Universidade Federal de Campina Grande
Centro de Engenharia Elétrica e Informática
Coordenação de Pós-Graduação em Ciência da Computação

Um Arcabouço para o Desenvolvimento de Aplicações para Monitoramento Pervasivo de Ambientes

Romeryto Vieira Lira

Dissertação submetida à Coordenação do Curso de Pós-Graduação em Ciência da Computação da Universidade Federal de Campina Grande - Campus I como parte dos requisitos necessários para obtenção do grau de Mestre em Ciência da Computação.

Área de Concentração: Ciência da Computação
Linha de Pesquisa: Engenharia de Software

Orientadores:
Angelo Perkusich
Hyggo Oliveira de Almeida

Campina Grande, Paraíba, Brasil

FICHA CATALOGRÁFICA ELABORADA PELA BIBLIOTECA CENTRAL DA UFCG

L768a Lira, Romeryto Vieira.

Um arcabouço para o desenvolvimento de aplicações para monitoramento pervasivo de ambientes / Romeryto Vieira Lira. – 2013.

71 f. : il. color.

Dissertação (Mestrado em Ciência da Computação) – Universidade Federal de Campina Grande, Centro de Engenharia Elétrica e Informática.

"Orientação: Prof. Dr. Angelo Perkusich, Prof. Dr. Hyggo Oliveira de Almeida".

Referências.

1. Arcabouço de Software. 2. Monitoramento Pervasivo. 3. Engenharia de Software. 4. Computação Pervasiva. 5. Computação Ubíqua. I. Perkusich, Angelo. II. Almeida, Hyggo Oliveira de. III. Título.

CDU 004.415.2.01(043)

"UM ARCABOUÇO PARA O DESENVOLVIMENTO DE APLICAÇÕES PARA MONITORAMENTO PERVASIVO DE AMBIENTES"

ROMERYTO VIEIRA LIRA

DISSERTAÇÃO APROVADA EM 27/08/2013

HYGGO OLIVEIRA DE ALMEIDA, D.Sc, UFCG
Orientador(a)

ANGELO PERKUSICH, D.Sc, UFCG
Orientador(a)

MARCOS RICARDO ALCÂNTARA MORAIS, D.Sc, UFCG
Examinador(a)

KYLLER COSTA GORGÔNIO, Dr., UFCG
Examinador(a)

CAMPINA GRANDE - PB

Resumo

Aplicações para monitoramento pervasivo de ambientes que são dotados de infraestruturas implantadas são cada vez mais necessárias em vários domínios diferentes, como na agricultura, redes elétricas, fábricas industriais, redes ferroviárias, sistemas de transporte, entre outros.

Para o bom funcionamento dos equipamentos existentes nestes ambientes é importante efetuar monitoramento e consequentes ações de manutenção, de modo a evitar danos e prejuízos.

Considerando estes fatores, pode-se verificar que mesmo para domínios distintos, estas aplicações têm características arquiteturais comuns que em muitos casos não são aproveitadas, provocando assim o retrabalho e a recriação de várias funcionalidades similares que poderiam ser reusadas a cada aplicação desenvolvida em domínio diferente.

Neste trabalho, é apresentado um Arcabouço para o desenvolvimento de aplicações para monitoramento pervasivo de ambientes que engloba tais características arquiteturais e funcionalidades similares, oferecendo ao desenvolvedor ferramentas de desenvolvimento que tornam mais simples a construção deste tipo de aplicações, independente do domínio em questão.

Palavras Chave: Arcabouço de Software, Monitoramento Pervasivo, Engenharia de Software, Computação Pervasiva, Computação Ubíqua

Abstract

Applications for pervasive monitoring of enviroments that have physically deployed infrastructures are increasingly necessary in several areas, including agriculture, electrical networks, industrial plants, rail networks, transportation systems, and others.

For the proper functioning of the existing equipment in these environments, it is important to perform monitoring and subsequent maintenance actions, in order to prevent damage and losses.

Considering these factors, we can verify that even for different domains, those applications have common architectural features that are not exploited, thus causing rework, and recreating several identical functionalities that could be reused for each application developed in different domains.

This work presents a Software Framework to the develop applications for pervasive monitoring of environments encompassing such architectural features, providing development tools to the developer to make simpler to build this type of applications, regardless of the domain in question.

Keywords: Software Framework, Pervasive Monitoring, Software Engineering, Pervasive Computing, Ubiquitous Computing

Agradecimentos

A Deus, o grande arquiteto do universo que nunca nos deixa desamparados.

Aos meus pais Manoel Lira de Sousa e Maria Vieira Mendes Lira, por todo o empenho e apoio, principalmente nesses 7 anos que estou longe de casa e lutando por uma vida melhor. Agradeço todo o amor incondicional dado a mim como filho.

Aos meus irmãos José Rômulo e Silvia por todo o incentivo, apoio e fraternidade.

Aos amigos de mestrado, Maurílio Silva, Lorena Maia e Antonio Dias, por compartilharem bons momentos de trabalho e amizade durante este tempo na pós-graduação.

