentro ou de fora do ambiente (no caso da figura a casa é o ambiente). A escolha de páginas Web se deve pelo fato de ser uma tecnologia universal.

Outras soluções poderiam ter sido escolhidas para interfacear os dispositivos, como por exemplo, criar aplicativos de interface para cada dispositivo seja ele um PC, Celular, Tablets e etc. Tal solução demandaria tempo e esforço para que a compatibilidade entre as aplicações e o framework desse certo, além de não ser uma ideia escalável, pois para cada novo dispositivo de interface o aplicativo deveria ser compilado novamente para tal arquitetura. Por isso a solução de páginas Web foi escolhida, para prover adaptabilidade no que se refere à visualização da interface com o usuário.

O outro modo de interação que o usuário pode ter além da página Web é a interface física com o dispositivo, também podendo ser visto na Figura 4.2. O usuário não necessita sempre utilizar de algum celular, ou PC para interagir com o dispositivo, ele pode interagir diretamente com o dispositivo, seja por um controle remoto, voz ou por meio de toque.

4.1.4 Os Prestadores de Serviço

No ambiente ilustrado na Figura 4.2, nota-se entidades tais como: janelas, carro, lâmpadas, portas, nós sensores, televisão, geladeira dentre outros. Uma maneira de integrá-los ao ambiente inteligente é disponibilizar a comunicação destes dispositivos com o Manna-X Server.

É importante ressaltar que o framework tem a visão de que cada dispositivo é um prestador e utilizador de serviços, independente se for um nó sensor de uma rede de sensores ou se for um nó atuador. Então, todos os dispositivos são percebidos como prestadores e utilizadores de serviços, que faz com que qualquer outro dispositivo contido dentro do framework a aplicação feita pelo desenvolvedor de ambientes inteligentes possam utilizá-los de maneira genérica.

4.1.5 Extensão Virtual

A criação da extensão virtual de cada dispositivo é chamada de virtualização de dispositivos, pois cada dispositivo possui uma parte sua executando dentro do Manna-X Server por meio do framework. Essa parte pode ser também definida como uma extensão do dispositivo que pode servir para aumentar o poder de processamento, pois muitos dispositivos podem ter seu hardware limitado devido à finalidade do dispositivo. Outra forma de conceber a virtualização é percebê-la como um aumento de memória para o dispositivo, e também como uma extensão de comunicação e interação, tanto entre o usuário (IHC), quanto para os outros dispositivos (M2M).

Para que o framework funcione não é necessário que nenhuma alteração seja feita nos dispositivos físicos em nível de protocolo de comunicação. Porém os dispositivos devem ser idealizados e desenvolvidos cientes de que farão parte do framework e por isso devem se comunicar com seus respectivos Drivers (extensão virtual). Para que tal solução funcione os dispositivos devem saber previamente qual nó da rede está diretamente ligado ao framework (dentro do Manna-X Server) e enviar suas mensagens a ele, pois todas as mensagens enviadas do dispositivo físico para esse nó receptor serão encaminhadas para seus respectivos Drivers.

Tal nó receptor não necessita ser o coordenador da rede, no entanto ele deve ser capaz de realizar, além dos encaminhamentos dos pacotes ao framework, tarefas de gerenciamento de rede, e análise da rede, e logo, deve conseguir se comunicar com todos os nós da sua rede.

4.2 ARQUITETURA PROPOSTA

O framework Manna-X como dito, fica contido dentro do Manna-X Server, um computador que está conectado às mais diferentes redes que compõem o ambiente. Como visto, essa versão do framework contempla a centralização e desta forma o Manna-X promove a integração entre os dispositivos e a aplicação feita pelo desenvolvedor. O framework não pode ser confundido com uma IDE de desenvolvimento como Netbeans, Eclipse, Microsoft Visual Studio, e sim como uma biblioteca utilizada pela aplicação do desenvolvedor. O framework quando referenciado e corretamente utilizado, cria o ambiente para uma solução Manna-X.

A Figura 4.3 mostra a arquitetura do framework Manna-X. Como pode ser visto existem duas grandes divisões, a parte de hardware (dispositivos e nós receptores) e a parte de software (biblioteca). Apesar de ser dividido dessa maneira o framework diferentemente

de outras arquiteturas de camadas, se comporta de outra maneira. Por exemplo, soluções em comunicação como o modelo OSI, requer que em ambos os lados da comunicação implementem as camadas definidas, ou para computadores intermediários somente as primeiras camadas. Na arquitetura Manna-X somente o Manna-X Server que implementa a parte de software, ou seja, os dispositivo não necessitam modificar seu código ou seu hardware para se adaptarem ao sistema, o sistema é que se adapta a eles.

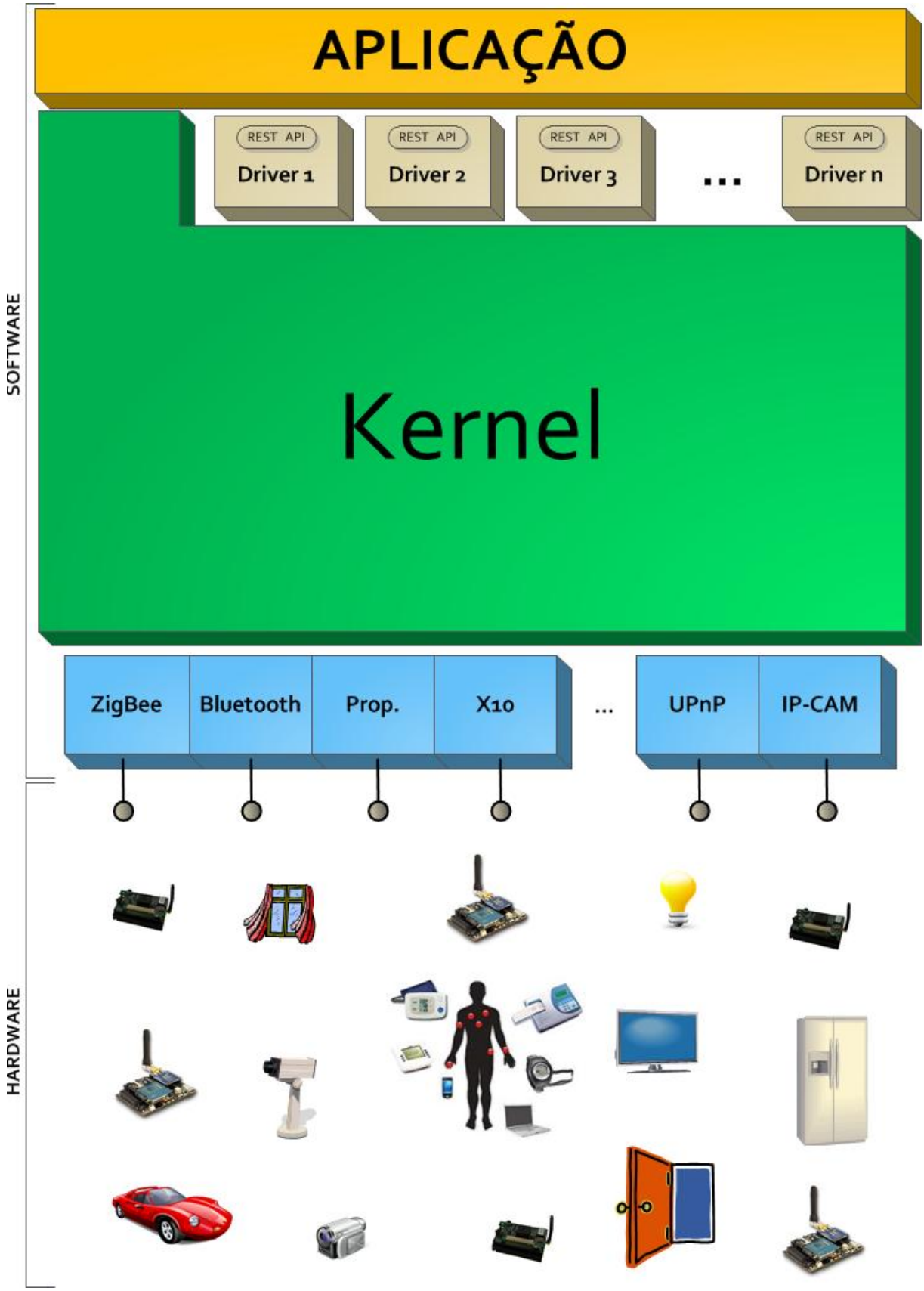

Figura 4.3: Arquitetura do Framework Manna-X

Quando houver a necessidade de comunicação, o Manna-X Server usa os seus meios de comunicação com todos os tipos de redes, logo, o dispositivo já possui nativamente um modo de comunicação e por isso se comunicam.

4.2.1 Redes de Dispositivos

A parte de hardware é composta pelas redes dos dispositivos e pelos dispositivos propriamente ditos. As redes dos dispositivos são as redes formadas pela tecnologia de comunicação que cada dispositivo possui, como por exemplo, ZigBee, Z-Wave, X10, Bluetooth ou até mesmo protocolos proprietários. Essas redes possuem suas peculiaridades, mas em geral todas seguem um mesmo padrão de rede e por isso podem ser generalizadas. Além disso, a parte de hardware é composta pelos meios de comunicação, que podem ser cabos, antenas, módulos, ou qualquer hardware que está conectado diretamente ao Manna-X Server, que gerenciam ou sabem como se comunicar com os dispositivos que utilizam a mesma tecnologia de comunicação.

4.2.2 A Camada de Software

Dentro da camada de software está contido o framework propriamente dito, sua divisão é por camadas e módulos. Ele é considerado um framework vertical, pois é específico para aplicações que gerenciam dispositivos em redes heterogêneas e que por isso são úteis para a solução de ambientes inteligentes. O framework está dividido em quatro grandes partes, no qual uma é considerada Frozenspot (serviços já implementados pelo framework) e três Hotspots (parte flexível que deve ser implementada).

A primeira parte do framework considerada um Hotspot é a camada de Gerentes de Rede. Ela é responsável por fazer a comunicação do framework com as redes do ambiente, essa camada que provê a adaptação com os dispositivos, dentro dela podem existir quantos módulos Gerentes de Rede forem necessários. A segunda parte de baixo para cima da figura é um Frozenspot e é chamada Kernel do Manna-X, ela é o núcleo do framework, a parte que faz toda a integração e gerenciamento, e por isso fornece meios para a criação de um ambiente inteligente, ela é um módulo único no framework.

A terceira parte, considerada um Hotspot é a camada mais acima encontrada na Figura 4.3, chamada Aplicação, essa camada é totalmente desenvolvida pelo desenvolvedor de ambientes inteligentes que utiliza as demais camadas do framework e também é considerada um módulo único. A última camada, também considerada um Hotspot, é a camada dos Drivers dos dispositivos, que está no meio das outras duas camadas já

citadas. Cada módulo é considerado uma virtualização do dispositivo, logo é ela deve ser implementada pelo fabricante do dispositivo.

Os módulos mostrados na parte de software da figura anterior são agentes que trabalham de maneira independente, mas cooperam uns com os outros para a formação do framework. Desta maneira é dito que a arquitetura do framework é orientada a serviços, pois os módulos prestam serviços uns para os outros e de forma assíncrona, sendo assim, pode-se dizer também que é uma arquitetura orientada a eventos.

Muitas são as vantagens da utilização dessas arquiteturas para formar a arquitetura do framework. A justificativa para tal utilização deve-se pela dinamicidade que é um ambiente inteligente. Tais ambientes precisam gerir uma grande quantidade de dispositivos e lidar com sua mobilidade, dinamicidade e extensibilidade, para isso o paradigma de orientação a eventos modela bem esse ambiente onde dispositivos entram e saem da rede, mudam de estados disparando eventos para deixar todo um sistema atualizado (Gámez e Fuentes, 2011), da mesma maneira em uma arquitetura de agentes para manter todos atualizados a orientação a eventos é a melhor escolha.

O paradigma orientado a serviços modela bem o que os agentes têm a oferecer, de maneira concreta, de fácil entendimento, além do que os dispositivos são encarados como provedores de serviços facilitando o manuseio já que todos os módulos são encarados como agentes.

A figura a seguir mostra o framework mais detalhadamente no que se refere à subdivisões dos módulos, mais especificamente do Kernel. As quatro grandes camadas da parte de software do framework serão explicadas nas próximas seções.

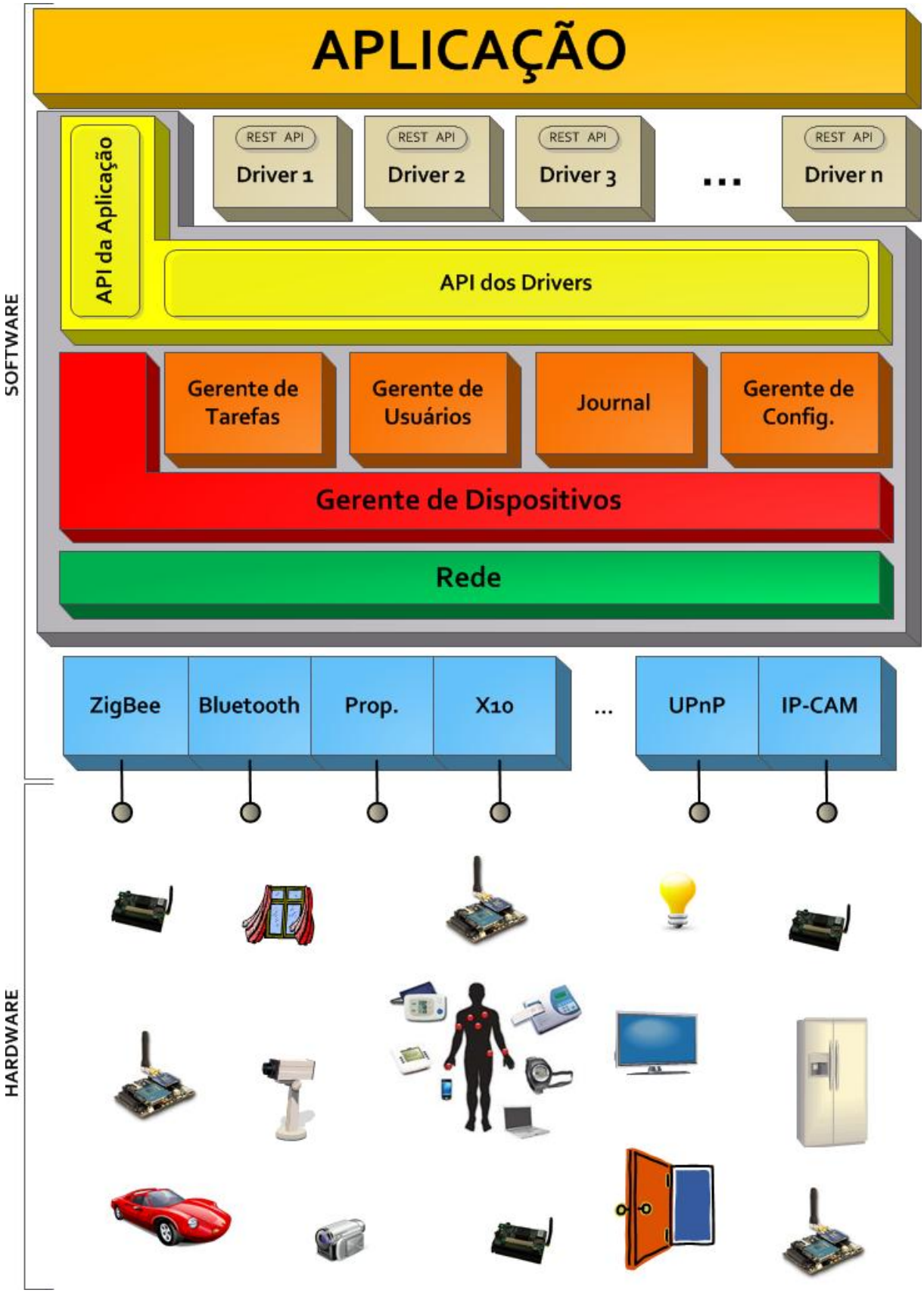

Figura 4.4: Arquitetura detalhada do Framework Manna-X

4.3 GERENTES DE REDE

Essa camada é a primeira parte de software que tem contato com a camada de hardware diretamente. Sua principal função é gerenciar as redes que são específicas para cada tecnologia de comunicação. Como visto na Figura 4.5, essa camada pode possuir mais de um gerente (ou módulo, componente) e cada um está diretamente conectado com o meio externo para se comunicar com os dispositivos de sua respectiva rede.

Cada módulo diferentemente de uma arquitetura em camadas só se comunica com a camada de Rede dentro do Kernel e não pode se comunicar com os outros módulos do sistema. Seu trabalho é fornecer informações para a camada de Rede sobre a rede que gerencia, tais como, dispositivos encontrados, estatísticas da rede, informações sobre a rede e envio de pacotes para os dispositivos. Outro serviço prestado é o disparo de eventos quando um novo nó é encontrado na rede, quando um nó existente deixa a rede, ou quando um pacote chega.

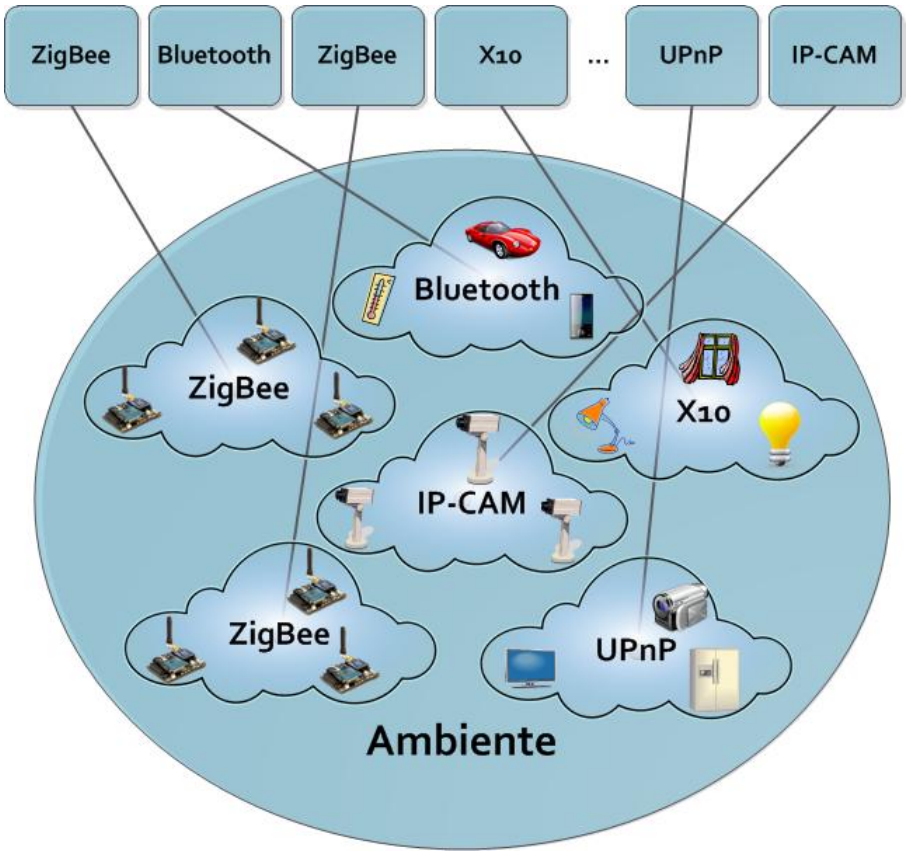

Figura 4.5: Exemplo de Gerentes de Rede no Framework Manna-X

Além da função gerencial, essa camada serve de empacotador e desempacotador de pacotes, cada módulo sabe como tratar pacotes de uma tecnologia de comunicação específica, por exemplo, Bluetooth, ZigBee, UPnP, X10, ou até mesmo protocolos proprietários. Não é necessário que exista uma antena ou hardware por módulo, pois em um mesmo meio duas ou mais tecnologias podem coexistir. Um exemplo disso é a rede doméstica (IEEE 802) formada por roteadores, os dispositivos dessa rede podem estar conectados por Wireless ou por Ethernet e podem possuir diversos protocolos dependendo do fabricante como TCP/IP, mas também podem utilizar outros protocolos de mais alto nível para formar sub-redes de dispositivos específicos como UPnP. Assim, diversas sub-redes, até mesmo sub-redes proprietárias, podem estar sobre uma mesma tecnologia, mas virtualmente serem diferentes, como a das câmeras IP por exemplo.