Aos meus amigos de graduação Arinaldo Medeiros, Heverton Stuart, Abraão Morais, Cayo Mesquita, Gustavo Jansen, Cláudia Miriany, George Alves e Vicente Matias pelo companherismo, descontração e boas conversas.

Aos amigos do laboratório e da Signove: Marcos Fábio, Ivo Augusto, Diego Bezerra, André Gomes, Felipe Pontes, Ádrian Lívio, Mateus Máximo, Giovanni Calheiros, Kyller Gorgônio e Glauber Ferreira, pelo apoio e o aprendizado.

A Ademar Netto, Geronilson Almeida, Marília Reül, Cleyton Souza, Diego Charles, Lívia Almeida e Gilson Barbosa pela força e apoio.

Aos meus professores de graduação, em especial, ao Prof. Hyggo, Prof. Camilo Medeiros(*In Memoriam*), Prof(a). Eliane Araújo, Prof. Elmar e Prof. Carlos Eduardo por terem aberto portas e oportunidades de iniciação ao ensino e à pesquisa que ajudaram muito no meu amadurecimento acadêmico.

Aos meus orientadores de mestrado, Prof. Hyggo e Prof. Ângelo, pela credibilidade, pela participação essencial neste trabalho, pelas oportunidades e grandes ensinamentos, pela paciência, auxílio e idéias valiosas, não somente durante o mestrado, mas durante todos estes 4 anos que faço parte da família *Embedded*.

Às secretárias da COPIN, Vera Oliveira e Rebeka Lemos pela presteza, profissionalismo e atenção oferecidas sempre que as procurei.

A CAPES, HP e Compal pelo apoio financeiro durante o período da pesquisa.

Enfim, a todos que contribuíram de forma direta ou indireta para a realização deste trabalho.

Conteúdo

Lista de Figuras

Lista de Tabelas

Lista de Códigos Fonte

Capítulo 1

Introdução

Dentre as novas tendências e campos de estudo que surgiram na Ciência da Computação desde os anos 90, destaca-se a Computação Pervasiva. Tal conceito surgiu da visão *Weiser* [46]. Tanto na indústria quanto na academia há diversos trabalhos na área de computação pervasiva e serviços pervasivos [49]. Com o objetivo de tornar os serviços computacionais mais simples através de ambientes que se adaptam e interagem com as pessoas de forma transparente e onipresente, a Computação Pervasiva surgiu quebrando o paradigma existente na época, o qual era caracterizado por sistemas computacionais que eram apenas programas, dispositivos ou ferramentas que liam arquivos e executavam programas, mas não se integravam ou interagiam diretamente com os usuários em si [30]. Dentro desse campo destacam-se os serviços e aplicações pervasivas que têm por objetivo fornecer aos usuários informações na hora certa, tempo certo e de forma satisfatória para assim atender necessidades humanas específicas. Tais serviços, através dos seus dados coletados, podem auxiliar os usuários a tomar decisões, sem necessariamente o usuário conhecer ou ver o serviço que lhe está provendo tais informações, ou seja, tais serviços são transparentes para quem os usa [39].

Dentro desta abordagem, aplicações que têm grande destaque são aquelas voltadas para o monitoramento pervasivo de ambientes e consequente disparo de ações de manutenção baseadas em informações de contexto. Nesses ambientes se destaca o monitoramento através do uso de sensores, sendo este ramo um dos mais importantes da Computação Pervasiva [48].

Aplicações para o monitoramento pervasivo são cada vez mais necessárias para diminuir os gastos com manutenção de equipamentos ou controle de ambientes. Vendo a questão financeira, para muitas empresas, um equipamento ou estrutura com problema, mesmo que

por poucas horas, pode resultar em perdas monetárias consideráveis, seja no domínio das companhias de energia elétrica, indústrias, empresas de agronegócio, empresas de geração de energia renovável, sistemas de transporte, entre outros [36]. Tendo em vista este enfoque, o monitoramento pervasivo é uma solução viável, visto que provê uma forma onipresente de monitoramento, procurando minimizar as ocorrências de erro humano e alertando de forma transparente às pessoas ou sistemas responsáveis em agir para problemas que porventura tenham ocorrido nas estruturas ou ambientes monitorados.

Para coletar dados de monitoramento, geralmente usam-se sensores que são acoplados ao equipamento ou ambiente a ser monitorado. Estes dados monitorados fornecem importantes informações, como medições de temperatura, umidade, intensidade sonora, valor de corrente elétrica passante, entre outras, que são coletadas periodicamente com o objetivo de detectar situações anômalas de forma rápida, podendo assim, efetuar ações de manutenção/correção de modo a evitar ou minimizar eventuais prejuízos que tais anomalias ou problemas possam causar [2, 8].

Embora existam aplicações para o monitoramento pervasivo de diversos tipos e em diversos domínios, para cada aplicação desenvolvida existe um trabalho, um esforço, custo e uma complexidade acentuada na sua construção, visto que não é trivial desenvolver diversas funcionalidades como coleta de dados, gerenciamento de regras de disparo de alertas para consequentes ações de manutenção baseadas nas medições recebidas, para cada aplicação de monitoramento pervasiva a ser criada.

Considerando estes fatores, pode-se verificar que mesmo para domínios distintos, estas aplicações têm características arquiteturais comuns que não são aproveitadas, provocando assim o retrabalho e a recriação de várias funcionalidades idênticas que poderiam ser reusadas a cada aplicação desenvolvida em domínio diferente.

Neste trabalho, apresenta-se um Arcabouço para o desenvolvimento de aplicações para monitoramento pervasivo de ambientes que engloba as características arquiteturais comuns, oferecendo ao desenvolvedor um conjunto de ferramentas de desenvolvimento que tornam mais simples a construção deste tipo de aplicações, escondendo a complexidade que envolve a criação de aplicações deste tipo, não precisando se preocupar com as particularidades de cada domínio em questão.

1.1 Problemática

O monitoramento e ações de manutenção em ambientes que têm *infraestruturas implantadas* são atividades cada vez mais necessárias. Neste trabalho entenda-se *infraestruturas implantadas* como um ambiente que contém uma ou mais entidades (objetos, equipamentos, grãos, etc) que possam ser guardadas, armazenadas ou até mesmo instaladas. É necessário que elas, na parte maior do tempo, estejam em perfeito estado, bom funcionamento e em condições normais. No entanto muitas dessas infraestruturas não têm manutenções periódicas ou o número de ações de manutenção é limitado devido a restrições financeiras. Isso ocorre devido aos procedimentos de manutenção serem caros dependendo do equipamento inspecionado/consertado. Se a *Manutenção Preventiva* é cara, a *Manutenção Corretiva* pode ser mais cara ainda devido aos custos financeiros associados a estres procedimentos. Fazer previsão e prevenção de problemas nessas infraestruturas ou ambientes também é caro, porém corrigir é bem mais custoso [36]. Um dos domínios de infraestruturas implantadas em que existe tal problema são os ambientes industriais, caracterizados pelo maquinário industrial que constantemente precisa de manutenção [12, 22, 38, 40].

Além do domínio das indústrias, pode-se verificar que vários outros domínios de infraestrutura implantada necessitam de *Manutenção Preventiva* e estão expostos ao problema citado.

No domínio da rede de Energia elétrica a *Manutenção Preventiva* e o reparo de qualidade são os pilares para evitar interrupções nos serviços da rede [26]. É importante prever ou detectar problemas nos equipamentos que compõem a infraestrutura da rede, visto que uma queda de energia pode ter impacto direto na vida humana, quando afeta um hospital por exemplo, e em outros casos o impacto é principalmente econômico.

Liu [29] reconhece que as estratégias de manutenção periódicas tradicionais são muito caras, e que a *Manutenção Preventiva* juntamente com o uso da tecnologia da informação estão mais aptas para detecção e previsão de falhas em equipamentos e aptas para gerar ações de manutenção no tempo certo. Ele ressalta ainda que tal problema ocorre fortemente nos ambientes de usinas hidroelétricas, confirmando assim que os aparelhos e maquinário existentes nestas carecem de uma logística de manutenção mais barata e simples.

Já em outro domínio, *Hau* [17] afirma que os custos de manutenção representam uma

porção substancial dos gastos relacionados aos sistemas de geração de energia eólica. Dado que nestes sistemas o equipamento fundamental para o funcionamento são as turbinas eólicas, em que a confiabilidade, gerenciamento e manutenção têm atraído grande interesse, objetivando reduzir custos com operações de manutenção [31].

Com o objetivo de prover manutenção das turbinas eólicas, considerando inclusive seus componentes individuais como rolamentos, engrenagens, gerador etc, se o procedimento for efetuado continuamente, os custos operacionais e financeiros podem ser diminuídos, visto que alertas de manutenção poderiam ser disparados de forma mais efetiva e rápida, diminuindo o risco dos equipamentos terem danos mais sérios [42] .

Já no domínio do sistema ferroviário, tanto os próprios trilhos quanto os vagões (tanto em trens quanto em metrôs) necessitam de manutenção preventiva devido ao desgaste que acontece no decorrer do tempo, gerando problemas como insegurança e o alto custo de manutenção corretiva envolvido [5]. Com base nessa visão, é necessário assegurar que todos os componentes devem funcionar corretamente e que possíveis falhas futuras podem ser detectadas com antecedência. Tal domínio necessita que ações de manutenção como tarefas de inspeção, substituição de componentes e lubrificação sejam feitos de forma rápida e com antecedência de modo a garantir a segurança dos usuários do serviço ferroviário [16]. Tais problemas podem ser detectados por sensores acústicos, visto que tais ocorrências podem ser descobertas a partir da intensidade ou característica do som produzido pelo atrito das rodas metálicas dos vagões com os trilhos [41].

Na agricultura, equipamentos, manutenção e reparo representam uma grande parcela dos custos e despesas [45]. Neste domínio dentre as principais ferramentas de trabalho e maquinário agrícola destaca-se o trator. Tratores necessitam de manutenção preventiva para manter o bom funcionamento do equipamento minimizando custos de manutenção [24]. Problemas nesses veículos podem ser detectados por sensores que monitoram a qualidade e nível do óleo dos motores, a vibração dos componentes das carrocerias, a velocidade do torque do motor e também o consumo de combustível, mantendo assim o bom estado da frota [1, 23].