Outra característica dos módulos dessa camada é que dois módulos da mesma tecnologia de comunicação podem coexistir sem problemas, pois são duas instâncias diferentes do mesmo tipo de rede. Um exemplo é um ambiente que possua duas redes ZigBee. Dois Gerentes ZigBee são instanciados ao mesmo tempo, porém cada um está ligado a um coordenador de rede (hardware) e cada um gerencia sua própria rede ZigBee, podendo ser visto na Figura 4.5.

O Manna-X Server pode não possuir todas as entradas e saídas suficientes para determinadas aplicações, neste caso, adaptadores podem ser criados para se comunicarem via TCP/IP e um Gerente de Rede específico para esta aplicação pode ser criado, resolvendo o problema.

Pelo fato de ser considerado um Hotspot, os Gerentes de Rede podem ser implementados conforme a necessidade de uso, e por isso podem ser criados diversos tipos de rede, mesmo que não sendo de uma rede verdadeira, ou que seja uma sub-rede dentro de alguma rede, propiciando também a integração com redes proprietárias.

Os Gerentes de Rede para se comunicarem com a Camada de Rede devem seguir o mesmo protocolo. O protocolo criado incide diretamente no comportamento do Gerente, logo, eles devem respeitar algumas ações, que são:

- Fornecer a lista de todos os dispositivos;
- Informar quando um dispositivo entra ou sai da rede;
- Permitir o envio de mensagens para os dispositivos;
- Informar quando uma mensagem for recebida de algum dispositivo;
- Informar detalhes da rede;

- Atribuir IDs únicos para cada dispositivo, de maneira que o dispositivo ao entrar na rede sempre receba o mesmo ID.

4.4 KERNEL

O Kernel é a parte implementada pelo framework (Frozenspot), seu objetivo é fornecer uma interface para a camada de aplicação e para os drivers de dispositivos, bem como também realizar a comunicação com os dispositivos de forma ubíqua para o desenvolvedor. Dentre suas funcionalidades estão o gerenciamento das redes de dispositivos, prover para as outras camadas a visão de transparência de dispositivos, ou seja, as camadas que a utilizam não necessitam saber como se comunicar com os dispositivos, para elas todos os dispositivos estão em uma mesma rede.

Além disso, o Kernel também é responsável por executar tarefas requeridas pela Aplicação, gerenciar os usuários do sistema, fazer a auditoria das tarefas executadas e atualizar o sistema de uma maneira autonômica.

O Kernel está divido de forma vertical, onde módulos em um mesmo nível horizontal podem ser acessados por eles mesmos ou pelo nível superior e inferior, mas verticalmente os módulos só podem acessar o nível horizontal de cima ou a de baixo, conforme visto na Figura 4.4. De baixo para cima a primeira camada (nível) é chamada de camada de Rede, logo após ela, o Gerente de Dispositivos e em outro nível acima, estão os Gerentes do Kernel que é composto pelo o Gerente de Tarefa, Gerente de Usuários, Journal e Gerente de Configuração. Por fim, no último nível do Kernel, fazendo a interface com os Drivers de Dispositivos e com a camada de Aplicação, a API que pode ser utilizada por eles. As partes internas do Kernel serão explicadas nas próximas subseções.

4.4.1 Camada de Rede

A camada de Rede é a primeira camada do Kernel. Sua função é fazer o intermédio entre os Gerentes de Redes e os demais módulos do Kernel. Seu principal objetivo é criar a ideia de transparência dos dispositivos, tal ideia significa dizer que aos olhos dos módulos acima da camada de Rede todos os dispositivos são iguais, ou seja, os módulos não sabem qual a tecnologia de comunicação, para eles todos são iguais.

Tal visão facilita a implementação, utilização e o gerenciamento dos dispositivos devido à generalização criada. A generalização criada faz com que os dispositivos ganhem IDs da camada de Rede para que sejam acessados e referenciados pelos demais módulos. Essa

generalização cria uma grande rede virtual que agrupa todos os dispositivos de todas as redes gerenciadas pelos Gerentes de Rede.

Embora a camada de Rede só possa ser acessada pelo Gerente de Dispositivos as funções exercidas por essa camada são a de fornecer ações e informações sobre a rede de dispositivos gerenciados. As ações que a camada de Rede deve prestar são:

- Fornecer a lista de todos os dispositivos;
- Redirecionar as mensagens vindas do Gerente de Dispositivos para o Gerente de Rede correto;
- Disparar eventos de rede como: novos nós encontrados, nós que deixaram a rede, pacotes recebidos e erros;
- Criar IDs únicos para cada dispositivo encontrado nas sub-redes.

4.4.2 Gerente de Dispositivos

O Gerente de Dispositivo é o módulo agente que cuida dos Drivers de Dispositivos. Mais do que isso, ele que faz a ligação entre o nó encontrado pela camada de Rede com o Driver de Dispositivo correto. Sua responsabilidade é de fornecer meios ou encontrar automaticamente o Driver correto para cada dispositivo e instanciá-lo.

Como visto na Figura 4.4 essa camada fica acima da camada de Rede e por isso ela tem suas funcionalidades estendidas para expor algumas funcionalidades da camada de baixo como:

- Fornecer a lista dos dispositivos já com a visão de Drivers;
- Redirecionar as mensagens vindas dos demais módulos do Kernel para a camada de Rede;
- Permitir a escolha do Driver para o dispositivo;
- Redirecionar mensagens vindas da Camada de Rede para o Driver correto.

4.4.3 Gerente de Tarefas

Uma das funcionalidades do framework Manna-X é a possibilidade de serem registradas tarefas para serem executadas ao longo do tempo. Essas tarefas são ações rotineiras que o usuário faz no ambiente, mas podem ser automatizadas. Na busca em ajudar o desenvolvedor de ambientes à criação desse mecanismo, no framework Manna-X foi proposto

dentro no Kernel o Gerente de Tarefas. Sua função é executar tarefas programadas pelo usuário por meio da camada de Aplicação, a fim de ajudar o usuário do ambiente em tarefas rotineiras. Como entrada o Gerente de Tarefas recebe uma tarefa que contêm comandos básicos para abrir portas, acender luzes, dentre outros comandos, por exemplo.

Esse gerente não é considerado alguma inteligência ambiental, e sim um executor de tarefas já programadas. Porém a inteligência ambiental pode ser implementada em nível de camada de Aplicação e essa programar tarefas para serem executadas pelo Gerente de Tarefas.

As tarefas feitas pelo usuário podem conter literais, cláusulas, loops e referências a serviços prestados pelos dispositivos, pois os Drivers de Dispositivos exportam suas APIs que são serviços utilizados como sensores e atuadores.

O gerente de tarefas não é um loop que fica verificando todas as condições das tarefas programadas, seu acionamento é feito somente quando algum evento é disparado por algum módulo, seja um módulo do Kernel ou um Driver de Dispositivo. Seu funcionamento pode ser mais bem visto na Figura 4.6.

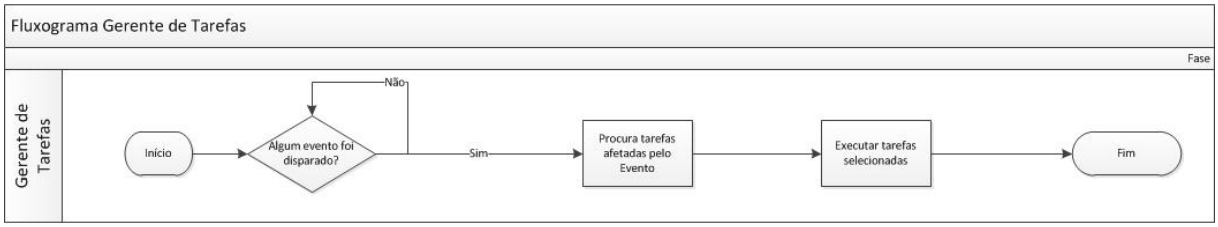

Figura 4.6: Funcionamento do Gerente de Tarefas

As tarefas programadas possuem a seguinte forma:

$$ON < evento > DO < \text{iterações} > \tag{4.1}$$

Tal forma significa dizer que, no disparo de algum evento, faça as iterações programadas, que podem conter loops, cláusulas if, a chamada de serviços da API de qualquer Driver de Dispositivo ou módulo, e suas combinações.

4.4.4 Gerente de Usuários

O módulo Gerente de Usuários tem por objetivo gerenciar os usuários que utilizam os dispositivos do ambiente, mais do que isso, esse módulo é responsável por controlar o acesso e permitir que dados sobre os usuários fossem relacionados, como sua posição dentro do ambiente, dispositivos que pertencem a ele, seus gostos, dentre outras características.

Tais informações podem ser utilizadas pelo Gerente de Tarefas, podendo assim criar tarefas mais complexas levando em consideração os usuários do ambiente.

4.4.5 Journal

A fim de criar uma auditoria das ações ocorridas dentro do ambiente, o módulo chamado Journal tem por objetivo registrar todas as ações feitas pelos usuários ou pelo Gerente de Tarefas a fim de guardar um histórico do ambiente.

4.4.6 Gerente de Configuração

O framework Manna-X foi projetado para tratar da dinamicidade, extensibilidade, heterogeneidade dos dispositivos, e por isso foi projetado em módulos agentes que utilizam o paradigma orientado a eventos e serviços. Os módulos prestam serviços uns para os outros de maneira assíncrona e por isso disparam eventos quando mudam de estados.

A Figura 4.3 mostra todos os módulos do sistema, desde a camada de Aplicação, os Drivers, Gerentes de Rede e o Kernel, todos são módulos recarregáveis do framework, ou seja, em tempo de execução caso exista alguma atualização, estes módulos podem ser trocados, ou se necessário, removidos e adicionados conforme ocorra a evolução do framework. Para integrá-los e gerenciá-los que o Gerente de Configuração foi idealizado.

A visão do Gerente de Configuração é criar um sistema autonômico, ele é responsável por manter atualizados todos os módulos do Kernel e dos Drivers de Dispositivos ou até adicionar novos módulos em um futuro. Para realizar tal atividade um servidor deve ser hospedado pela Internet e ele será o responsável por possuir todas as últimas versões de todos os módulos do Kernel, bem como possuir todos os Drivers de todos os dispositivos.

Tal servidor chama-se, Manna Central Server (MSC) e poderá atender a quantos Manna-X Servers forem necessários, seu trabalho será o de um repositório de módulos que será usado pelo Gerente de Configuração de cada Manna-X Server para mantê-los atualizados. Além disso, o MSC tem a possibilidade de prover novas versões do framework Manna-X tornando assim o sistema autonômico e atualizado. É importante ressaltar que cada Manna-X Server possui somente uma única instância do framework.

4.4.7 API de Interface

A API de Interface tem a responsabilidade de expor as funcionalidades do Kernel para os módulos fora dele. Como visto na Figura 4.3 a API faz fronteira com dois módulos: a Aplicação e os Drivers de Dispositivos. Porém cada módulo possui suas peculiaridades e

pode necessitar de algumas funções que o outro não necessitaria saber, bem como funções que são permitidas para um e não para o outro.

Com o intuito de fornecer uma interface para os Drivers de Dispositivos e outra para a camada de Aplicação, a API de Interface possui duas interfaces criadas para solucionar tal problema. A primeira interface foi idealizada para a utilização da Aplicação, com ela é possível:

- Descobrir os Drivers de Dispositivos e assim obter informações e utilizar seus serviços;
- Cadastrar, remover e modificar tarefas;
- Cadastrar, remover e modificar usuários do ambiente;
- Verificar auditorias das ações passadas;
- Cadastrar e remover Gerentes de Rede.

Já a segunda API foi desenvolvida para os Drivers de Dispositivos, ela não contêm todas as funcionalidades da outra API, pois ela é a interface que os dispositivos usarão e logo, não podem ter acesso a todas as informações da outra interface. Porém, ela possui outras funcionalidades como:

- Descobrir os outros Drivers de Dispositivos e assim obter informações e utilizar seus serviços;
- Enviar pacotes para seu respectivo dispositivo real;
- Persistir dados.

4.5 DRIVERS DE DISPOSITIVOS

Os dispositivos encontrados nas sub-redes são diversos e muitas vezes heterogêneos. Isto implica que cada dispositivo tenha suas peculiaridades como pouco poder de processamento, pouca memória, economia de energia por serem sistemas embarcados e móveis, protocolo de comunicação específico, dentre outras características.

Nem todos os dispositivos embarcados são limitados, mas não seria uma má ideia aumentar seu poder de processamento ou de memória, bem como poder integrá-los na Internet das Coisas. Visando solucionar esses problemas que os Drivers de Dispositivos foram idealizados.

O Driver de Dispositivo é uma extensão virtual dos dispositivos reais no que se refere a um aumento de funcionalidades, poder de processamento e aumento de memória. Os Drivers visam expandir a experiência do usuário com os dispositivos, melhorando a Interface Humano-Computador e criando novas experiências. Mais do que isso, ele visa melhorar a comunicação entre dispositivos (M2M) de modo a facilitar a troca de pacotes e utilização de serviços.

Quem implementa o Driver de um dispositivo específico é o próprio fabricante, pois ninguém melhor que eles sabe qual a melhor interface para o usuário e como se comunicar com o dispositivo. A comunicação com o dispositivo é feita por meio dos Drivers, que expõem uma API de serviços para que os outros módulos possam utilizar e um conjunto de páginas Web para fazer a interface com o usuário. Suponha uma lâmpada que possua seu Driver no framework Manna-X, quando o usuário abre a página da lâmpada ele poderá ver uma imagem com a foto de uma lâmpada acesa ou apagada para saber o estado dela e dois botões para acendê-la e apagá-la. Quando o usuário pressiona o botão para acender a lâmpada o driver já sabe qual o comando que será enviado para a lâmpada, essa mensagem é enviada para o Kernel, que despacha a mensagem para o Gerente de Rede correto. Esse gerente empacota a mensagem com o protocolo correto, por exemplo, ZigBee, e a envia para o dispositivo real correto. Quando a mensagem chega ao dispositivo, o dispositivo sabe que deve acender a luz, pois o Driver foi feito pelo fabricante e por isso enviou o comando correto para acender a luz.

O comando enviado para o dispositivo fica na parte de dados do pacote do protocolo de comunicação específico. Quem faz o pacote é o Gerente de Rede que sabe como formar o cabeçalho e demais partes do pacote dependendo do protocolo. O Driver pode se comunicar com o dispositivo real a qualquer momento, independentemente da interação do usuário ou dos módulos pela API, além disso, somente o Driver correto que pode receber as mensagens vindas do seu dispositivo real.

A partir do momento que o dispositivo real recebe a mensagem e muda ou não seu estado, ele pode enviar uma mensagem confirmando o comando e descrevendo seu novo estado para o Driver. Essa mensagem é recebida pelo Gerente de Rede, que passa para Kernel que a despacha para o Driver correspondente. O Driver pode então mudar a imagem da lâmpada da página Web para informar o novo estado para o usuário e, além disso, o Driver deve disparar um evento para todos os módulos interessados informando sua alteração de estado.

As mensagens trocadas pelo dispositivo e seu respectivo Driver não segue um protocolo, cada fabricante é livre para criar o seu, ou seja, o conteúdo das mensagens só depende do

fabricante, a única parte que é fixa é o pacote criado pelo Gerente de Rede e o protocolo da rede utilizada.

O disparo de eventos é essencial na arquitetura Manna-X, sem o disparo de eventos não é possível saber se um dispositivo alterou seu estado, por isso é um dever do Driver sempre disparar eventos de mudança de estado. O estado do dispositivo representa suas variáveis e por isso o Driver deve disparar eventos mostrando quais variáveis foram alteradas.

A partir da comunicação entre o Driver e seu respectivo dispositivo, o fabricante pode aproveitar disso para armazenar dados no servidor, aumentar o poder de processamento como, por exemplo, uma câmera que faz o reconhecimento facial, a câmera pode não possuir um bom processador para fazer tal atividade, mas ela pode enviar a imagem para o Driver para que ocorra tal processamento no servidor. Por causa do Driver o fabricante pode aumentar a experiência do usuário criando uma boa página Web, que possua mais funcionalidades, como por exemplo, uma geladeira que informa que alimento está faltando ou acabando, qual a temperatura das divisões internas, quanto está gastando de energia, ou até mesmo pode possuir uma câmera interna para o usuário olhar os alimentos sem abrir a porta evitando a perca de temperatura.

O framework não visa servir de *gateway* para a comunicação direta entre os dispositivos, mesmo que servisse seria difícil que os dispositivos conseguissem descobrir como se comunicar com outro dispositivo, mesmo porque algumas redes não suportam descoberta de serviços nem a passagem de dados grandes, como fotos e vídeos, podendo dificultar a comunicação. Mas para permitir a comunicação dos dispositivos o framework propõe que tal solução seja feita entre os Drivers de Dispositivos, ou seja, por meio deles qualquer dispositivo pode se comunicar com o outro e descobrir serviços.

Os Drivers podem, através da API fornecida pelo Kernel, descobrir os dispositivos na rede e seus serviços e assim realizar uma comunicação entre módulos utilizando tais serviços aumentando a experiência do usuário com o ambiente através da interação dos dispositivos. Tal ideia resolve e facilita a passagem dos dados, pois o tamanho não é mais um problema já que são processos sendo executados no servidor, e a descoberta de serviços, facilitada pela API.

Para integrar os dispositivos na Internet das Coisas, os Drivers devem além de implementar as páginas Web para o dispositivo, criar uma API REST para facilitar a comunicação entre máquinas, mais o que isso, possibilitar que Mashups possam ser criadas a partir das APIs.

Em um ambiente mais de um dispositivo pode ser igual, e por causa disso a mesma classe Driver será instanciada mais de uma vez pelo Gerente de Dispositivos. O que ocorre

é que cada Driver coordena seu respectivo dispositivo real, sem nenhuma complicação e conflito, pois o Gerente de Dispositivo o relaciona com o ID correto da Camada de Rede.

O paradigma do Driver é gerenciar um respectivo dispositivo real, porém esse paradigma não é uma necessidade. A fim de criar novas experiências para os ambientes inteligentes os Driver podem também controlar dispositivos virtuais. Tais dispositivos não existem realmente no ambiente, os Drivers são programas que simulam a existência de um dispositivo real e podem ter diversas utilidades.

Um exemplo de utilidade é o Driver que simula um dispositivo que adquire informações intangíveis sobre o usuário a fim de fornecer mais informações para o ambiente se adaptar ao humano. Tais informações podem ser adquiridas por meio das redes sociais, ou seja, o usuário por exemplo, escreve que está triste na sua rede social, a partir desse momento o Driver interpreta tal informação e dispara eventos sobre o usuário, tornando o ambiente ciente de contexto.

4.6 CAMADA DE APLICAÇÃO

A camada de Aplicação é a parte implementada pelo usuário no framework, é ela que faz o uso do Kernel e dos Drivers de Dispositivo e por isso pode utilizar as informações obtidas do jeito que for necessário. A função do framework é ajudar no gerenciamento de ambientes inteligentes, mas do que isso, gerenciar os dispositivos e as redes do ambiente.

O framework tem por objetivo ser genérico para qualquer ambiente ou solução e por isso seu paradigma é ajudar no gerenciamento dos dispositivos e das redes e não na criação de uma inteligência ambiental.

A camada de Aplicação é um módulo que utiliza serviços do Kernel e dos Drivers através de uma API fornecida pelo framework. A partir da API, a camada de Aplicação pode implementar qualquer tipo de ambiente ou solução utilizando seus dados.

4.7 INTERFACES DO FRAMEWORK

Como visto na Figura 4.1, existem três atores no sistema, o Desenvolvedor de Ambientes, o Fabricante de Dispositivos e o Usuário Final, porém somente dois fazem uso direto do framework e utilizam suas interfaces. Pode-se separar as funcionalidades utilizadas por cada ator. A Figura 4.7 mostra as funcionalidades do Desenvolvedor de Ambientes. O desenvolvedor tem a possibilidade de descobrir os dispositivos conectados, descobrir os serviços prestados por eles, utilizar esses serviços, criar/alterar e remover tarefas, adicionar

e remover Gerentes de Rede, criar/alterar e remover usuários e ler o as auditorias do Journal.