Também relacionado ao campo da agricultura têm-se os depósitos de grãos. Tais instalações, bem como *containers* e cilos que armazenam tais alimentos, necessitam de manutenção e inspeção. Tal necessidade ocorre devido ao fato de que os níveis de umidade e a temperatura às quais estes estão expostos são os fatores primordiais que podem causar deterioração

das sementes. Tais fatores podem ser monitorados a partir de sensores com o objetivo de lançar alertas para ações de manutenção [20, 21]. Também neste contexto, os trabalhos mais recentes afirmam que o não monitoramento das condições do local de armazenamento podem ocasionar a perda de propriedades nutricionais destas sementes, aliados à contaminação por fungos, microorganismos e por insetos [11, 28].

Com base no estudo da problemática nestes domínios do monitoramento pervasivo de ambientes, observou-se que desenvolver sistemas para monitoramento pervasivo de ambientes segue uma arquitetura padrão. A arquitetura é ilustrada na Figura 1.1. Com base neste padrão arquitetural, na listagem a seguir é explicado cada componente do mesmo.

- **Os mecanismos de aquisição de informação.** São sensores de diversos tipos, usados para obter-se medições da entidade que está sendo monitorada;
  Exemplos: Sensor Acústico, sensor de temperatura, sensor de umidade etc.

- **As Regras de disparo de alertas para ações de manutenção/inspeção.** São regras que determinam quando um alerta de manutenção/inspeção será enviado aos atuadores;
  Exemplo: Quando a temperatura de um depósito de grãos estiver abaixo de 15^{o}C ou acima de de 30^o, deve-se enviar alerta para a pessoa responsável, para tomada de providências objetivando manter a qualidade das sementes;

- **Alertas de ação de manutenção.** Alertas que são disparados assim que o problema for detectado;

- **Os Atuadores.** Pessoas, equipamentos, ou procedimentos automatizados que vão agir sobre o problema detectado.

Considerando estes fatores, pode-se verificar que mesmo para domínios distintos, estas aplicações têm características arquiteturais comuns que não são aproveitadas, provocando assim o retrabalho e a recriação de várias funcionalidades idênticas que poderiam ser reusadas a cada aplicação desenvolvida em domínio diferente. Neste trabalho, é apresentado um Arcabouço para o desenvolvimento de aplicações para monitoramento pervasivo de ambientes que engloba tais características arquiteturais, oferecendo ao desenvolvedor ferramentas

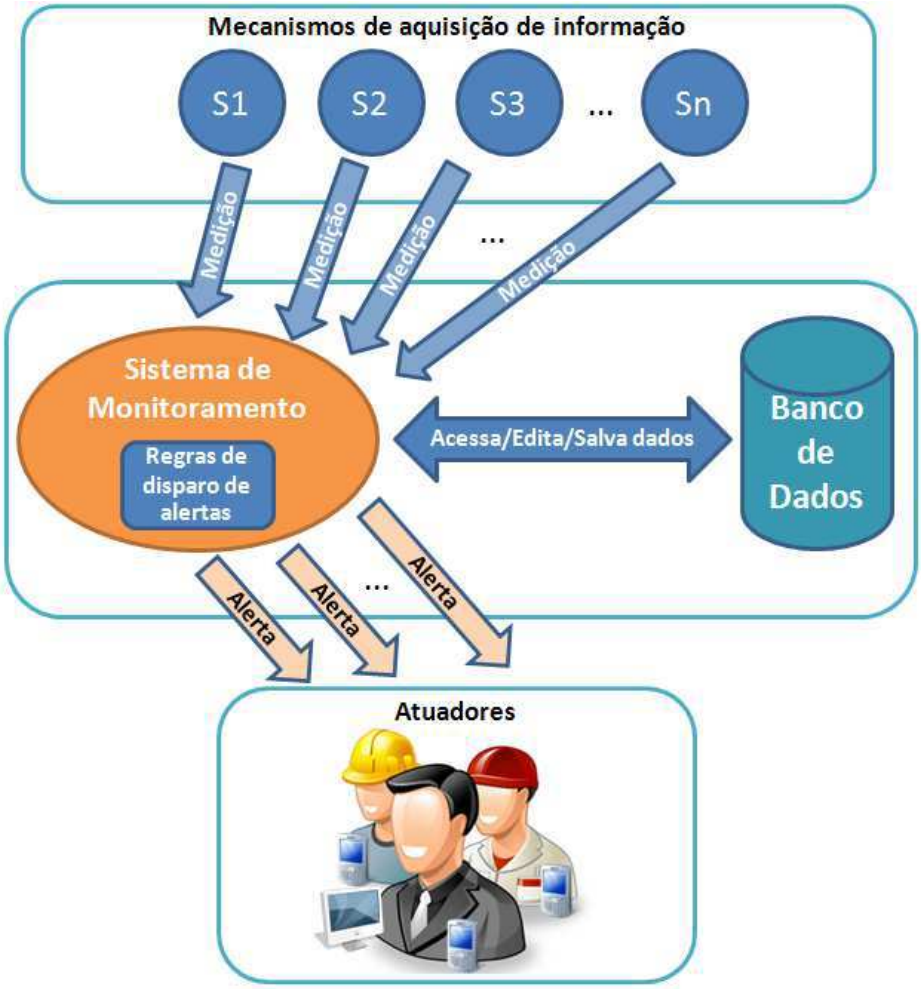


Figura 1.1: Arquitetura Padrão de Aplicação de Monitoramento Pervasivo de Ambientes.

de desenvolvimento que tornam mais simples a construção deste tipo de aplicações, independente do domínio em questão.