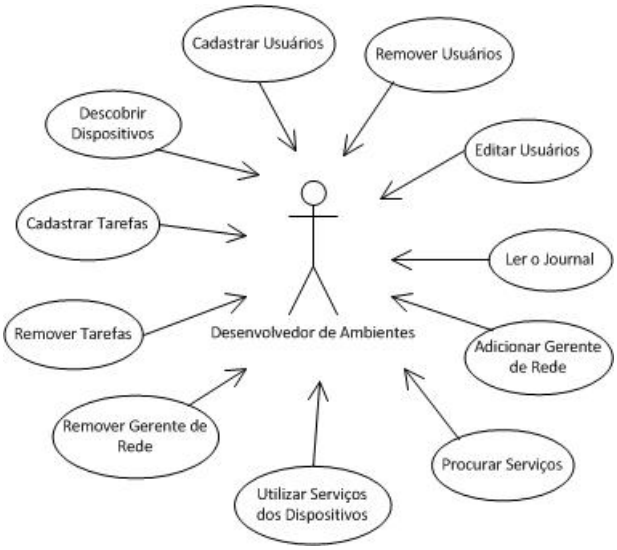

Figura 4.7: Funcionalidades para Desenvolvedor de Ambientes

Do outro lado do framework, os fabricantes de dispositivos utilizam algumas interfaces para interagir com o framework, tais funcionalidades utilizadas por eles podem ser vistas na Figura 4.8. Os fabricantes tem a possibilidade de armazenar dados no framework, realizar algum processamento dentro do Driver, criar uma API REST para a Web das Coisas, Descobrir e utilizar serviços de outros dispositivos, aumentar a experiência do usuário criando páginas Web e novas interações com o usuário.

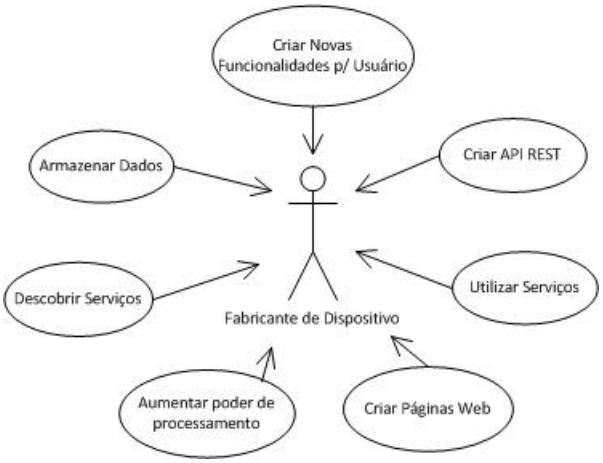

Figura 4.8: Funcionalidades para o Fabricantes de Dispositivos

Os quatro módulos do framework trabalham com sinergia para criarem o ambiente idealizado. Para que isso ocorra cada um deles possui interfaces para com os outros e essas interfaces podem ser melhor vistas na Figura 4.9.

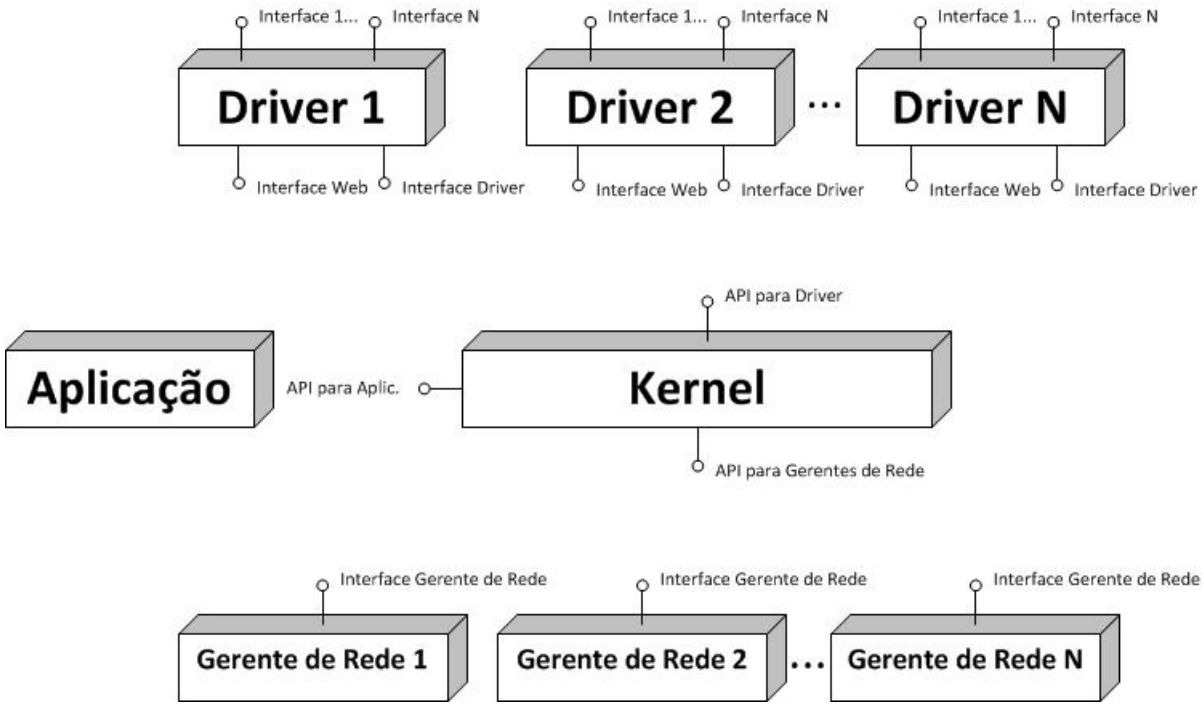

Figura 4.9: Interfaces dos módulos do framework

A visão mostrada na figura representa todos os módulos do framework, podendo ser notado que, somente deverá existir um único módulo de Kernel e uma única de Aplicação. O número de Gerentes de Rede e Drivers de Dispositivos pode ser variado, dependendo da solução. O número de gerentes significa o número de redes que o ambiente possui, e o número de Driver significa quantos dispositivos o ambiente possui. Esse número pode ser diferente do número de dispositivos reais, pois alguns Drivers podem representar dispositivos virtuais e logo não existem realmente.

Cada módulo do framework possui suas interfaces, ou seja, serviços que os outros módulos podem utilizar. O Gerente de Rede só se comunica com a camada de Rede contida dentro do Kernel, logo, sua interface, só é conhecida pelo Kernel. O Kernel possui diversas interfaces para que os outros módulos possam utilizar seus serviços, essas interfaces são chamadas de API e só podem ser utilizadas pela classe de módulos específica.

A API para Gerentes de Rede só pode ser utilizada pelos Gerentes de Rede se comunicarem com o Kernel, esses serviços foram descritos nas seções anteriores, mas em suma tratam o recebimento de mensagens, a descoberta de dispositivos e outros serviços

que uma rede fornece. Essa API está contida no bundle do Kernel, com ela os Gerentes de Rede podem acessar o Kernel. Ela é descrita da seguinte forma:

```
1  public interface ManagersAPI
2  {
3          public void NovoNo(NoSensor no, long id);
4
5          public void NoSaiu(NoSensor no, long id);
6
7          public void MensagemRecebida(PacoteMannaX pacote, long id);
8
9          public void NoEntrou(NoSensor no, long id);
10
11 }
```

Para o framework acessar os Gerentes de Rede, eles devem seguir uma interface genérica chamada de NetManager. A interface é um bundle que deve ser instalado junto com o Kernel Manna-X na plataforma OSGi. Por isso o Gerente de Rede, que pode ser implementado pelo desenvolvedor de ambientes ou adquirido de terceiros, deve implementar e publicar a interface por meio do OSGi. Sua descrição está a seguir:

```
1  public interface NetManager
2  {
3          public static final String MANNAX_BUNDLE_TYPE =
4          "MannaXBundle-Type";
5
6          public static final String NM_BUNDLE_TYPE = "NetworkManager";
7
8          public void EscanearRede();
9
10         public int EnviarMensagem(Integer id,int [] mensagem);
11
12         public Boolean EnviarBroadcast(int [] mensagem);
13
14         public List<NoSensor> getNos();
15
16         public void RemoverNo(Integer id);
17 }
```

A API para a Aplicação é destinada somente para a camada de Aplicação, que é feita pelo desenvolvedor de ambientes, ela contêm informações úteis para que o desenvolvedor

crie o ambiente inteligente como, lista de dispositivos, criação de tarefas, criação de novo Gerente de Rede, leitura do Journal, dentre outras funcionalidades. Do mesmo modo que os Gerentes de Rede, ela está incluída dentro do bundle do Kernel Manna-X e através dela a Aplicação pode acessar o Kernel. Sua descrição segue abaixo:

```
1  public interface AppAPI
2  {
3          public List<Dispositivo> getDispositivos();
4
5          public List<Dispositivo> getDispositivosSemDriver();
6
7          public Boolean InstallDriver(Dispositivo disp, String driver);
8
9          public Boolean cadastrarTarefa(Tarefa t);
10
11         public Boolean removerTarefa(Tarefa t);
12
13         public List<Tarefa> getTarefas();
14
15         public Boolean cadastrarUsuario(Usuario u);
16
17         public Boolean removerUsuario(Usuario u);
18
19         public List<Usuario> getUsuarios();
20
21         public Boolean installGerenteRede(String path,String porta);
22
23         public Journal getJournal();
24 }
```

Para que o Kernel acesse a Aplicação o processo é o mesmo que para o Gerente de Rede. Um bundle deve ser instanciado contendo a interface que a Aplicação deve implementar e publicar por meio do OSGi. A interface "e descrita a seguir:

```
1  public interface Aplicacao
2  {
3          public static final String MANNAX_BUNDLE_TYPE =
4          "MannaXBundle-Type";
5
6          public static final String APP_BUNDLE_TYPE = "Aplication";
```

```
7
8           public String DispositivoEncontrado(Dispositivo disp);
9
10          public void DriverNotInstalled(Dispositivo disp);
11
12          public void DriverStopped(Dispositivo disp);
13
14          public void DriverStarted(Dispositivo disp);
15
16          public void DispositivoEntrou(Dispositivo disp);
17
18          public void DispositivoSaiu(Dispositivo disp);
19
20  }
```

A última interface exclusiva do Kernel é a API para Drivers, ela é exclusiva para os Drivers de Dispositivos utilizarem, pois serve para que eles mandem mensagens para seus respectivos dispositivos reais, disparem eventos e encontrem outros Drivers de Dispositivos. Para persistirem os dados eles devem utilizar as funcionalidades do OSGi. A API para os Drivers fica dentro do bundle do Kernel e é descrita a seguir:

```
1  public interface DriverAPI
2  {
3           public int EnviarPacote(Long idDriver, int[] mensagem);
4
5           public List<ServiceReference> getDispositivos();
6
7           public void DispararEvento(Long idDriver,Evento evento);
8  }
```

Os Drivers de Dispositivos possuem várias interfaces públicas, como visto na Figura 4.9, mas somente uma exclusiva para o Kernel do framework utilizar. Essa interface serve para que o Kernel avise o Driver sobre eventos ocorridos e mensagens chegas de seu dispositivo real. Essa interface exclusiva para o Kernel é um bundle OSGi que deve ser instalado junto com o bundle do Kernel, e é descrita a seguir:

```
1  public interface Driver
2  {
3           public static final String MANNAX_BUNDLE_TYPE =
4           "MannaXBundle-Type";
```

```
5
6           public static final String DRIVER_BUNDLE_TYPE = "Driver";
7
8           public void PacoteRecebido(int[] mensagem);
9
10          public List<Evento> getEventos();
11
12          public Boolean ModoOff();
13
14          public void ModoOn();
15
16          public HttpServlet getHttpServlet();
17
18          public HttpServlet getApiServlet();
19  }
```

As outras interfaces dos Drivers são públicas, ou seja, qualquer outro módulo pode utilizar. Essas interfaces são interfaces em comuns entre os módulos, ou seja, são interfaces que descrevem serviços para que outros Drivers, ou a camada de Aplicação possam utilizar. Elas podem ou não ser bundles separados, porém, deve saber que, como qualquer interface em OSGi, ela deve ser publicada por alguém que prestará o serviço.

Os Drivers tem a possibilidade de implementar qualquer interface pública, além de criar sua própria interface. A utilidade de criar interfaces já conhecidas é, por exemplo, a procura por algum serviço específico. Suponha que um Driver de um microfone quer realizar a comunicação direta com algum dispositivo que reproduza o som no ambiente. O Driver do microfone pode então procurar por todos os dispositivos que implementam a interface de caixa de som, e pedir para que o usuário selecione a saída (dispositivo) que desejar. Desse modo, o fabricante de dispositivo já conhece as funções/serviços previamente em tempo de construção, e logo, pode utilizá-las em seu Driver.

A interface pública específica do Driver de Dispositivo não serve para a procura em tempo de construção, a menos que já seja conhecida previamente, porém serve para a chamada genérica de serviços pelo Gerente de Tarefas, e também para a utilização genérica da camada de Aplicação.

O Driver de Dispositivo deve além dessas interfaces explicadas, implementar obrigatoriamente a interface web. Essa interface é pública e sua função é fornecer meios de comunicação para humanos e máquinas fora do framework. Essa interface fornece páginas Web para que a camada de Aplicação possa utilizá-las como interface gráfica e

uma API REST para que dispositivos que não pertençam ao framework possa utilizar seus serviços. Essa API REST pode ser utilizada pelos módulos do framework, porém ela não se faz necessária, pois as mesmas funcionalidades encontradas na API REST podem ser encontradas pelas interfaces publicadas internamente do framework.

A camada de Aplicação que controla o acesso do usuário do ambiente com o dispositivo, desse modo, ele pode ou não utilizar as páginas web, e permitir ou não a exposição de serviços REST para a Internet.

4.8 FUNCIONAMENTO DO FRAMEWORK

O funcionamento do framework tem como base todas as características já citadas para realizar o trabalho de gerenciar as redes de dispositivos. Tais características são: a resolução da interoperabilidade entre os dispositivos, mobilidade, extensibilidade, flexibilidade, modularização, serviço de descoberta de rede, auto-organização, ciência de contexto, interface intuitiva, proteger a rede de pessoas e dispositivos mau intencionados, permitir ou ter alguma inteligência ambiental, conseguir tratar a dinamicidade dos dispositivos, ser ubíquo; não ter um alto custo de implantação e tratar a heterogeneidade dos dispositivos.

Por isso, para tratar tais características o funcionamento do framework segue alguns conceitos da arquitetura orientada a serviço (SOA) e também da arquitetura orientada a eventos (EOA). O framework em si está executando a todo o momento, isso quer dizer que, processamentos ocorridos dentro de cada módulo serão escalonados normalmente pelo sistema operacional, porém em certos pontos o framework se comporta como uma arquitetura EOA E SOA.

A parte orientada a serviço diz respeito aos serviços prestados pelos módulos do framework uns para os outros, mesmo dentro da Kernel, suas partes também trabalham desta maneira. Quando um módulo deseja utilizar o serviço do outro simplesmente ele chama um método do módulo e utiliza o serviço. Já a orientação a eventos é utilizada em conjunto com os serviços. Não só os Drivers de Dispositivos podem disparar eventos, mas qualquer módulo ou parte interna do Kernel. Esses eventos servem para avisar os módulos que mudanças estão acontecendo no framework, e caso forem avisados executem sua regra de negócio.

Em especial o Gerente de Tarefas é o melhor beneficiado desse funcionamento. Graças ao disparo dos eventos o Gerente de Tarefas pode executar tarefas programadas e, além disso, indexar as tarefas pelos eventos fazendo com que a procura seja mais rápida. Outra vantagem é a otimização de processamento, caso os eventos não existissem o Gerente de

Tarefas deveria percorrer todos os módulos verificando se alguma coisa foi alterada no módulo, gerando desencontros e muito processamento desperdiçado.

A seguir serão explicados quatro funcionamentos básicos do framework Manna-X, a fim de explicar seu funcionamento, que são: quando o Driver de Dispositivos recebe um comando, quando o Driver de Dispositivos recebe uma mensagem de seu dispositivo real, e quando um dispositivo sai e é encontrado pelo framework.

Quando o framework começa nenhum Gerente de Rede foi instanciado logo nenhum dispositivo pode ser encontrado. Somente o que existe é o Kernel e a Aplicação do desenvolvedor de ambientes inteligentes. Logo que a aplicação cadastrar um Gerente de Rede e conectar seu adaptador no Manna-X Server os dispositivos poderão ser encontrados. A Figura 4.10 mostra o fluxograma sobre o que ocorre quando um dispositivo é encontrado por algum Gerente de Rede.

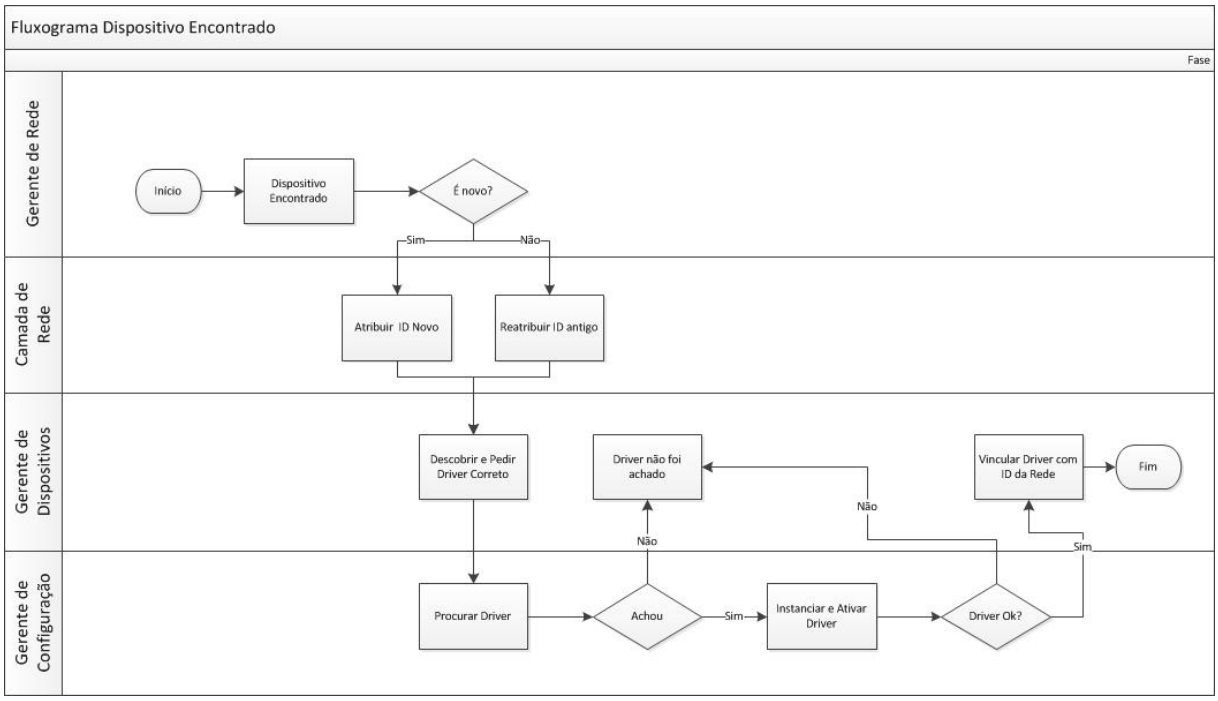

Figura 4.10: Fluxograma Dispositivo Encontrado

Assim que um dispositivo é encontrado é dever do Gerente de Rede identificar o dispositivo de sua rede e atribuir a ele um identificador único, assim se o dispositivo sair e entrar na rede o framework poderá saber se o dispositivo é novo ou não. Após esse passo, o Gerente de Rede através da API do Kernel se comunica com a Camada de Rede avisando do dispositivo encontrado, esse atribui um ID da rede virtual para o dispositivo e avisa o Gerente de Dispositivo. Assim que recebe o aviso de dispositivo encontrado o Gerente de Dispositivo pede para o Gerente de Configuração uma instância de Driver

de Dispositivo correta. Caso o Driver não esteja instalado no framework o Gerente de Configuração busca no Manna Central Server e faz o download do Driver. Caso não exista Driver, o Gerente de Dispositivo é avisado que não existe Driver para tal dispositivo e encerra a atividade. Já se o Driver for achado, ele é instalado e instanciado pelo Gerente de Configuração, ele então avisa o Gerente de Dispositivo que o Driver já está executando, então o Gerente de Dispositivo vincula o Driver com o ID criado pela camada de Rede e avisa a Aplicação.

Quando um dispositivo sai da rede o processo é mais curto, e pode ser visto na Figura 4.11. O Gerente de Rede ao identificar a saída de um dispositivo de sua rede, avisa a Camada de Rede por meio da API. A partir desse ponto, a Camada de Rede remove o nó da rede virtual, mas mantém dados do dispositivo caso volte outrora. Então, ele avisa o Gerente de Dispositivos que também altera seus dados sobre o dispositivo. Logo após, o Gerente de Dispositivos avisa o Driver que seu dispositivo real saiu da rede, e o questiona se continuará operando mesmo sem o dispositivo. Caso não opere o Gerente de Configuração desativa o Driver. Por fim, a Aplicação é avisada sobre a saída do dispositivo real.

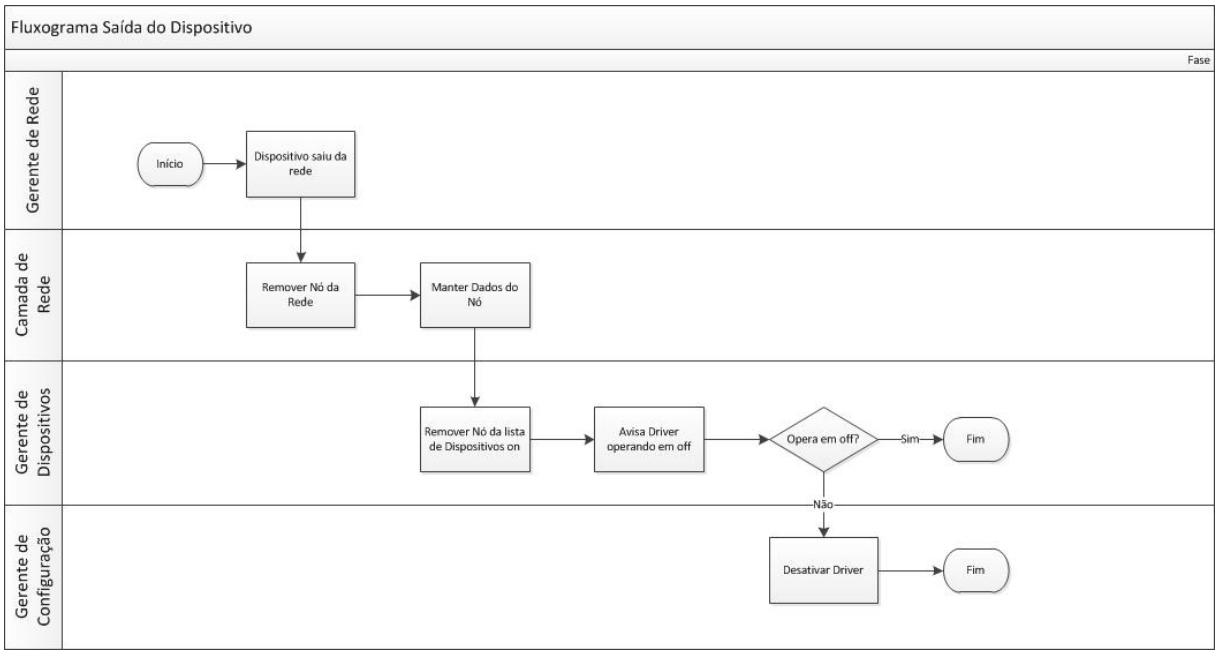

Figura 4.11: Fluxograma Saída de Dispositivo

Outro funcionamento a ser observado refere-se a comandos recebidos pelos Drivers de Dispositivos. Esse comando ocorre quando algum módulo chama serviços prestados pelo Driver e pode ser mais bem visto pela Figura 4.12. Ao receber algum comando o Driver realiza sua regra de negócio que pode necessitar ou não do envio de mensagem para seu

dispositivo real. Caso seja necessário o Driver de Dispositivo envia a mensagem para o Kernel por meio da API de comunicação, a mensagem então é enviada para o Gerente de Dispositivos que envia a mensagem para a Camada de Rede informando o ID da rede virtual. A Camada de Rede identifica o Gerente de Rede correto e o identificador da sub-rede e o despacha para o Gerente de Rede correto.

O Gerente de Rede empacota a mensagem no protocolo da sub-rede e tenta enviar a mensagem para o dispositivo real por três vezes. Após as tentativas o Gerente de Rede retorna uma mensagem informando o sucesso ou a falha na entrega de pacote. Essa mensagem é encaminhada até o Driver correspondente para que tome as atitudes necessárias. É importante lembrar que sempre que ocorra uma mudança nos dispositivos é necessário que o Driver dispare um evento para avisar todo framework.

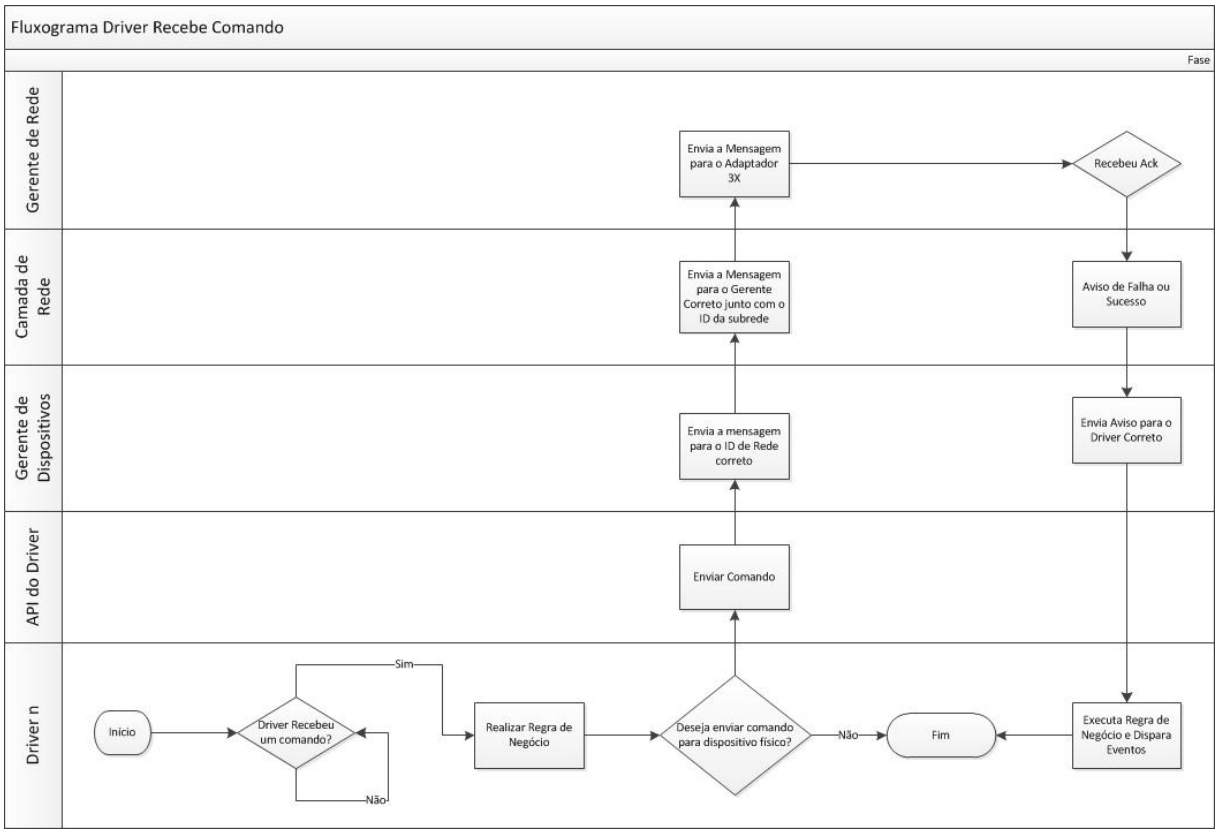

Figura 4.12: Fluxograma Comando Recebido

Por fim, o último funcionamento apresentado é a recepção de mensagens provindas dos dispositivos reais. A Figura 4.13 exemplifica o processo a ser descrito. Quando o dispositivo real envia uma mensagem para o framework ele primeiramente é recebido pelo Gerente de Rede, pois é ele que possui o contato direto com o adaptador da sub-rede. Após a recepção, o Gerente de Rede desempacota o pacote e despacha a mensagem para

a Camada de Rede sempre por meio da API de comunicação. A Camada de Rede repassa a mensagem para o Gerente de Dispositivos que encontra o Driver correto e por meio da API avisa o Driver de Dispositivo que uma mensagem chegou. A partir desse momento, o Driver realiza sua regra de negócio.

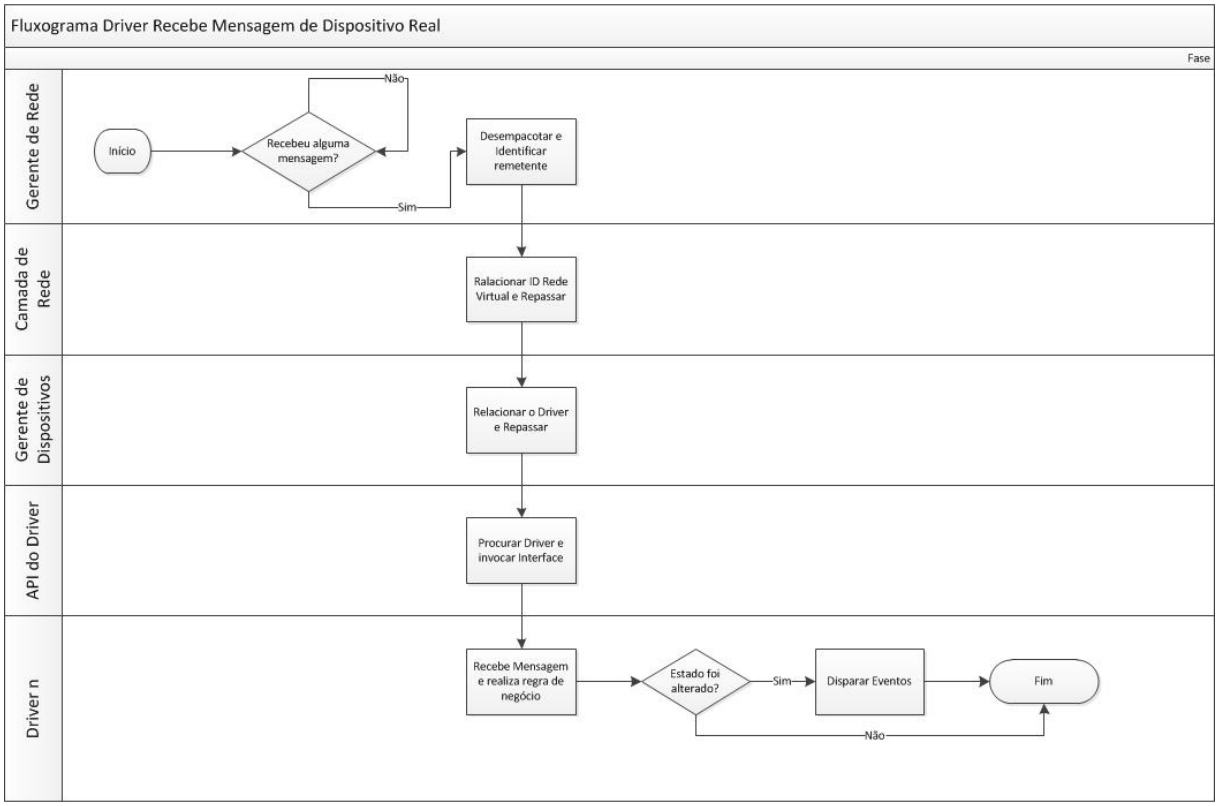

Figura 4.13: Fluxograma Recebendo Mensagem do Dispositivo Real

Capítulo

5

EXPERIMENTOS E RESULTADOS

Este capítulo apresenta os experimentos e resultados relacionados ao desenvolvimento de aplicações usando o framework Manna-X. Os experimentos foram divididos em dois cenários para uma análise detalhada sobre o framework. A primeira parte do capítulo trata do cenário de teste para avaliar a performance do framework. A segunda parte do capítulo trata, em particular, do desenvolvimento e implementação do framework em um ambiente real para ser provado sua validade. Por fim, uma comparação com outros trabalhos da literatura mostra o diferencial de uma solução para AI utilizando o framework Manna-X.

5.1 AVALIANDO O DESEMPENHO DO MANNA-X

A avaliação de sistemas computacionais pode ser realizada por meio de diferentes tipos de ferramentas tais como a experimentação real, o uso de ambientes de simulação e a elaboração de modelagem matemática. A fim de analisar a validade e o desempenho do framework proposto, com respeito à eficácia e eficiência, do ponto de vista global no contexto de ambientes inteligentes, foi-se utilizada uma experimentação real no qual o framework pôde ter sido testado.

Para ajudar nas análises, o método GQM (Basili et al., 1994) foi utilizado. A ideia básica de GQM é derivar métricas de software a partir de perguntas e objetivos. Este método foi originalmente criado por Victor Basili e Weis como resultado de experiências

práticas e pesquisas acadêmicas. Método apoia a definição *top-down* do processo de medição e a análise *bottom-up* dos dados resultantes.

GQM parte do princípio de que toda coleta dos dados deve ser baseada em um fundamento lógico, em um objetivo ou meta, que é documentado explicitamente. O primeiro passo nessa abordagem é definir metas de alto nível a serem alcançadas na avaliação. Após a identificação das metas, um plano GQM é elaborado para cada meta selecionada. O plano consiste, para cada meta, em um conjunto de questões quantificáveis que especificam as medidas adequadas para sua avaliação. As questões identificam a informação necessária para atingir a meta e as medidas definem operacionalmente os dados a serem coletados para responder as questões.

Para analisar o framework, foi definida uma meta exibida na Tabela 5.1. As métricas usadas para analisar tal meta representam de forma quantitativa a eficiência do Manna-X. Estas métricas foram aplicadas durante a execução dos testes. Os testes foram realizados de uma maneira genérica em um cenário simulado. Foi criado um Gerente de Rede genérico, que simula uma rede real. Sua função é chamar métodos do Kernel de maneira aleatória, e dessa maneira simulando uma rede. Em linhas gerais o Gerente de Rede genérico fica criando novos dispositivos, avisando a saída e entrada de dispositivos já conhecidos, e enviando mensagens dos dispositivos simulados para o Kernel. As ações são disparadas a cada 1 segundo.

Além disso, foram criados também Drivers genéricos que ficam simulando de maneira aleatória a vida de um Driver. Os Drivers executam ações também em intervalos de 1 segundo. Desta maneira o cenário simulado imita a utilização do framework enquanto seus dados de eficácia e eficiência são coletados. Para a construção do cenário primeiramente instala-se o bundle do Kernel junto com os bundles de interface. Logo em seguida instala-se manualmente o Gerente de Rede genérico. Os Drivers são instalados automaticamente pelo Gerente de Configuração, todos eles foram igualmente implementados.

Esse cenário foi testado alterando o número de nós gerado pelo Gerente de Rede, cada teste foi realizado dez vezes. Os números de nós testados foram: 5, 10, 50, 100, 1000 nós, totalizando 50 testes. Cada teste teve duração mínima de 2 minutos, o procedimento para o encerramento dos testes foi dois minutos após a criação de todos os nós do teste, por exemplo, se o Gerente de Rede no teste com dez nós, demorar cinco minutos para instanciar todos os dez nós, o tempo total do teste seria cinco minutos mais dois minutos, totalizando sete minutos.

Tabela 5.1: Avaliação dos resultados utilizando GQM

Meta 1: Analisar a viabilidade de instalação, execução, escalabilidade e a performance do Manna-X							
Perguntas	**Métrica**		**5**	**10**	**50**	**100**	**1000**
1.1. O tempo de resposta do Manna-X é aceitável?	Tempo de resposta entre a mensagens chegar no GD até o Driver (μs)	Média	569.25	855.41	768.40	502.92	655.33
1.2. Como se comporta o Manna-X conforme novos nós são inseridos?		Max	15227.93	96938.49	109708.79	15519.03	35539.54
1.3. Quantos nós o Manna-X suporta?		Min	318.30	337.81	354.75	345.51	344.49
1.4. Os requisitos de memória e CPU exigidos pelo Manna-X são aceitáveis considerando as características das AmI?		Variância	1500994.45	2.22	2.23	420767.66	1431279.52
		Desvio Padrão	1225.15	4718.32	4724.96	648.66	1196.36
	Tempo de resposta entre a mensagens chegar no Driver até o GD (μs)	Média	25.20	43.05	31.47	28.15	10.39
		Max	604.78	36788.13	32291.27	63267.72	27287.68
		Min	10.26	7.70	3.59	1.02	1.02
		Variância	756.33	308103.12	63515.68	389843.48	49990.05
		Desvio Padrão	27.50	555.07	252.02	624.37	223.58
	Tempo de instalação dos Drivers (ms)	Média	23.09	17.12	14.77	16.93	43.90
		Max	79.86	77.69	92.31	249.44	463.57
		Min	10.85	10.50	9.65	9.11	9.53
		Variância	357.46	179.69	87.16	366.89	1305.85
		Desvio Padrão	18.90	13.40	9.33	19.15	36.13
	Consumo de memória Heap usado pelo framework (Mb)	Gráficos					
	Carregamento da CPU(%)						
	Número de Classes Carregadas						

Os testes foram executados usando sobre IDE Eclipse[1] em um notebook Core 2 duo de 2.00Ghz e 4Gb de RAM rodando o sistema operacional Windows 7 64 bits e a plataforma Java versão 7. Para executar os módulos OSGi foi-se utilizado a plataforma OSGi Apache Felix[2] junto com a ferramenta Bndtools[3].

Como visto na Tabela 5.1 a busca por responder a Meta principal do GQM formalizou quatro perguntas que dessas quatros perguntas originaram-se seis métricas de avaliação. Na Tabela 5.1 cada métrica avalia uma quesito. Para um melhor entendimento as três últimas métricas não foram colocadas na tabela e serão mostradas em gráficos posteriormente. As primeiras métricas foram divididas em Média, Valor Máximo, Valor Mínimo, Variância e Desvio Padrão. Elas tratam sobre o tempo entre troca de mensagens entre os extremos do Kernel e a instalação de um Driver.

Os testes mostram que o número de nós não influencia diretamente nos tempos do framework, além disso, o framework conseguiu atingir um tempo bom tanto para as trocas de mensagens como para a instalação dos drivers.

Para verificar a pergunta 1.4, as três últimas métricas foram aferidas e somente um gráfico por teste será mostrado no trabalho. Os gráficos estão no Apêndice A. Os testes mostraram o que era esperado, quanto mais nós maior o consumo de memória, maior a criação de classes e maior o uso da CPU. A leitura do gráfico é feito da seguinte maneira: no gráfico de Memória a parte verde representa o espaço livre na memória principal que o programa possui e em azul a parte usada em Mb. No gráfico de uso da CPU a linha verde ostra o carregamento da CPU em porcentagem. Por fim, no gráfico do número de classes instanciadas pelo framework é mostrado em azul, e o número total de classes da JVM em verde.

Os gráficos de uso de memória mostram claramente que até 100 nós o uso médio fica abaixo dos 10 Mb, somente no gráfico de 1000 nós que esse número aumenta gradativamente até 60Mb, porém no fim se estabiliza em 25 Mb. Nesses gráficos dá para ver claramente o uso do Garbage Collector do Java nas quedas bruscas de consumo de memória.

O número de classes também aumenta linearmente conforme aumenta o número de nós da rede, pode-se perceber que no gráfico com 5 nós o número de classes estabiliza perto das 200 classes carregadas pela JVM, já com 1000 nós esse número passa para mais de 2000 classes.