1.2 Objetivos

Neste trabalho tem-se como objetivo desenvolver um Arcabouço de software para o desenvolvimento de aplicações para o monitoramento pervasivo de ambientes, independente de domínio. Tal Arcabouço esconde a complexidade na construção de aplicações, possibilitando que o desenvolvedor possa implementar de forma mais simples e menos burocrática suas aplicações de monitoramento, diminuindo tempo, evitando retrabalho, esforço e recursos na concepção de novas aplicações.

Para alcançar o objetivo principal, os seguintes objetivos específicos são considerados:

1. Definição e implementação do Arcabouço, aproveitando as características arquiteturais comuns das aplicações de monitoramento pervasivo;

2. Definição de diretrizes a serem seguidas por desenvolvedores para utilização do Arcabouço;

3. Desenvolvimento de três aplicações em domínios diferentes usando o Arcabouço.

1.3 Relevância

Em ambientes onde existem infraestruturas implantadas, problemas que comprometem o bom funcionamento destas podem trazer grandes prejuízos. Isso acontece porque os gastos com *Manutenção Corretiva*, ou meios de monitoramento manuais através de verificação humana, são altos [36]. Com base nisto uma das principais contribuições planejadas deste trabalho é que através do Arcabouço proposto é possível o desenvolvimento aplicações para o monitoramento pervasivo de ambientes que são capazes de disparar ações de manutenção no tempo certo (baseado em regras previamente definidas), possibilitando que os atuadores resolvam o problema o quanto antes possivel.

Tem-se também que dentre as contribuições objetivadas pelo Arcabouço, muitas vem da necessidade de que a construção de aplicações seja simples [32] e rápida [9]. Este é responsável por esconder a complexidade na criação de aplicações para o monitoramento pervasivo de ambientes de modo que o desenvolvedor não precisa implementar a lógica de negócio relativa ao armazenamento de medições, gerenciamento e interpretação e validação das regras de ações de manutenção para infraestrutura escolhida, nem se preocupe com o desenvolvimento das funcionalidades relativas ao envio de alertas, visto que o Arcabouço provê estas funcionalidades independente do domínio em questão.

Por conseguinte, é relevante destacar a importância de minimizar custos no desenvolvimento de aplicações computacionais, pois em muitas ocasiões tal aspecto é um fator limitante. Um reflexo disso é a existência de vários trabalhos que se dedicam a estimar o custo de desenvolvimento de modo a economizar recursos financeiros e humanos. De fato, procedimentos que têm como objetivo otimizar, agilizar, e diminuir o esforço no desenvolvimento

de aplicações, são fundamentais para diminuir tais gastos [2,8,18,27]. Tal necessidade existe também na área de monitoramento pervasivo de ambientes, visto que para cada domínio todo o trabalho é refeito mesmo nas camadas onde existem características comuns [13]. Portanto, como estes aspectos são contemplados pelo Arcabouço apresentado, o trabalho também contribuirá para a economia de recursos durante o processo de construção das aplicações de monitoramento pervasivo de ambientes, independente de domínio em questão.

1.4 Organização

Essa dissertação está estruturada da seguinte forma:

- No **Capítulo 2** são descritos os conceitos envolvidos no desenvolvimento desse trabalho. Inicialmente são apresentados aspectos da computação pervasiva e suas implicações. Em seguida, são abordados conceitos relativos ao Monitoramento Pervasivo de ambientes descrevendo a parte de sensores e principais estratégias para disparos de ação de manutenção através de alertas para os atuadores.

- No **Capítulo 3** são descritas as abordagens relacionadas com o trabalho proposto nos principais domínios de aplicações para o monitoramento pervasivo de ambientes.

- No **Capítulo 4** é apresentada uma visão geral do Arcabouço, destacando a interação entre os módulos. Neste é apresentado toda a parte de organização arquitetural do trabalho, bem como as diretrizes que o desenvolvedor tem que seguir para desenvolver aplicações para o monitoramento pervasivo de ambientes usando o Arcabouço. Em seguida são abordados aspectos sobre o uso e a extensibilidade do mesmo.

- No **Capítulo 5**, é apresentada a prova de conceito e a validação do Arcabouço através de aplicações desenvolvidas para três domínios diferentes.

- No **Capítulo 6** são apresentadas as considerações finais e os trabalhos futuros, destacando-se as contribuições desse trabalho e suas implicações.