[1] `http://www.eclipse.org/`
[2] `http://felix.apache.org/site/index.html`
[3] `http://bndtools.org/`

A carga da CPU em linhas gerais apresenta-se homogênea em todos os testes, pode-se perceber que quanto mais nós maior o uso, devido aos Drivers estarem executando. Pode-se perceber picos de uso enquanto o framework encontra-se instalando os Drivers, logo após o uso estabiliza-se.

Os testes realizados mostram apenas alguns aspectos de utilização do framework, obviamente o consumo de memória, e CPU pode aumentar devido a que os Drivers são implementados por terceiros. OS testes foram realizadas fazendo com que se tenha uma ideia do consumo médio de uma solução usando o framework Manna-X. Os resultados se mostraram satisfatórios no contexto de ambientes inteligentes, pois os tempos foram mínimos, e o consumo e uso de memória e CPU foram baixos perto do que computadores atuais podem ter.

5.2 MANNA-LAB, USANDO O MANNA-X NO DESENVOLVIMENTO DE UM LABORATÓRIO INTELIGENTE

A solução experimental utilizando o Framework Manna-X foi realizada no laboratório Manna[4], localizado na Universidade Estadual de Maringá. O protótipo desenvolvido serviu como prova de conceitos e contou com a participação de alunos de iniciação científica no papel de desenvolvedores de ambientes inteligentes.

Cada aluno recebeu a missão de desenvolver um dispositivo inteligente que pudesse ser integrado ao framework. Desta maneira, pode-se observar como um desenvolvedor faria uso do Manna-X para integrar o dispositivo real programando o Driver de Dispositivo.

Outros pesquisadores receberam a missão de implementar o Kernel e a camada de aplicação. Os dispositivos criados pelos alunos foram:

- A porta inteligente;
- A lâmpada inteligente;
- Sensores de umidade e temperatura;
- Sensores de campo magnético;
- Sensor de identificação humana;

[4] `http://www.din.uem.br/~manna/`

- Mini estação meteorológica;
- Sensores de gás.

O Kernel foi implementado em um computador Desktop Intel Core i5 com 4Gb de memória RAM, o sistema operacional utilizado foi o Linux Ubuntu 11.10 e a plataforma Java versão 7. Para programar em OSGi foram utilizados a plataforma Eclipse[5] junto com a ferramenta Bndtools[6] , e a plataforma OSGi Apache Felix[7].

No ambiente somente duas redes foram utilizadas: a rede ZigBee e a rede de sensores sem fio MICAz. Por esta razão, no computador onde o Kernel foi implementado dois adaptadores foram conectados. Esses adaptadores conectados são módulos que possuem antenas e foram ligadas através de portas USB no Manna-X Server. Dois Gerentes de Rede foram desenvolvidos para gerenciarem tanto a rede ZigBee quanto a rede dos dispositivos MICAz.

A solução criada tem o objetivo de proporcionar aos usuários do laboratório de pesquisa, um laboratório inteligente, onde os dispositivos possam ser acessados por páginas Web e por meio de gestos e voz. Além disso, algumas tarefas foram programadas como, por exemplo: acender a luz quando a porta se abrir, apagar a luz quando ninguém estiver no laboratório, acender e apagar a luz, abrir e fechar a porta por comandos de voz e gestos.

Outra funcionalidade do laboratório inteligente é poder escolher quais dispositivos podem ser publicados para a Internet, tornando assim os dispositivos parte da Internet das Coisas a fim de se comunicarem. Para que tal cenário se realizasse a arquitetura da Camada de Aplicação foi criada como pode ser visto na Figura 5.1.

A solução mostrada na Figura 5.1 mostra o que foi implementado à nível de Camada de Aplicação e Gerentes de Rede. A aplicação desenvolvida é um módulo OSGi que serve de interface para o usuário. O módulo contém um servidor Web que possibilita a interação dos usuários bem com a navegação pelas páginas do Drivers de Dispositivos. Além disso, a solução permite disponibilizar quais serviços dos dispositivos podem ser acessíveis por meio da API REST fornecida, em prol da Internet das Coisas.

O módulo interno da Camada de Aplicação chamado Inteligência Ambiental serve para organizar os dispositivos na aplicação de modo a fornecer uma interface adaptada para Laboratórios Inteligentes. A inteligência de um laboratório não é muito diferente de uma casa. Ela deve possibilitar uma interface intuitiva para os dispositivos, possibilitar o

[5] `http://www.eclipse.org/`
[6] `http://bndtools.org/`
[7] `http://felix.apache.org/site/index.html`

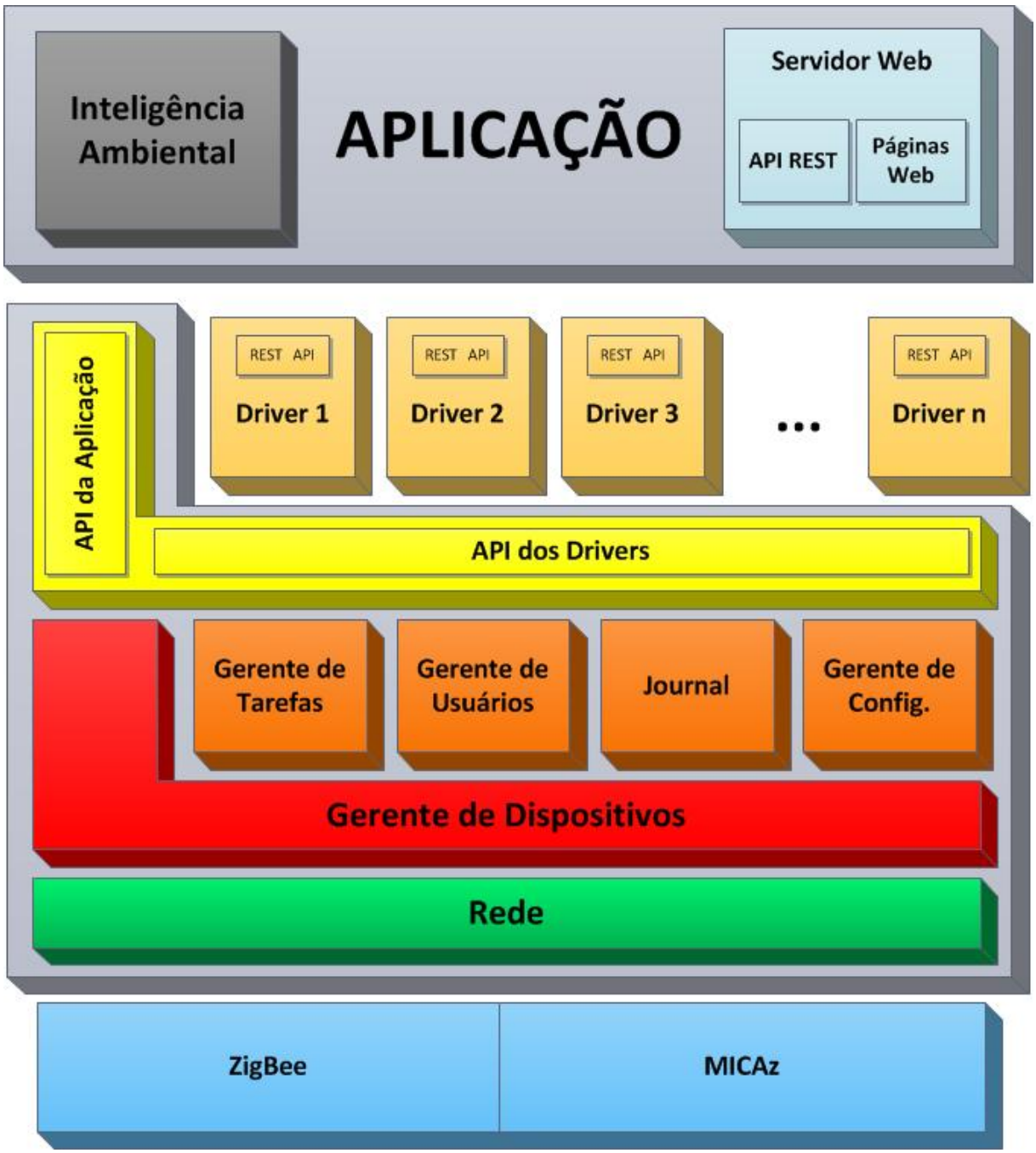

Figura 5.1: Solução Manna-X Lab

cadastramento e permissões de usuários no laboratório, bem como proporcionar um bom ambiente de trabalho.

As próximas seções detalham como foi o desenvolvimento dos dispositivos do laboratório feito pelos alunos de iniciação científica.

5.2.1 A Porta Inteligente

Para provisionar a porta com inteligência, foi utilizado um nó sensor WaspMote[8], que possui acelerômetro, relê e uma antena ZigBee, podendo ser visto na Figura 5.2. O nó fica acoplado na parte traseira da porta, onde tem uma ligação com a fechadura eletrônica. Como fonte de energia o nó utiliza uma bateria.

Para abrir a porta o nó simplesmente deve deixar passar corrente para a fechadura por meio do relê. Para verificar se a porta está aberta ou fechada, o acelerômetro presente na placa WaspMote foi utilizado e através de cálculos é possível inferir se a porta está ou aberta ou fechada.

A única mudança efetuada para que a porta se comunique com o framework foi passar a enviar seus pacotes para o nó coordenador da rede ZigBee que está ligado diretamente ao framework. O código do dispositivo foi feito em C e possui serviços de abertura de porta e verificação de seu estado (aberto ou fechado).

Figura 5.2: Porta Inteligente

5.2.2 A Lâmpada Inteligente

O laboratório de pesquisa possui uma lâmpada fluorescente que era acionada por um interruptor comum. A integração da lâmpada ao ambiente inteligente também foi feita utilizando um nó WaspMote, que possui um relê e uma antena ZigBee. Assim, o programa contido no nó ao receber comandos via ZigBee, libera ou fecha a corrente utilizando o relê, fazendo assim com que as lâmpadas acendam ou apaguem.

[8] `http://www.libelium.com/products/waspmote`

O nó implementado pode ser visto na Figura 5.3. Na figura pode ser visto somente o nó, junto com o relê, a antena, a bateria que fornece sua energia e o cabo preto que sai do nó. Esse cabo está ligado diretamente na lâmpada. Assim, pode-se dizer que a solução funciona como um interruptor, que ao receber mensagens de seu Driver acende, ou apaga a luz.

Figura 5.3: Nó Controlador da Lâmpada Inteligente

5.2.3 Sensores de Umidade e Temperatura

O sensoriamento da temperatura e da umidade foi realizado a partir de três nós MICAz[9] que possuem sensores de temperatura e umidade acoplados na placa expansora. Cada nó foi colocado em uma parte do laboratório para tal monitoramento. O nó troca informações com a base, conectada ao servidor, enviando periodicamente tais sensoriamentos. Os nós se comunicam por um protocolo proprietário porém utilizam o padrão IEEE 802.15.4 e eles podem ser vistos na Figura 5.4.

[9] `http://www.memsic.com/products/wireless-sensor-networks/wireless-modules.html`

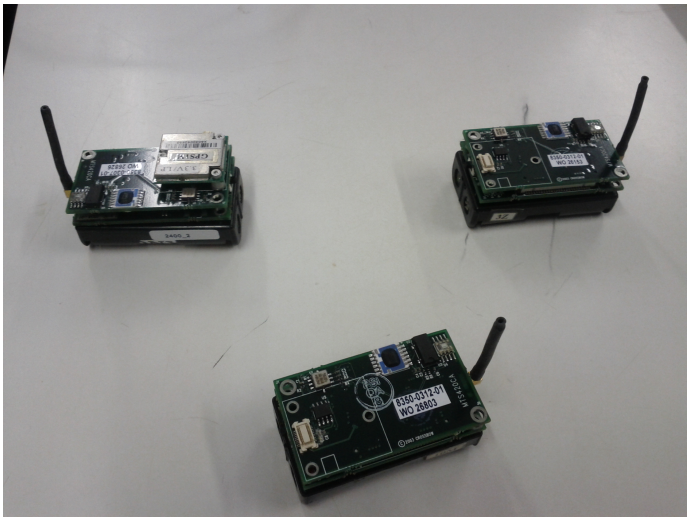

Figura 5.4: Nós Sensores de Temperatura e Umidade

5.2.4 Sensores de Campo Magnético

O sensor de campo magnético tem por finalidade ser utilizado para detectar a presença de carros no estacionamento. Como dentro do laboratório e nem perto dele existe estacionamento, sua utilização é só para fins experimentais. Os nós utilizados são nós MICAz com uma placa expansora que possui um magnetômetro de dois eixos. A finalidade dentro do laboratório também pode ser vista para ajudar o pesquisador a testar suas invenções já utilizando a solução Manna-X. A Figura 5.5 mostra a foto dos nós.

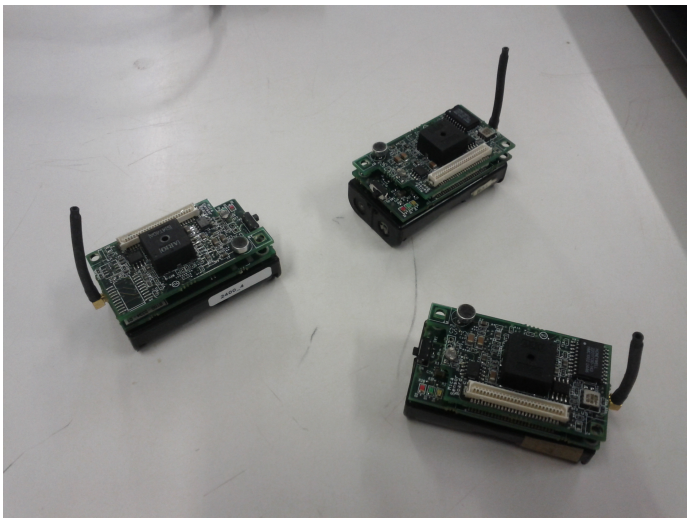

Figura 5.5: Sensores de Campo Magnético

5.2.5 Mini Estação Meteorológica

A Mini Estação Meteorológica é um trabalho resultante de um dos projetos do laboratório de pesquisa feito pelos alunos de iniciação científica(Goes, 2011). Por ser feito em um único nó WaspMote e possuir ZigBee para a comunicação, seu trabalho foi utilizado na solução proposta para fins de teste mas também de monitoração do ambiente.

Sua ideia na solução é prestar serviços climáticos para informar o usuário. O nó sensor possui acoplado um sensor de temperatura e umidade, um anemômetro, um cata-vento e um pluviômetro. A Figura 5.6 mostra o dispositivo implementado.

Figura 5.6: Mini Estação Meteorológica

5.2.6 Sensor de Gás

O sensoriamento de gases foi desenvolvido a partir de um nó WaspMote que possui uma antena ZigBee, uma placa de expansão para sensores de gás. A placa de sensor Waspmote gases foi projetada para monitorar os parâmetros ambientais, como temperatura, umidade, pressão atmosférica e 14 diferentes tipos de gases. Ele permite a inclusão de 6 sensores de gases, ao mesmo tempo, o regulamento de seu poder é através de um sistema de detectores de estado sólido e a ampliação do sinal de saída de cada um deles através de um estágio de amplificação não-inversora de um ganho máximo de 101 controlada por um potenciômetro digital configurável através do Barramento de Circuito Inter-Integrado, I2C. A Figura 5.7 mostra o sensor implementado, e os gases que podem ser monitorados são:

- Monóxido de carbono - CO;

- Dióxido de carbono - CO_2;
- Oxigênio molecular - O_2;
- Metano - CH_4;
- Hidrogênio molecular - H_2;
- Amônia - NH_3;
- Isobutano - C_4H_{10};
- Etanol - CH_3CH_2OH;
- Tolueno - $C_6H_5CH_3$;
- Sulfeto de hidrogênio - H_2S;
- Dióxido de Nitrogênio - NO_2;
- Ozônio - O_3;
- Compostos Orgânicos Voláteis (COVs).

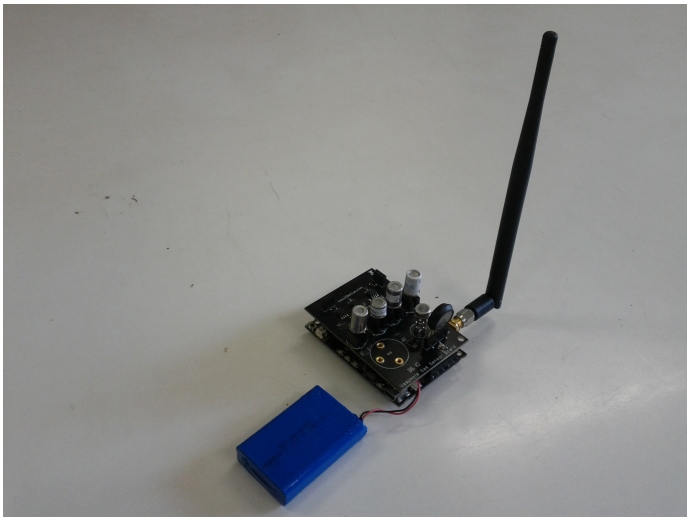

Figura 5.7: Sensor de Gases

5.2.7 Sensor de Identificação Humana

O sensor de identificação humana tem o objetivo de prover a ubiquidade da solução. Esse identificador deve ser capaz de detectar os usuários do ambiente e poder se comunicar com as pessoas. Para tal solução, o sensor Kinect[10] da Microsoft foi utilizado, prestando serviços como identificação humana, identificação de comandos por voz e gestos.

Na solução sua função é a de um dispositivo sensor, ele serve para detectar a presença do usuário no laboratório e suas ações como, entrar e sair no ambiente, acenar com a mão, fazer gestos programados para a realização de uma interface real com o usuário, bem como também receber comandos de voz para servir de entrada no sistema. Devido ao fato de ser uma criação da empresa Microsoft, foi optado por utilizá-lo em ambiente Windows pela facilidade de programação e suporte oferecido. O nó então na verdade passa a ser o Kinect e o computador que possui o sistema operacional Windows.

Para a comunicação com o framework foi necessário a criação de um Gerente de Rede especial para achá-lo no computador da rede local e assim por meio de Sockets trocar mensagens, fazendo com que o Kinect e o computador que o conecta sejam um só dispositivo para o servidor. Pode-se encarar este gerente como um gerente de redes de Kinects, pois foi projetado para possuir quantas ligações forem necessários com nós Kinects. A Figura 5.8 mostra a solução.

Figura 5.8: O Kinect ligado ao computador formando juntos o nó Kinect

[10] `http://www.xbox.com/pt-BR/Kinect`

5.3 COMPARANDO COM OUTROS TRABALHOS

A solução proposta serve como prova de conceito para a utilização do framework implementado. O framework por ser genérico pode ser utilizado em quaisquer ambientes tais como: casas, lojas, bares, hotéis, laboratórios, estádios de futebol, etc. A utilização do framework pelos estudantes e pesquisadores que utilizaram o laboratório mostrou-se de fácil entendimento e programação.

A fim de comparar o presente trabalho com outros encontrados na literatura, foram selecionados quatorze características importantes para uma solução em ambientes inteligentes e assim aferidos e tabulados dezenove trabalhos encontrados na literatura. Outros trabalhos foram encontrados, porém somente alguns foram escolhidos por serem mais parecidos com o Manna-X. Na 5.3 à dissertação pode ser encontrado o resultado.

As características escolhidas partiram da aferição e do desenvolvimento do material teórico encontrado na literatura, elas são: Domínio da Aplicação, essa característica não é uma aferição comparativa, simplesmente serve como análise para identificar que tipo de aplicação foi feita; Trata a Interoperabilidade, essa característica é utilizada para saber se a solução proposta trata a questão da interoperabilidade dos dispositivos de alguma forma; Possui Heterogeneidade, essa característica avaliadora mede se existe na solução desenvolvida diversos tipos de dispositivos, ou seja, se a solução suporta dispositivo heterogêneos.