Capítulo 2

Fundamentação teórica

2.1 Computação Pervasiva

Os últimos anos foram marcados por grandes mudanças na área de Ciência da Computação. Neste âmbito destaca-se o surgimento da *Computação Pervasiva*, apresentada inicialmente por *Weiser* [46]. A mesma caracteriza-se pelo fato dos sistemas computacionais deixarem de ser apenas ferramentas ou dispositivos onde se armazenam arquivos e se executam programas, para se integrar efetivamente na vida dos próprios usuários [30].

Soluções que aplicam os conceitos deste paradigma tem como objetivo fazer com que a vida se torne mais simples através de ambientes que sejam sensitivos, adaptativos e que sejam capazes de produzir respostas para atender as necessidades humanas de forma transparente e onipresente. Para suprir tais necessidades surgiu uma nova categoria de aplicação chamada *Aplicação Pervasiva*. Esta é um meio computacional pelo qual o usuário realiza ou é levado a realizar uma tarefa ou ação e não um software escrito apenas para explorar as capacidades de um dispositivo [39].

Tendo em vista este enfoque, as aplicações deste tipo têm como funcionalidade fundamental o fornecimento de informações ou serviços no tempo certo e de forma satisfatória. Neste sentido tem-se que qualquer informação que possa ser usada para caracterizar uma situação de uma determinada entidade é chamada de *Contexto*. Tais informações não são associadas apenas às características pessoais dos usuários, mas sim ligadas também a questões de tempo, meios computacionais usados, características físicas do ambiente circundante ou de objetos, aparelhos ou equipamentos com os quais o usuário interage ou tem uma relação,

seja de necessidade direta ou indireta em sua vida [30].

2.2 Monitoramento Pervasivo de ambientes

Baseadas no enfoque da Computação Pervasiva, anteriormente citado, aparecem com destaque as aplicações para o monitoramento pervasivo de ambientes. Estas caracterizam-se por um conjunto de serviços ou aplicações que de forma onipresente e transparente monitoram ambientes, entidades, estruturas etc. através de sensores, para assim ajudar os usuários destes sistemas tomar a melhor decisão seja no campo de manutenção preventiva ou corretiva [7, 48]. O potencial deste tipo de aplicações está presente em vários domínios, como por exemplo, no domínio do agronegócio através do monitoramento das lavouras e gerenciamento de qualidade de grãos [20, 21, 25], no domínio ferróviário para garantir o perfeito funcionamento de vagões, rodas, rolamentos e trilhos [36, 41], assim como para gerenciar estruturas, motores, aparelhagem no domínio industrial [12, 22, 38, 40], ou no domínio de redes e usinas de energia elétrica e renovável [10, 26, 43], entre outros.

Através de determinados sensores para captura de dados de ambientes ou infraestruturas implantadas nestes, é possível capturar uma ou mais características que podem auxiliar no monitoramento de problemas e assim servir de base para o disparo de alertas de ações de manutenção, caso sejam necessárias [29]. Redes de sensores montadas nestes ambientes podem ajudar aos usuários na tomada de decisão tendo como suporte as informações coletadas e os requisitos de bom funcionamento do ambiente em questão.

O monitoramento pervasivo de ambientes pode ser feito com vários tipos de sensores, como por exemplo, sensores de temperatura, umidade relativa do ar, sensor de intensidade sonora, sensor de corrente elétrica, sensor de distância, sensor de vibração para motores, entre outros. Nas Figuras 2.1(a), 2.1(b) e 2.1(c), são ilustrados alguns destes sensores.

Nas aplicações para o monitoramento pervasivo de ambientes, antes de disparar um alerta na ocorrência de alguma situação anômala, é preciso saber como detectar tal situação. Para os diversos domínios existentes, existem infraestruturas distintas a serem monitoradas através de sensores. Cada sensor tem suas características em comum, representadas principalmente pelo tipo de dados que ele gera, *e.g.* número decimal, como também pela unidade que é medida, *e.g.* "^{o}C". Para detectar situações anômalas ou problemas que possam ocorrer nas

(a) Sensor de Temperatura (b) Sensor de Corrente (c) Sensor de Som

Figura 2.1: Exemplos de sensores em domínios diferentes

entidades monitoradas, tais sistemas de monitoramento definem regras [47]. Como exemplo, para um sensor de temperatura e umidade instalado num cilo de grãos, pode-se elaborar uma regra que diz o seguinte: *Caso a temperatura seja maior que 35^{o}C ou a umidade relativa do ar seja maior que 70%, enviar um alerta para quem for responsável verificar a situação, pois os grãos podem estragar ou perder qualidade caso o clima no cilo não seja controlado* [20, 21]. Tais alertas podem ser feitos de várias maneiras, mas as que mais se destacam são:

- **Verificação Humana.** A verificação humana neste caso acontece quando uma pessoa tem que ir no sensor (caso exista), ver a medição e ela mesmo avisar a alguém (enviar um alerta) ou ela mesmo corrigir o problema.

- **SMS (*Short Message Service*)** [19]. O SMS é parte integral da tecnologia GSM (*Global System for Mobile Communications*). São mensagens que podem ser enviadas e/ou recebidas por dispositivos tais como telefones celulares. Aplicações para o monitoramento pervasivo de ambientes podem usar este tipo de tecnologia para mandar mensagens de alerta para consequentes ações de manutenção.

- **Email** [14,33]. Aplicações para o monitoramento pervasivo de ambiente podem enviar email para o atuador responsável, caso algum problema ocorra na infraestrutura ou ambiente monitorado.

- ***Web Service*** [15]. Quando algum problema ocorrer, a ação de manutenção pode ser um procedimento realizado por uma rotina computacional que está num servidor web. No exemplo do reservatório de grãos, quando a regra for quebrada, pode ser programada a chamada a um método de um *Web Service* que ajusta as condições ambientes através de um sistema atuador local que liga um arcondicionado para controlar a temperatura e umidade relativa do ar, para que desta forma os grãos não estraguem.

Capítulo 3

Trabalhos relacionados

Embora existam várias maneiras de efetuar-se o monitoramento envolvendo sensores juntamente com procedimentos de verificação de problemas feitos apenas por pessoas, existe um crescente interesse e muitas novas aplicações que tornam tal monitoramento transparente e onipresente para quem precisa saber o *status* da estrutura ou ambiente monitorado. Em outras palavras, a pessoa não precisa estar verificando os dados oriundos dos sensores, dado que existe um sistema que faz este trabalho e pode informar caso algum problema esteja acontencendo na entidade monitorada [48].

É neste foco que caracterizam-se muitas aplicações de monitoramento pervasivo de ambientes em diversos domínios. No domínio ferróviário existe a necessidade de que as linhas de trens possam ser monitoradas para detectar problemas, seja no vagão, ou até mesmo nos trilhos [36, 41]. No campo da agricultura e agronegócio é cada vez mais crescente o interesse em monitorar o estado de conservação dos equipamentos que lidam com a lavoura assim como a safra propriamente dita [25]. No domínio da energia elétrica, sistemas de monitoramento pervasivo podem ser usados para monitorar turbinas, assim como também no campo das energias renováveis, *e.g.* energia eólica e suas extensas usinas [10, 26, 43]. No domínio industrial destacam-se aplicação para o monitoramento de ambientes industriais, caracterizados pelo maquinário industrial que constantemente precisa de manutenção e reparo [12, 22, 38, 40].

Assim pode-se comprovar que o monitoramento pervasivo de ambientes é cada vez mais necessário, dando espaço ao surgimento de várias aplicações em vários outros domínios.

3.1 Discussão sobre sistemas pervasivos através do uso de sensores

Nos anos 2000, *Yoshimi* [48] apresentou uma discussão sobre a criação de sistemas pervasivos a partir do monitoramento através de sensores. Ele sustentava a idéia de que era necessária a existência de uma estrutura através da qual era possível selecionar vários sensores sistematicamente a fim de monitorar e dar suporte ao monitoramento pervasivo de ambientes. A partir disso os autores se forcaram principalmente na parte relacionada aos sensores, vendo problemas e alternativas relacionadas à criação de aplicações para os mesmos.

3.2 Domínio do Transporte Público

Neste trabalho *Chardsutthi* [6] ressalta que o monitoramento e a manutenção de frotas de ônibus é de grande importância e ao mesmo tempo um problema crítico. O autor reforça que a manutenção preventiva é a estratégia mais eficiente para reduzir os custos de reparos e consertos.

Ele provê informações de alertas necessárias para rastrear e agendar ações de manutenção baseados nos dados dos veículos, como consumo diário de combustível, emissão de gás carbônico, consumo de energia, entre outros aspectos incluindo também o gerenciamento de pessoas. Através disto ele desenvolveu um programa de manutenção preventiva baseado num *checklist* para ser implementado pelos técnicos. No trabalho o autor destaca que *"A prevenção é melhor que cura"* também para este domínio do transporte público de passageiros.

3.3 Domínio Ferroviário

Em *Rabatel* [36], é apresentado um sistema de monitoramento através de sensores com foco em manutenção preventiva objetivando a detecção de situações anômalas em equipamentos de linhas de trens. Neste trabalho, inicialmente, foram instalados sensores em locais estratégicos, como motores, rodas, etc. para coletar informações importantes (como temperatura, aceleração, velocidade) para análise e detecção de possíveis anomalias, fornecendo a exata localização geográfica onde o problema ocorreu.

Em seguida desenvolveu-se uma aplicação de monitoramento composta por uma camada de sensores, outra responsável pelo envio de medições, outra pela leitura e tratamento de medições e detecção de situações anômalas, além de outra camada responsável por disparar alertas. Todas estas funcionalidades trabalham em conjunto para prover uma solução que atenda as necessidade do domínio em questão, ou seja monitoramento de equipamentos e estruturas associadas ao campo ferroviário. Porém tudo teve que ser feito praticamente *do zero*, e a solução proposta serve apenas para o domínio corrente.

Já *Thakkar* [41] propôs um sistema de monitoramento de vagões de trens/metrôs e trilhos baseados na intensidade sonora gerada pelo barulho do atrito das rodas dos vagões com os trilhos. O autor trabalha na idéia de que os defeitos podem ser detectados baseados na característica do som produzido. Tal aplicação de monitoramento foi feita, instalando sensores nas rodas dos vagões, considerando a calibração do rolamento, eixos e das juntas.