Outra característica aferida foi o Tipo de Rede, nessa foi verificada a variedade de redes utilizadas pela solução, ela é importante para analisar o escopo dos dispositivos no que se refere a que tipo de rede eles devem utilizar; Descoberta de Serviços Dinamicamente, muitas soluções não permitem a descoberta de serviços dinamicamente, somente em tempo de construção, essa característica é importante para ambientes inteligentes pois são muito dinâmicos; Descoberta de Dispositivos Dinamicamente, outra característica importante em ambientes dinâmicos devido a mobilidade dos dispositivos nas redes.

Internet das Coisas, essa característica avalia se a solução proposta permite meios para que a tendência da Internet das Coisas seja concretizada; Extensibilidade, característica importante, pois significa dizer se a solução proposta pode aumentar o número de dispositivos dinamicamente, ou seja, expandir o número de dispositivos; Arquitetura da Solução, nessa característica somente é tabulada a questão de arquiteturas centralizadas e descentralizadas, a principal diferença na divisão dessa característica se deve pelo fato de existir um único ponto central na solução por onde a tomada de decisão e a passagem dos dados é realizada.

Tabela 5.2: Tabela Comparativa de Soluções para AI

Características \ Soluções	**Domínio da Aplicação**	**Trata Interoperabilidade**	**Possui Heterogeneidade**	**Tipo de Rede**	**Descoberta de Serviços Dinamicamente**	**Descoberta de Dispositivos Dinamicamente**	**Internet das Coisas**	**Extensibilidade**	**Arquitetura da Solução**	**Comunicação entre Dispositivos**	**OSGi**	**Extensão do Dispositivo**	**Interface Humano–Computador**	**Ciente de Contexto**
Manna-X	Genérica para qualquer ambiente	Sim	Sim	Qualquer Tipo	Permitido por meio de OSGi	Sim	Sim	Sim	Centralizada	OSGi e API REST	Sim	Sim através dos Drivers	Páginas Web	Sim
A Secure Ambient Assisted Living (AAL)	AmI para doentes em Hospitais e Casas	Sim, com adaptadores	Sim	Rede IP	Implementação Fechada	Não mencionado	Não	Não	Centralizada	Não	Sim	Não	Páginas Web	Não mencionado
Application Architecture for Ambient Intelligence	Ambientes Cientes de Contexto	Não mencionado	Não mencionado	Não mencionado	Não	Não	Não	Não	Centralizada	Não	Não	Não	Não mencionado	Sim
Virtual Sensors: Rapid Prototyping	AmI com Kinect	Não	Não	Não mencionado	Não	Não	Não	Não	Centralizada	Não	Não	Não	Celular	Não
Towards a Reference Middleware Architecture for Ambient Intelligence Systems	Criação de um middleware para AmI	Não mencionado	Sim	Não mencionado	Sim	Não mencionado	Não	Não mencionado	Centralizada	Sim	Futuramente	Não	PDA	Sim
ubiHome: An Infrastructure for Ubiquitous Home Network Services	Infraestrutura genérica para Ambientes Ubíquos	Sim, com adaptadores	Sim	Rede IP	Sim	Sim	Não	Sim	Descentralizada	Não	Não	Não	PC	Sim
Towards a RESTful Plug and Play Experience in the Web of Things	Ambientes Inteligentes com IoT	Sim, com adaptadores	Sim	Rede IP	Sim	Sim	Sim	Sim	Descentralizada	Sim API REST	Não	Não	Páginas Web	Permite
DomoML - an Integrating Devices Framework for Ambient Intelligence Solutions	Framework para integração de dispositivos em Ambientes Inteligentes	Sim	Sim	Não mencionado	Sim	Sim	Não	Sim	Centralizada	Sim API REST	Não	Não	Páginas Web	Permite
RESTful Integration of Heterogeneous Devices in Pervasive Environments	Solução integradora de dispositivos heterogeneos	Sim, com adaptadores	Sim	Todos com Adaptadores para rede IP	Sim	Sim	Sim	Sim	Centralizada*	Sim API REST	Não	Não	Celular	Sim
Modeling the Ambient Intelligence Application System Concept, Software, Data, and Network	Modelo de Ambiente Inteligente	Sim, com adaptadores	Sim	Todos com Adaptadores para rede IP	Sim	Sim	Não	Sim	Centralizada	Não	Sim	Não	PDA	Sim
TheWeb of Things: interconnecting devices with high usability and performance	Web das Coisas	Não	Sim	Rede IP	Sim	Não mencionado	Sim	Sim	Descentralizada	Não	Não	Não	Páginas Web	Não
Home Networking and Control based on UPnP: An Implementation	Ambientes Inteligentes com UPnP	Sim, com adaptadores	Sim	UPnP	Sim	Sim	Não	Sim	Centralizada	Não mencionado	Não	Não	Páginas Web	Não
A Resource Oriented Architecture for the Web of Things	Web das Coisas	Sim, com adaptadores	Sim	Rede IP	Sim	Não	Sim	Sim	Descentralizada	Sim	Não	Não	Páginas Web	Não
An Object-based Virtual Network in Ubiquitous Computing Environment	Middleware para Ambientes Inteligentes	Sim	Sim	Qualquer Tipo	Não	Sim	Não	Sim	Descentralizada	Sim	Não	Não	PC	Não
FamiWare: a family of event-based middleware for ambient intelligence	Middleware para Ambientes Inteligentes	Sim, com adaptadores	Sim	Todos com Adaptadores para rede IP	Sim	Sim	Não	Sim	Descentralizada	Sim	Não	Não	Não mencionado	Sim
Domain Independent Architecture and Behavior Modeling for Pervasive Computing Environments	Genérica para qualquer ambiente	Sim	Sim	Qualquer Tipo	Não	Não	Não	Sim	Centralizada	Não	Não	Não	PC e Smartphones	Sim
Embedding Internet Technology for Home Automation	Solução para AmI com WoT	Sim	Somente IPv6	Rede IP, IPv6	Sim, do sistema	Sim	Sim	Sim	Descentralizada	Sim	Não	Não	Páginas Web	Não
Knowledge-Aware and Service-Oriented Middleware for deploying pervasive services	Middleware para Ambientes Inteligentes	Não	Não	ZigBee	Sim	Sim	Sim	Sim	Centralizada	Não	Não	Não	Páginas Web	Não
A Middleware for Intelligent Environments and the Internet of Things	Ambientes Inteligentes com IoT	Sim	Sim	Qualquer Tipo	Não	Não	Sim	Não	Centralizada	Não	Não	Não	PC, Páginas Web	Sim

Comunicação entre Dispositivos, essa característica analisa se a solução permite com que os dispositivo se comuniquem; OSGi, verifica se a solução proposta utiliza a tecnologia OSGi; Extensão dos Dispositivos, essa característica significa dizer se a solução proposta possui de algum modo algo para dar mais suporte aos dispositivos estendendo suas funcionalidades e possibilidades; Interface Humano-Computador, analisa quais os meios de comunicação que a solução possui para com o usuário humano; e por fim Ciência de

Contexto, característica importante em ambientes inteligentes, ela avalia quais soluções dão suporte a essa característica.

A comparação realizada não visa julgar a realidade do momento em ambientes inteligentes, pois não foram avaliados todos os trabalhos da literatura, porém serve como base para análise de características importantes, qual se dá maior peso na hora do desenvolvimento.

Pode se perceber que muitas soluções já se preocupam em resolver problemas como a interoperabilidade e heterogeneidade dos dispositivos. Porém muitas soluções focam em limitar a comunicação dos dispositivos somente na rede IP, mesmo que através de adaptadores. Esse fato pode ser compreendido pela facilidade e conhecimento de se trabalhar com a rede IP, que já é bastante difundida no mercado.

Outra análise a ser feita é a descoberta de serviços e dispositivo e extensibilidade, como visto, a grande maioria já toma essas características como essenciais em suas aplicações, embora não exista uma relação certa, pode-se perceber que sempre onde uma característica está presenta a outra também se faz.

A tendência de expor seus dispositivos para a Internet das Coisas, ainda se mostra pouca na literatura por se tratar de um tema novo, ainda mais integrando com ambientes inteligentes. Como pode ser visto também na maioria dos casos onde se tem Internet das Coisas, geralmente se limita a usar somente a rede IP, salve algumas exceções.

Os trabalhos escolhidos para análise mostram-se bem divididos na relação arquitetura centralizada ou descentralizada, pode-se perceber que quando se trabalha com Internet das Coisas geralmente as arquiteturas são descentralizadas, e utilizam a rede IP como protocolo de comunicação. A comunicação entre os dispositivos em ambientes descentralizados como aferido mostra que todos permitem a comunicação, já nos centralizados apenas alguns permitem.

Poucas soluções utilizam OSGi atualmente, pode-se perceber que sua utilização depende muito da modelagem da solução visto que ela pode ser feita de inúmeras maneiras. A interface entre homens e máquinas é em sua maioria proporcionada por meio de páginas Web e como pôde ser visto, ambientes cientes de contexto foram encontrados na maioria dos trabalhos revisados.

A última característica analisada refere-se a extensão do dispositivo, essa característica não pode ser percebida por nenhum dos middlewares que faz com que a solução Manna-X diferencie-se das outras, provendo meios e capacidades para ajudar o ambiente a ficar mais rico de inteligência, usabilidade e funcionalidades. O framework originado resulta de uma união de características analisadas da literatura que foram julgadas mais importantes com

a finalidade de ajudar desenvolvedor de ambientes inteligentes e fabricantes a evoluírem o campo da Engenharia de Computação Invisível.

Capítulo

6

CONCLUSÕES E TRABALHOS FUTUROS

Ambientes Inteligentes é uma área de pesquisa ainda considerada recente, com grandes desafios a serem vencidos para que possa ser amplamente implementados e utilizados. As aplicações ubíquas cientes de contexto são capazes de ofertar suas tarefas o tempo todo e em todo lugar aos seus usuários finais, adaptando-as às características e necessidades dos mesmos de acordo com a personalida de cada participante e a disponibilidade de recursos apresentada pelo ambiente físico associado.

Por isso, a maneira de construir tais ambientes não é algo trivial e muitas dificuldades foram percebidas ao longo da pesquisa, tais como: a solução da interoperabilidade entre os dispositivos, mobilidade, extensibilidade, flexibilidade, modularização, serviço de descoberta de rede, auto-organização, ciência de contexto, interface intuitiva, proteger a rede de pessoas e dispositivos mau intencionados, usabilidade, permitir ou ter alguma inteligência ambiental, conseguir tratar a dinamicidade dos dispositivos, ser ubíquo; não ter um alto custo de implantação e tratar a heterogeneidade dos dispositivos.

Além disso, algumas soluções foram analisadas e muitos tipos de arquitetura e suas misturas observadas, como: Arquitetura Orientada a Eventos, Arquitetura Orientada a Serviço, Arquitetura Centralizada e Descentralizada, Arquitetura Orientada a Recursos, Arquitetura Orientada a Aspectos, Arquiteturas *Publish* e *Subscribe*, Arquiteturas voltadas a Web das Coisas(RESTful), em fim, inúmeras possibilidades de modelagem e suas combinações que o desenvolvedor de ambientes inteligentes deve analisar e escolher.

Como parte do processo de aquisição de conhecimentos, algumas questões científicas foram identificadas e enunciadas na seção 3.1. Segundo o que foi estudado pode se perceber que é possível criar soluções genéricas para Ambientes Inteligentes. Num ambiente podem existir diversos tipos de dispositivos, tanto sensores quanto atuadores e mistos. Não existe uma única maneira para integrar os dispositivos, como visto na literatura cada pesquisador desenvolve o ambiente de um modo peculiar, não podendo ser aferido qual o correto. O modo de gerenciar os dispositivos também fica a critério do desenvolvedor, ambas as arquiteturas centralizadas e descentralizadas possuem suas vantagens e desvantagens, cabe a cada solução escolher a que melhor se encaixa.

A interface virtual dos dispositivos se feita de maneira genérica mostra que a página Web é a mais indicada pela flexibilidade de poder ser apresentada em qualquer plataforma. As interfaces reais como voz e gestos também são muito indicadas para prover um ambiente ubíquo. A comunicação entre os dispositivos é de extrema importância para que os dispositivos cooperem no ambiente e por isso é essencial. O modo que eles se comunicariam pode ser variado, cada pesquisador mostrou na literatura uma forma.

A descoberta dos dispositivos e de seus serviços prestados deve ser obrigatória e principalmente dinâmica, pois como visto esses ambientes são constantemente alterados. Quanto ao acesso direto aos dispositivos não existe uma regra, porém o acesso direto propicia a um descontrole da solução e risco a invasões. O controle do acesso ao sistema do ambiente depende de como foi desenvolvido a solução, porém sistemas centralizados proporcionam maior segurança.

A integração com sistemas e dispositivos legados pode ser feito com ajuda de adaptadores, mas como dito cada solução deve ser feita para aceitar tais adaptadores. Esse problema é originário de dois outros problemas que são a interoperabilidade e a heterogeneidade dos dispositivos, embora muita são as soluções encontradas na literatura, poucas conseguem integrar diversos dispositivos de maneira simples sem limitar o uso de somente uma tecnologia de comunicação, que não é o ideal regra dispositivos de terceiros. Para manter o sistema atualizado devido a dinamicidade e mobilidade dos dispositivos uma boa maneira é desenvolver soluções orientadas a eventos, como no caso da solução proposta.

A fim de ajudar e resolver os problemas citados, que um framework/arcabouço foi projetado e foi chamado de Manna-X. O framework busca a união de dois mundos que andam separados, que é o do desenvolvedor de ambientes inteligentes e o do fabricante de dispositivos. O principal problema encontrado é a questão da interoperabilidade, ou seja, a falta de integração entre o ambiente e os dispositivos.

O desenvolvedor de ambientes inteligentes quando projeta um ambiente deve levar em conta o protocolo de comunicação com os dispositivos para fazer com que eles troquem

informações, porém, com dispositivos interoperantes não é possível realizar nenhuma integração. Por isso, o framework proposto visa criar meios para que essa união seja feita, e que o desenvolvedor não se preocupe mais com questões de baixo nível.

A metodologia de desenvolvimento para AI proposta visa mudar os paradigmas de como construir ambientes inteligentes, de modo que o desenvolvedor de ambientes inteligentes ao utilizar-se do framework não tenha preocupação com a compatibilidade dos dispositivos, e foque seus esforços, energia e tempo na solução, criando assim ambientes mais interativos, dinâmicos, inteligentes e interessantes. Um dos objetivos do framework é criar uma rede virtual de dispositivos que podem ser acessados por meio de software, facilitando a vida do "programador" de ambientes inteligentes.

A solução quando comparada com outras encontradas na literatura obteve seu diferencial no que se refere ao auxílio dado também ao fabricante de dispositivo. Quando utilizado o framework, os dispositivos contam com uma área "nas nuvens" para que seus dispositivos aumentem seu poder computacional e de armazenamento de dados. Mais do que isso, são esses "espaços" que viabilizam a integração dos dispositivos ao framework, criando uma extensão virtual para o dispositivo chamado de Driver de Dispositivos.

Por meio desses Drivers de Dispositivos os dispositivos podem trocar informações "nas nuvens", pesquisar por outros dispositivos, prestar serviços e utilizar serviços de outros dispositivos. Os Drivers servem como uma virtualização do dispositivo dentro do framework Manna-X, podendo o dispositivo realizar processamentos e armazenar dados.

Pode ser percebido que, quando bem projetado o dispositivo e seu Driver, existe um menor tráfego de troca de mensagens nas redes reais, pois como os Drivers ficam contidos dentro de uma mesma máquina a troca de mensagens por eles tem o tempo de uma chamada de método, não havendo ainda perdas de pacotes. Assim, ao invés dos dispositivos trocarem informações utilizando sua rede real como ZigBee, X10, Z-Wave, eles podem se comunicar a nível de software por meio dos Drivers, diminuindo o tráfego das redes reais.

Outra vantagem ainda obtida pela utilização dos Drivers de Dispositivos além de estender poderes dos dispositivos, o fabricante conta com um espaço no qual pode aumentar a experiência do usuário com seu dispositivo, já que os dispositivos podem ser acessados por meio de páginas Web, o fabricante poderia criar novas interações, melhorando a relação com o usuário final.

O framework Manna-X, além do que já foi mencionado, buscou integrar o ambiente inteligente junto com a Internet das Coisas, provendo meios para que os fabricantes de dispositivos exponham suas APIs REST para que comunicações fora do ambiente Manna-X possam ser realizadas. Isso contribui não somente para a maneira de construir

ambientes inteligentes, mas pelo fato de conseguir integrar dispositivos na Internet das Coisas que não possuíam meios de comunicação com a rede IP e nem um servidor Web internamente.

As experiências obtidas pelo ambiente criado para teste mostraram-se úteis no que se refere à possibilidade da criação de tarefas em grupo. Essa característica só pode ocorrer graças ao framework que viabilizou meios para que os dispositivos fossem integrados num mesmo sistema e assim tarefas em grupos criadas.

Os teste realizados na seção 5.1 mostraram que o framework se comporta como o esperado, pois seus tempos aferidos e seus uso de CPU e memória são aceitaveis para aplicações em AI. Pode-se se notar que o tempo de troca de mensagens não tem relação com o número de Drivers instalados, porém o uso da CPU e memória sim.

As dificuldades encontradas ao longo da pesquisa foram na parte de implementação do framework e de seus dispositivos. Outro problema encontrado foi a implementação do OSGi que por enquanto ainda no Brasil é pouco difundida. Existem poucos materiais didáticos disponíveis para o público, porém a tecnologia faz valer o esforço.

O escopo do arcabouço desenvolvido é somente auxiliar o desenvolvedor de ambientes e os fabricantes de dispositivos a se unirem, facilitando o trabalho de ambas as partes viabilizando meios para isso. Contudo, é de total responsabilidade do desenvolvedor de ambientes como ele implementará sua solução. O framework é somente uma ferramenta de auxílio, e não busca resolver questões de alto nível como, a inteligência do ambiente, somente fornece os meios.

Como trabalhos futuros muito ainda pode-se pesquisar, a criação de um framework que abranja até a parte de inteligência ambiente não está descartada. Outra possibilidade é a expansão do framework para uma arquitetura descentralizada, porém pesquisas devem ser feitas, para saber sua viabilidade e custo benefício. Outra ideia para trabalhos futuros é a integração desses ambientes desenvolvidos com o framework Manna-X com outras áreas maiores como: Cidades Inteligentes, Estados Inteligentes, Redes Veiculares Inteligentes, e dentre tantas outras áreas inovadores que estão por vir.

Referências

Ahmed, V.; Ladhake, S. Novel ultra low cost remote monitoring system for home automation using cell phone. In: *Computational Intelligence and Communication Networks (CICN), 2011 International Conference on*, 2011, p. 569 –573.

Akyildiz, I.; Su, W.; Sankarasubramaniam, Y.; Cayirci, E. A survey on sensor networks. *Communications Magazine, IEEE*, v. 40, n. 8, p. 102 – 114, 2002.

Alkar, A.; Gecim, H.; Guney, M. Web based zigbee enabled home automation system. In: *Network-Based Information Systems (NBiS), 2010 13th International Conference on*, 2010, p. 290 –296.

Allen, B.; Dillon, B. Environmental controland field bus systems. Dublin, 1997.

Aragues, A.; Martinez, I.; Del Valle, P.; Munoz, P.; Escayola, J.; Trigo, J. Trends in entertainment, home automation and e-health: Toward cross-domain integration. *Communications Magazine, IEEE*, v. 50, n. 6, p. 160 –167, 2012.

Atukorala, K.; Wijekoon, D.; Tharugasini, M.; Perera, I.; Silva, C. Smarteye integrated solution to home automation, security and monitoring through mobile phones. In: *Next Generation Mobile Applications, Services and Technologies, 2009. NGMAST '09. Third International Conference on*, 2009, p. 64 –69.

Augusto, J. C.; Nakashima, H.; Aghajan, H. Ambient intelligence and smart environments: A state of the art. In: Nakashima, H.; Aghajan, H.; Augusto, J. C., eds. *Handbook of Ambient Intelligence and Smart Environments*, Springer US, p. 3–31, 10.1007/978-0-387-93808-0_1, 2010.
Disponível em `http://dx.doi.org/10.1007/978-0-387-93808-0\_1` (Último Acesso em 30 de Novembro de 2012)

Basili, V. R.; Caldiera, G.; Rombach, H. D. *Encyclopedia of software engineering*, cáp. Goal, Question Metric Paradigm John Wiley and Sons, 1994.

Bavafa, M.; Navidi, N. Towards a reference middleware architecture for ambient intelligence systems. In: *Knowledge Engineering, 2010 8th International Conference on ICT and*, 2010, p. 98 –102.

Bittins, B.; Sieck, J.; Herzog, M. Supervision and regulation of home automation systems with smartphones. In: *Computer Modeling and Simulation (EMS), 2010 Fourth UKSim European Symposium on*, 2010, p. 444 –448.

Chandrasekhar, A.; P Kaimal, V.; Bhamare, C.; Khosla, S. Ambient intelligence : The next generation technology. *International Journal on Computer Science and Engineering IJCSE*, v. 3, n. 6, p. 2491–2497, 2011.

Corredor, I.; Martínez, J. F.; Familiar, M. S.; López, L. Knowledge-aware and service-oriented middleware for deploying pervasive services. *J. Netw. Comput. Appl.*, v. 35, n. 2, p. 562–576, 2012.
Disponível em `http://dx.doi.org/10.1016/j.jnca.2011.05.009`

Dey, A. K.; Abowd, G. D. Towards a better understanding of context and context-awareness. In: *In HUC '99: Proceedings of the 1st international symposium on Handheld and Ubiquitous Computing*, Springer-Verlag, 1999, p. 304–307.

Dias, C.; Pizzolato, N. *DomÓtica aplicabilidade e sistemas de automação residencial*, v. 6 Publicação Técnico-Científica do CEFET CAMPOS, p. 9–32, 2004.

yong Duan, P.; Li, H. A learning algorithm of dynamical associational multi-agents for intelligent environments. In: *Intelligent Control and Automation (WCICA), 2010 8th World Congress on*, 2010, p. 2659 –2663.

Ducatel, K.; Bogdanowicz, M.; Scapolo, F.; Leijten, J.; Burgelman, J.-C. Scenarios for ambient intelligence in 2010. In: *IST Advisory Group Final Report, European Commission, EC. Brussels*, 2001.

Ducatel, K.; Bogdanowicz, M.; Scapolo, F.; Leijten, J.; Burgelman, J.-C. Ambient intelligence : From vision to reality. In: *IST Advisory Group Draft Report, European Commission*, 2003.

Duquennoy, S.; Grimaud, G.; Vandewalle, J.-J. The web of things: Interconnecting devices with high usability and performance. In: *Embedded Software and Systems, 2009. ICESS '09. International Conference on*, 2009, p. 323 –330.

EIBA *Eib handbook series release 3.* Version 1.0., Twinhouse, Bruxelles, 1999.

FALUDI, R. *Building wireless sensor networks - A practical guide to the zigbee mesh networking protocol*. O'Reilly, I–XVIII, 1–300 p., 2011.
Disponível em `http://www.oreilly.de/catalog/9780596807733/index.html` (Último Acesso em 30 de Novembro de 2012)

FIELDING, R. T. *Architectural styles and the design of network-based software architectures*. Doctoral dissertation, University of California, Irvine, 2000.
Disponível em `http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm` (Último Acesso em 30 de Novembro de 2012)

FUENTES, L.; JIMÉNEZ, D. An aspect-oriented ambient intelligence middleware platform. In: *Proceedings of the 3rd international workshop on Middleware for pervasive and ad-hoc computing*, MPAC '05, New York, NY, USA: ACM, 2005, p. 1–8 (*MPAC '05*,).
Disponível em `http://doi.acm.org/10.1145/1101480.1101482` (Último Acesso em 30 de Novembro de 2012)

GÁMEZ, N.; FUENTES, L. Famiware: a family of event-based middleware for ambient intelligence. *Personal Ubiquitous Comput.*, v. 15, n. 4, p. 329–339, 2011.
Disponível em `http://dx.doi.org/10.1007/s00779-010-0354-0` (Último Acesso em 30 de Novembro de 2012)

GARCIA, A.; OLIVER, J.; GOSCH, D. An intelligent agent-based distributed architecture for smart-grid integrated network management. In: *Local Computer Networks (LCN), 2010 IEEE 35th Conference on*, 2010, p. 1013 –1018.

GESSINGER, A. K.; HENNIG, C. H. Zigbee : Conectividade wireless para automação e controle. *Congresso Internacional de Automação ISASHOW, São Paulo*, 2005.

GILL, K.; YANG, S.-H.; YAO, F.; LU, X. A zigbee-based home automation system. *Consumer Electronics, IEEE Transactions on*, v. 55, n. 2, p. 422 –430, 2009.

GOES, S. C. B. *Uma mini estação meterológica*. Relatório técnico de bolsa de iniciação científica, Universidade Estadual de Maringá, 2011.

GOMEZ, C.; PARADELLS, J. Wireless home automation networks: A survey of architectures and technologies. *Communications Magazine, IEEE*, v. 48, n. 6, p. 92 –101, 2010.

GUINARD, D. Towards opportunistic applications in a web of things. In: *Pervasive Computing and Communications Workshops (PERCOM Workshops), 2010 8th IEEE International Conference on*, 2010, p. 863 –864.

GUINARD, D.; ION, I.; MAYER, S. In search of an internet of things service architecture: Rest or ws-*? a developers' perspective. In: *Proceedings of Mobiquitous 2011 (8th International ICST Conference on Mobile and Ubiquitous Systems)*, Copenhagen, Denmark, 2011, p. 326–337.

GUINARD, D.; TRIFA, V.; WILDE, E. A resource oriented architecture for the web of things. In: *Internet of Things (IOT), 2010*, 2010, p. 1 –8.

HA, Y.-G.; SOHN, J.-C.; CHO, Y.-J. ubihome: An infrastructure for ubiquitous home network services. In: *Consumer Electronics, 2007. ISCE 2007. IEEE International Symposium on*, 2007, p. 1 –6.

HAN, J.; YUN, J.; JANG, J.; PARK, K.-R. User-friendly home automation based on 3d virtual world. *Consumer Electronics, IEEE Transactions on*, v. 56, n. 3, p. 1843 –1847, 2010.

IQBAL, M.; LIM, H. B.; NG, T. J. Ecosense: A context and semantics driven framework for eco-aware ambient environments. In: *Consumer Communications and Networking Conference (CCNC), 2010 7th IEEE*, 2010, p. 1 –5.

JARVINEN, H.; LITVINOV, A.; VUORIMAA, P. Integration platform for home and building automation systems. In: *Consumer Communications and Networking Conference (CCNC), 2011 IEEE*, 2011, p. 292 –296.

KAMEAS, A. Towards the next generation of ambient intelligent environments. In: *Enabling Technologies: Infrastructures for Collaborative Enterprises (WETICE), 2010 19th IEEE International Workshop on*, 2010, p. 1 –6.

KIM, S.-J.; SEO, H.-M.; PARK, W.-C.; KIM, S.-D. Network bridge system for interoperation of zigbee-upnp network. In: *Intelligent Networks and Intelligent Systems (ICINIS), 2011 4th International Conference on*, 2011, p. 125 –128.

KIM, T.; KIM, D.; PARK, N.; EUN YOO, S.; LOPEZ, T. Shortcut tree routing in zigbee networks. In: *Wireless Pervasive Computing, 2007. ISWPC '07. 2nd International Symposium on*, 2007.

KINECT, M. Página web oficial do microsoft kinect. 2012.
Disponível em `http://www.microsoft.com/en-us/kinectforwindows/develop/learn.aspx` (Último Acesso em 30 de Novembro de 2012)

KORTUEM, G.; KAWSAR, F.; FITTON, D.; SUNDRAMOORTHY, V. Smart objects as building blocks for the internet of things. *Internet Computing, IEEE*, v. 14, n. 1, p. 44 –51, 2010.

KOVATSCH, M.; WEISS, M.; GUINARD, D. Embedding internet technology for home automation. In: *Emerging Technologies and Factory Automation (ETFA), 2010 IEEE Conference on*, 2010, p. 1 –8.

LEE, K. Y.; CHOI, J. W. Remote-controlled home automation system via bluetooth home network. In: *SICE 2003 Annual Conference*, 2003, p. 2824 –2829 Vol.3.

LEHMANN, G.; RIEGER, A.; BLUMENDORF, M.; ALBAYRAK, S. A 3-layer architecture for smart environment models. In: *Pervasive Computing and Communications Workshops (PERCOM Workshops), 2010 8th IEEE International Conference on*, 2010, p. 636 –641.

LI, H.; WANG, J. Application architecture for ambient intelligence systems based on context ontology modeling. In: *Internet Technology and Applications (iTAP), 2011 International Conference on*, 2011, p. 1 –4.

LIBELUM. Descrição sobre a empresa. 2012.
Disponível em `http://www.libelium.com/company/about` (Último Acesso em 30 de Novembro de 2012)

MARKWALTER, B.; RUSSELL, C. Consumer electronics bus, a robust communications system. In: *Consumer Electronics, 1988. Digest of Technical Papers. ICCE., IEEE 1988 International Conference on*, 1988, p. 42 –43.

MAYER, S.; GUINARD, D. An extensible discovery service for smart things. In: *Proceedings of the Second International Workshop on Web of Things*, WoT '11, New York, NY, USA: ACM, 2011, p. 7:1–7:6 (*WoT '11*,).
Disponível em `http://doi.acm.org/10.1145/1993966.1993976` (Último Acesso em 30 de Novembro de 2012)

MESSIAS, A. R. Controle remoto e aquisição de dados via xbee/zigbee (ieee 802.15.4). 2012.

Disponível em http://www.rogercom.com/ZigBee/ZigBee.htm (Último Acesso em 30 de Novembro de 2012)

Mo, T.; Li, W.; Chu, W.; Wu, Z. An event driven model for context-aware service. In: *Web Services (ICWS), 2011 IEEE International Conference on*, 2011, p. 740 –741.

Moritz, G.; Zeeb, E.; Pruandter, S.; Golatowski, F.; Timmermann, D.; Stoll, R. Devices profile for web services and the rest. In: *Industrial Informatics (INDIN), 2010 8th IEEE International Conference on*, 2010, p. 584 –591.

Mrazovac, B.; Bjelica, M.; Papp, I.; Teslic, N. Smart audio/video playback control based on presence detection and user localization in home environment. In: *Engineering of Computer Based Systems (ECBS-EERC), 2011 2nd Eastern European Regional Conference on the*, 2011, p. 44 –53.

Muratori, J. R.; Bó, P. H. D. *Automação residencial: histórico, definições e conceitos*, cáp. Capítulo I. 62 ed Revista eletrônica: O setor elétrico, p. 70–77, 2011.
Disponível em http://www.osetoreletrico.com.br/web/documentos/fasciculos/Ed62_fasc_automacao_capI.pdf (Último Acesso em 30 de Novembro de 2012)

Norrie, L.; Murray-Smith, R. Virtual sensors: rapid prototyping of ubiquitous interaction with a mobile phone and a kinect. In: *Proceedings of the 13th International Conference on Human Computer Interaction with Mobile Devices and Services*, MobileHCI '11, New York, NY, USA: ACM, 2011, p. 25–28 (*MobileHCI '11*,).
Disponível em http://doi.acm.org/10.1145/2037373.2037378 (Último Acesso em 30 de Novembro de 2012)

NSF Report of the national science foundation workshop on fundamental research in networking. report to the nsf foundation directorate for computer, information science, and engineering. 2003.
Disponível em http://www.cs.virginia.edu/jorg/workshop (Último Acesso em Janeiro de 2004)

O'Driscoll, G. *The essential guide to home networking.* Upper Saddle River, NJ, USA: Prentice Hall PTR, 2000.

Outeirino, F.; Aranda, P.; de la Cruz Fernandez, J.; Jarauta, B. Universal bluetooth access control and security system for e-keys enviroments. In: *Emerging Security Information Systems and Technologies (SECURWARE), 2010 Fourth International Conference on*, 2010, p. 247 –250.

Park, S. O.; Kim, J. S.; Kim, S. J. An object-based virtual network in unbiquitous computing environment. In: *Multimedia and Ubiquitous Engineering (MUE), 2011 5th FTRA International Conference on*, 2011, p. 18 –21.

Perumal, T.; Ramli, A. R.; Leong, C. Y.; Samsudin, K.; Mansor, S. Middleware for heterogeneous subsystems interoperability in intelligent buildings. *Automation in Construction*, v. 19, n. 2, p. 160 – 168, 2010.
Disponível em `http://www.sciencedirect.com/science/article/pii/S0926580509001848` (Último Acesso em 30 de Novembro de 2012)

Peters, M.; Brink, C.; Sachweh, S. Domain independent architecture and behavior modeling for pervasive computing environments. In: *Complex, Intelligent and Software Intensive Systems (CISIS), 2012 Sixth International Conference on*, 2012, p. 327 –334.

Piyare, R.; Tazil, M. Bluetooth based home automation system using cell phone. In: *Consumer Electronics (ISCE), 2011 IEEE 15th International Symposium on*, 2011, p. 192 –195.

Poslad, S. *Ubiquitous computing: Smart devices, environments and interactions*, v. 1. Wiley, 473 p., 2009.
Disponível em `http://books.google.com/books?id=knfGIltq86kC` (Último Acesso em 30 de Novembro de 2012)

Raiten, S. Kinect-getting started-become the incredible hulk. 2011.
Disponível em `http://blogs.microsoft.co.il/blogs/shair/archive/2011/06/17/kinect-getting-started-become-the-incredible-hulk.aspx` (Último Acesso em 30 de Novembro de 2012)

Ramparany, F.; Benazzouz, Y.; Martin, M. Agenda driven home automation - towards high level context aware systems. In: *Ubiquitous, Autonomic and Trusted Computing, 2009. UIC-ATC '09. Symposia and Workshops on*, 2009, p. 125 –130.

Reinisch, C.; Kofler, M.; Kastner, W. Thinkhome: A smart home as digital ecosystem. In: *Digital Ecosystems and Technologies (DEST), 2010 4th IEEE International Conference on*, 2010, p. 256 –261.

Römer, K.; Kasten, O.; Mattern, F. Middleware challenges for wireless sensor networks. *Mobile Computing and Communications Review*, v. 6, p. 2002, 2002.

Roalter, L.; Kranz, M.; Möller, A. A middleware for intelligent environments and the internet of things. In: *Proceedings of the 7th international conference on Ubiquitous intelligence and computing*, UIC'10, Berlin, Heidelberg: Springer-Verlag, 2010, p. 267–281 (*UIC'10*,).
Disponível em `http://dl.acm.org/citation.cfm?id=1929661.1929690` (Último Acesso em 30 de Novembro de 2012)

Romero, D.; Hermosillo, G.; Taherkordi, A.; Nzekwa, R.; Rouvoy, R.; Eliassen, F. Restful integration of heterogeneous devices in pervasive environments. In: *Proceedings of the 10th IFIP WG 6.1 international conference on Distributed Applications and Interoperable Systems*, DAIS'10, Berlin, Heidelberg: Springer-Verlag, 2010, p. 1–14 (*DAIS'10*,).
Disponível em `http://dx.doi.org/10.1007/978-3-642-13645-0_1` (Último Acesso em 30 de Novembro de 2012)

Rosi, A.; Mamei, M.; Zambonelli, F.; Dobson, S.; Stevenson, G.; Ye, J. Social sensors and pervasive services: Approaches and perspectives. In: *Pervasive Computing and Communications Workshops (PERCOM Workshops), 2011 IEEE International Conference on*, 2011, p. 525 –530.

Rui, C.; Yi-bin, H.; Zhang-qin, H.; Jian, H. Modeling the ambient intelligence application system: Concept, software, data, and network. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, v. 39, n. 3, p. 299 –314, 2009.

Ruiz, L.; Loureiro, A. A. F.; da Costa, J. C.; Pozza, R.; Goncalves, P. C. Engenharia de computação invisível. jornada de atualização em informática. congresso da sociedade brasileira de computação. 2011.

Ruiz, L.; Nogueira, J.; Loureiro, A. Manna: a management architecture for wireless sensor networks. *Communications Magazine, IEEE*, v. 41, n. 2, p. 116 – 125, 2003.

Ruiz, L. B. *Manna: Uma arquitetura para gerenciamento de redes de sensores sem fio*. Tese de Doutoramento, Departamento de Ciência da Computação da Universidade Federal de Minas Gerais, 2003.

Sadri, F. Ambient intelligence: A survey. *ACM Comput. Surv.*, v. 43, n. 4, p. 36:1–36:66, 2011.

Disponível em http://doi.acm.org/10.1145/1978802.1978815 (Último Acesso em 30 de Novembro de 2012)

Sampaio, D.; Reis, L.; Rodrigues, R. A survey on ambient intelligence projects. In: *Information Systems and Technologies (CISTI), 2012 7th Iberian Conference on*, 2012, p. 1 –6.

Shelby, Z. Embedded web services. *Wireless Communications, IEEE*, v. 17, n. 6, p. 52 –57, 2010.

Silva, F. A.; de Moura Braga, T. R.; Ruiz, L. B.; Loureiro, A. A. F. Internal contexts inference system for ubiquitous context-aware applications. In: *Proceedings of the 12th International Conference on Information Integration and Web-based Applications & Services*, iiWAS '10, New York, NY, USA: ACM, 2010, p. 776–779 (*iiWAS '10*,).
Disponível em http://doi.acm.org/10.1145/1967486.1967612 (Último Acesso em 30 de Novembro de 2012)

Silva, T. R. d. M. B. *Tratamento de conflitos coletivos em sistemas ubíquos cientes de contexto*. Tese de Doutoramento, Universidade Federal de Minas Gerais, 2010.

Sommaruga, L.; Formilli, T.; Rizzo, N. Domoml: an integrating devices framework for ambient intelligence solutions. In: *Proceedings of the 6th International Workshop on Enhanced Web Service Technologies*, WEWST '11, New York, NY, USA: ACM, 2011, p. 9–15 (*WEWST '11*,).
Disponível em http://doi.acm.org/10.1145/2031325.2031327 (Último Acesso em 30 de Novembro de 2012)

Sriskanthan, N. Bluetooth based home automation system. *Microprocessors and Microsystems*, v. 26, n. 6, p. 281–289, 2002.
Disponível em http://linkinghub.elsevier.com/retrieve/pii/S014193310200039X (Último Acesso em 30 de Novembro de 2012)

Stirbu, V. Towards a restful plug and play experience in the web of things. In: *Semantic Computing, 2008 IEEE International Conference on*, 2008, p. 512 –517.

Tanz, J. Kinect hackers are changing the future of robotics. 2011.
Disponível em http://www.wired.com/magazine/2011/06/mf_kinect/all/1 (Último Acesso em 30 de Novembro de 2012)

VENKATESH, V.; VAITHYANATHAN, V.; KUMAR, M.; RAJ, P. A secure ambient assisted living (aal) environment: An implementation view. In: *Computer Communication and Informatics (ICCCI), 2012 International Conference on*, 2012, p. 1 –7.

VERSLYPE, D.; NELIS, J.; VERSCHUEREN, T.; HAERICK, W.; DE TURCK, F.; DEVELDER, C. Framework for ubiquitous discovery and access to home services. In: *Future Computing, Service Computation, Cognitive, Adaptive, Content, Patterns, 2009. COMPUTATIONWORLD '09. Computation World:*, 2009, p. 398 –403.

WEISER, M. The computer for the 21st century. *Scientific American*, v. 3, n. 3, p. 3–11, 1991.
Disponível em `http://wiki.daimi.au.dk/pca/_files/weiser-orig.pdf` (Último Acesso em 30 de Novembro de 2012)

WILHELM, M. A generic context aware gesture recognition framework for smart environments. In: *Pervasive Computing and Communications Workshops (PERCOM Workshops), 2012 IEEE International Conference on*, 2012, p. 536 –537.

WILLIG, A. Recent and emerging topics in wireless industrial communications: A selection. *Industrial Informatics, IEEE Transactions on*, v. 4, n. 2, p. 102 –124, 2008.

WONG, E. A phone-based remote controller for home and office automation. *Consumer Electronics, IEEE Transactions on*, v. 40, n. 1, p. 28 –34, 1994.

XU, T.; DAVID, B.; CHALON, R.; ZHOU, Y. A context-aware middleware for ambient intelligence. In: *Proceedings of the Workshop on Posters and Demos Track*, PDT '11, New York, NY, USA: ACM, 2011, p. 10:1–10:2 (*PDT '11*,).
Disponível em `http://doi.acm.org/10.1145/2088960.2088970` (Último Acesso em 30 de Novembro de 2012)

YIQIN, L.; FANG, F.; WEI, L. Home networking and control based on upnp: An implementation. In: *Computer Science and Engineering, 2009. WCSE '09. Second International Workshop on*, 2009, p. 385 –389.

Apêndice A

A.1 Gráficos de Desempenho do Manna-X para 5 nós

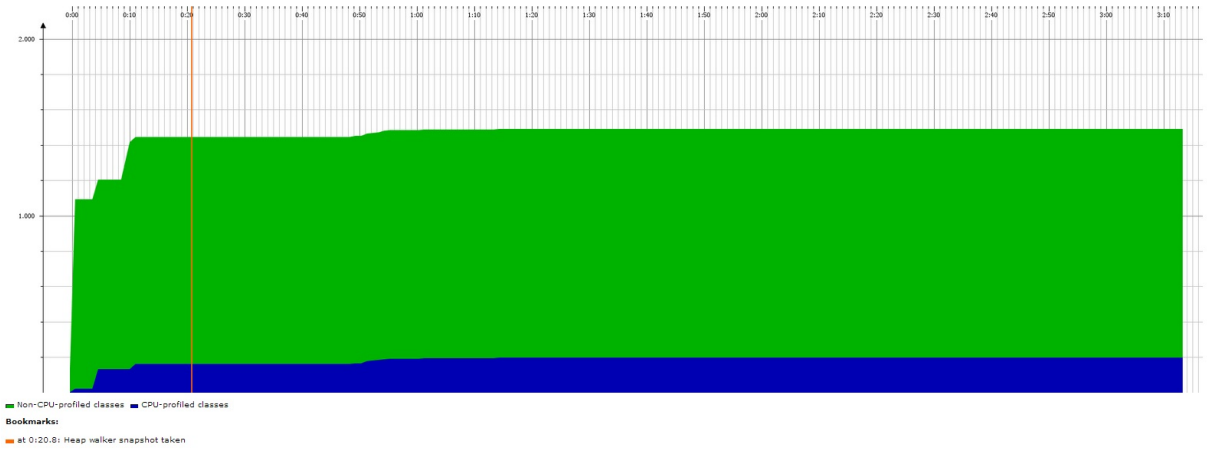

Figura 1.1: Gráfico de Classes para 5 nós

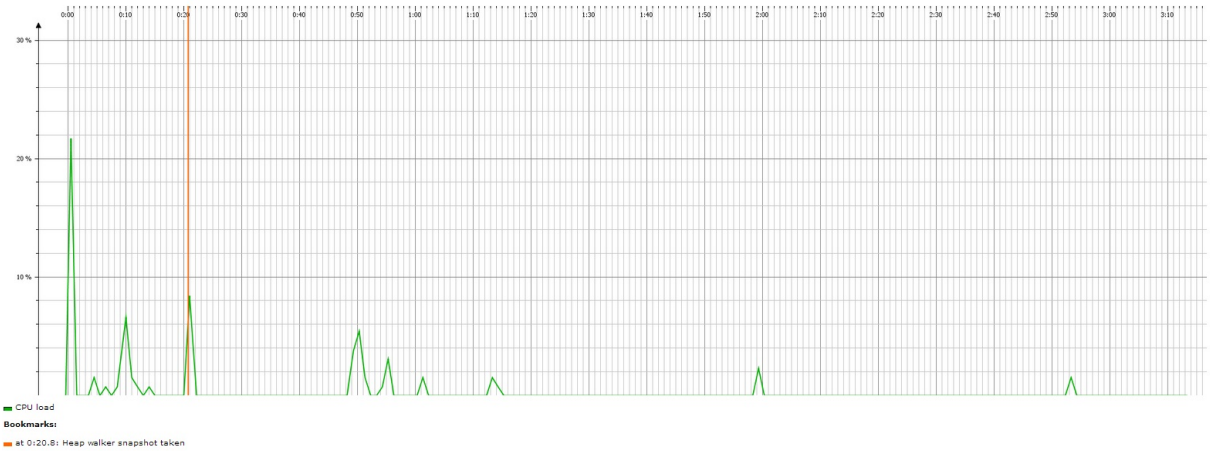

Figura 1.2: Gráfico de uso da CPU para 5 nós

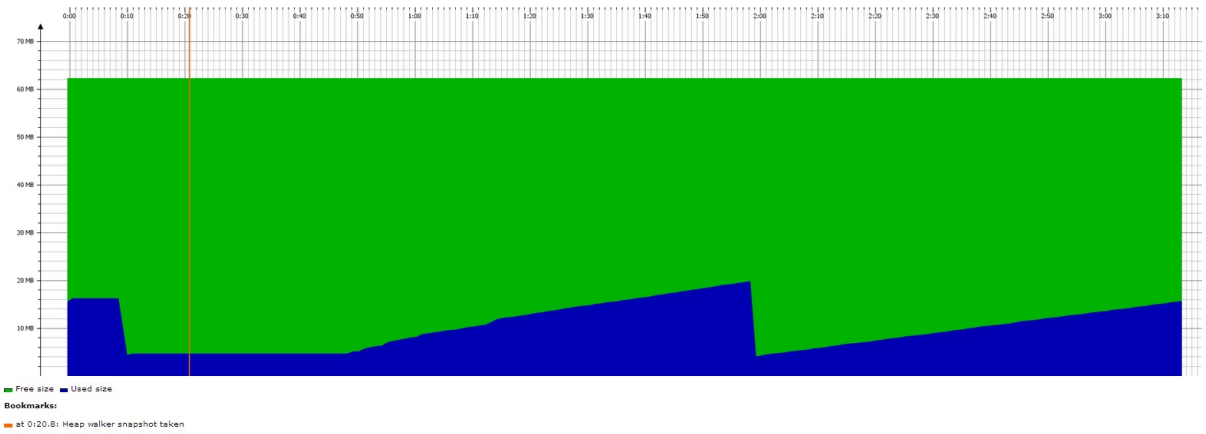

Figura 1.3: Gráfico de uso de Memória Principal para 5 nós

A.2 Gráficos de Desempenho do Manna-X para 10 nós

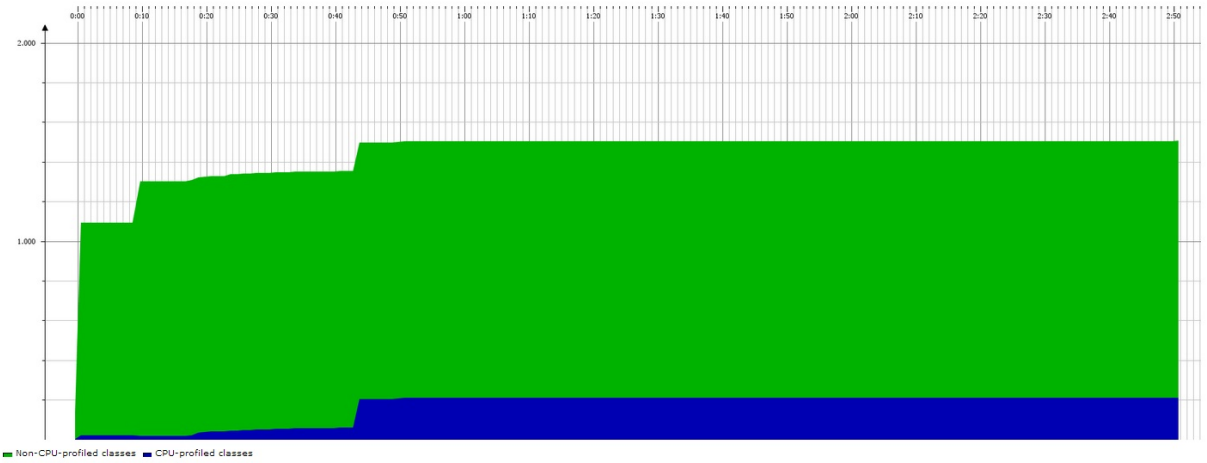

Figura 1.4: Gráfico de Classes para 10 nós

Figura 1.5: Gráfico de uso da CPU para 10 nós

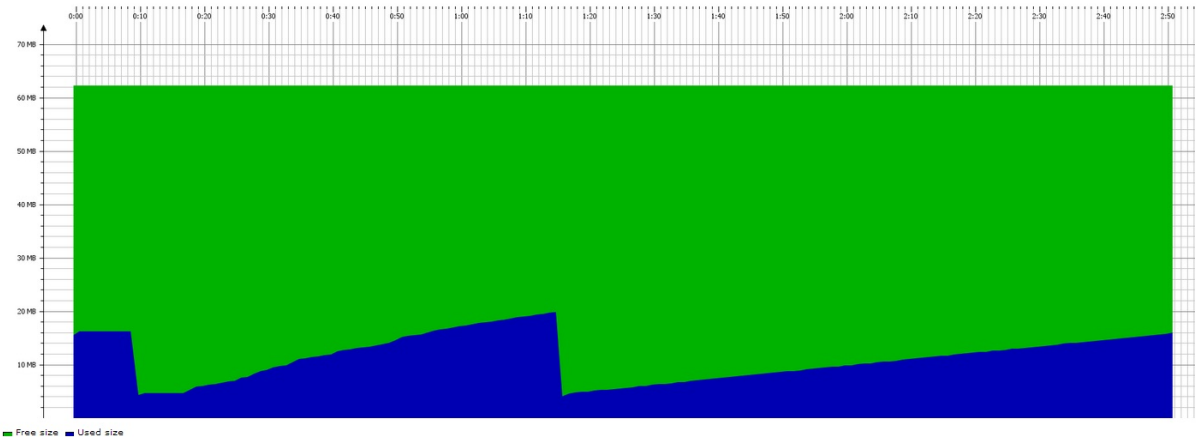

Figura 1.6: Gráfico de uso de Memória Principal para 10 nós

A.3 Gráficos de Desempenho do Manna-X para 50 nós

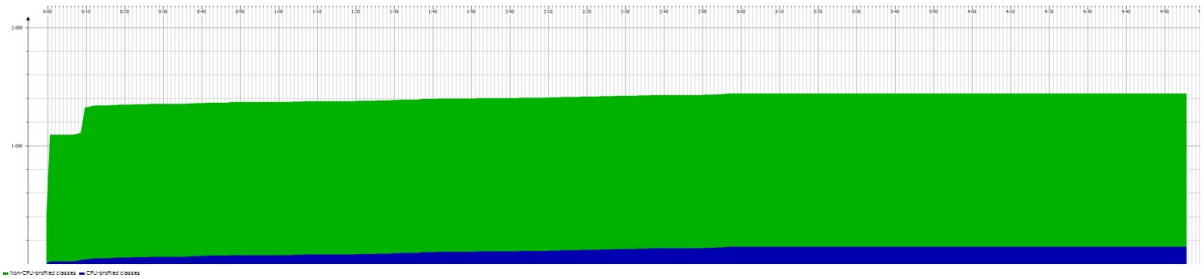

Figura 1.7: Gráfico de Classes para 50 nós

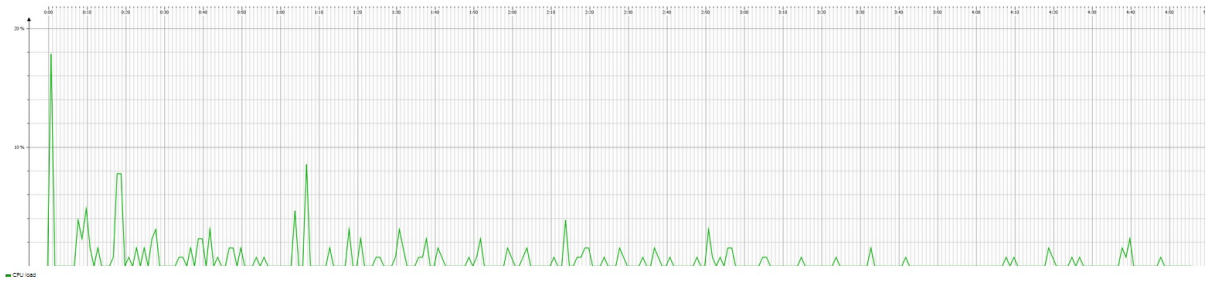

Figura 1.8: Gráfico de uso da CPU para 50 nós

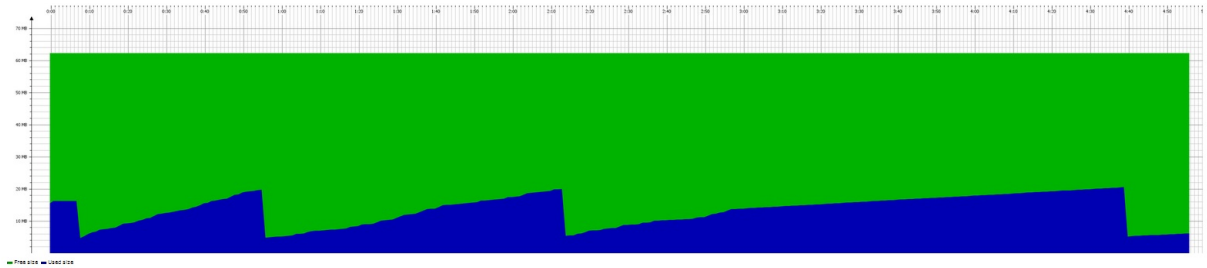

Figura 1.9: Gráfico de uso de Memória Principal para 50 nós

A.4 Gráficos de Desempenho do Manna-X para 100 nós

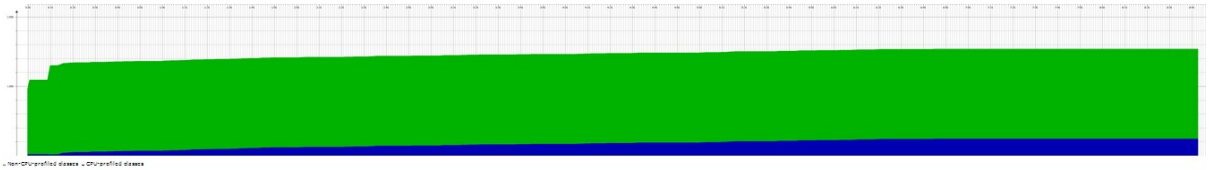

Figura 1.10: Gráfico de Classes para 100 nós

Figura 1.11: Gráfico de uso da CPU para 100 nós

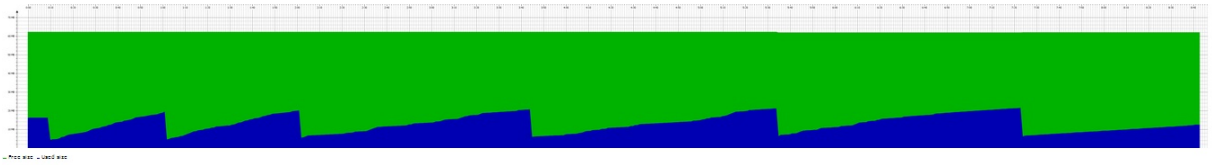

Figura 1.12: Gráfico de uso de Memória Principal para 100 nós

A.5 Gráficos de Desempenho do Manna-X para 1000 nós

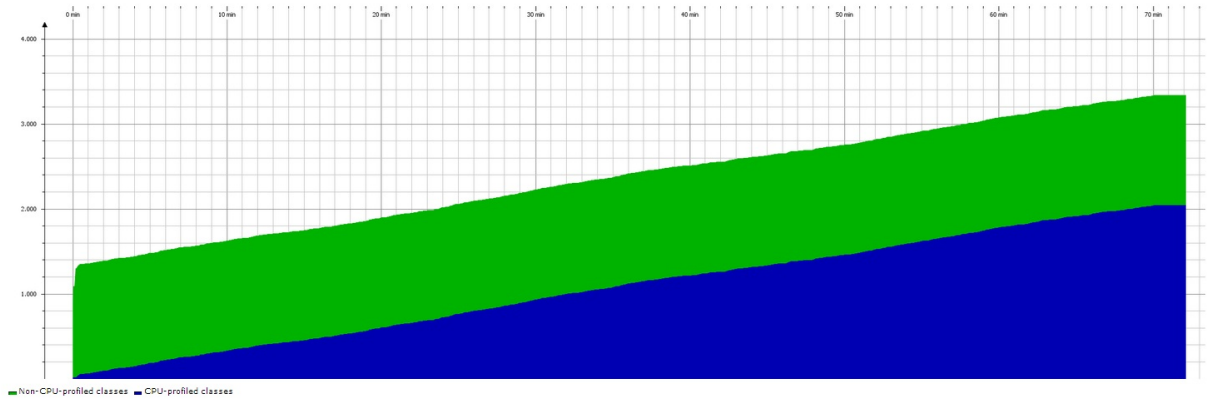

Figura 1.13: Gráfico de Classes para 1000 nós

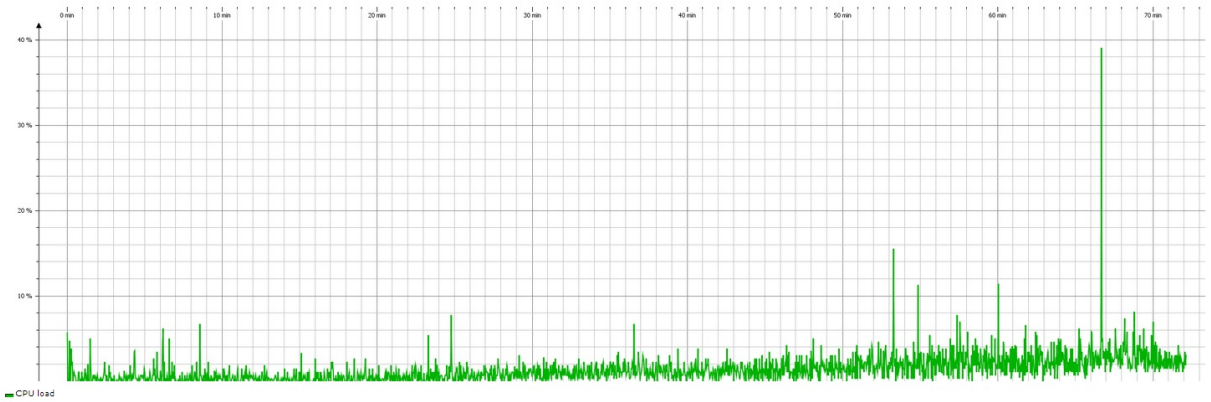

Figura 1.14: Gráfico de uso da CPU para 1000 nós

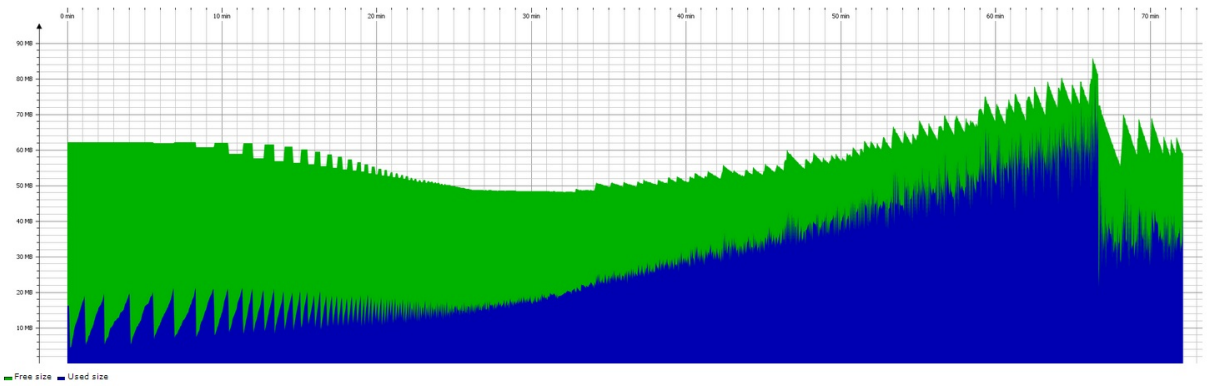

Figura 1.15: Gráfico de uso de Memória Principal para 1000 nós