Na Figura 3.1 é ilustrado o esquema utilizado para instalação dos sensores acústicos. Para identificar problemas, o principal aspecto considerado são os picos atingidos pela intensidade sonora monitorada.

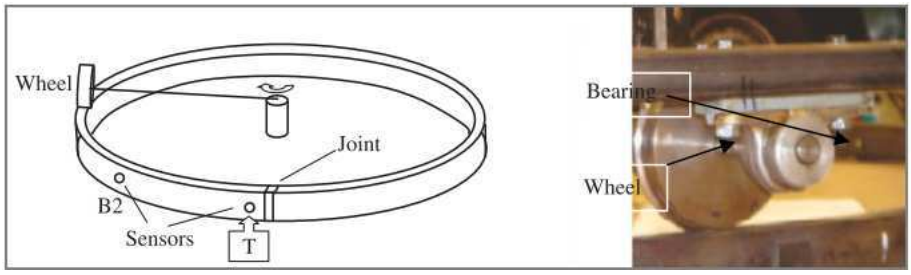


Figura 3.1: Esquema de instalação de sensores acústicos em trilhos de trem [41]

3.4 Domínio da Agricultura e do Agronegócio

Em *Kim* [25], apresenta-se uma aplicação de monitoramento em tempo real de ambientes agrícolas, como plantações, pastos, entre outros. Esta é composta de um eficiente sistema baseado em redes de sensores sem fio composta de robôs autônomos que podem detectar presença de fogo e informações sobre dados atmosféricos a partir dos sensores que eles portam. Para o gerenciamento das informações e consequente envio de mensagens de alerta na ocorrência de alguma situação emergencial, foi desenvolvida uma aplicação baseada em

banco de dados que é responsável por armazenar as informações dos sensores, processar e enviar os alertas.

Com enfoque semelhante, *Jia* [20, 21] destaca que o monitoramento das condições de armazenamento de grãos a fim de garantir a qualidade, segurança e bom estado de conservação é fundamental. Neste trabalho foi apresentada uma aplicação de monitoramento baseado em redes de sensores sem fio. Esta utiliza vários nós/sensores que foram implantados em celeiros, compondo assim uma estrutura de rede autônoma com o objetivo de coletar dados do ambiente, dos cilos e dos grãos armazenados, como temperatura e umidade relativa do ar. Após a coleta, os dados são enviados a uma central de controle. Os sensores, assim como os equipamentos de rede utilizados foram de baixo custo, assim obtendo um bom custo benefício no resultado final.

Na Figura 3.3 é ilustrado um exemplo de sensor de temperatura utilizado. Já na Figura 3.2 pode-se contemplar um esquema de organização dos sensores através dos vários celeiros monitorados.

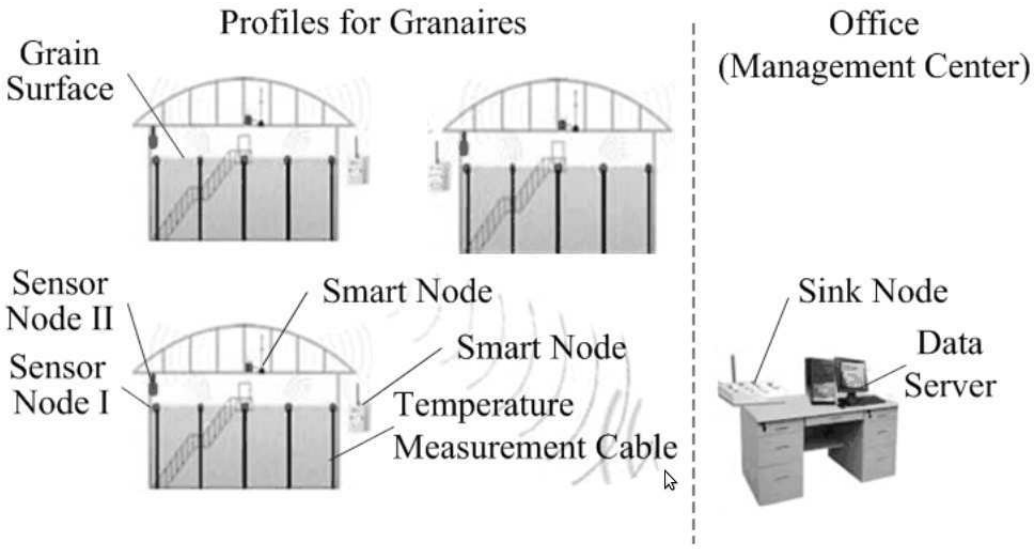


Figura 3.2: Esquema de organização dos sensores através de vários celeiros monitorados [21]

3.5 Domínio da Energia Elétrica

Em *Kume* [26] é apresentada uma aplicação no domínio da energia elétrica. O autor destaca que a manutenção e o reparo de equipamentos elétricos pode ser uma tarefa complicada e ao

Figura 3.3: Exemplo de sensor de temperatura utilizado no monitoramento de celeiros [20]

mesmo tempo muito perigosa. Neste trabalho o autor propõe uma abordagem de monitoramento que envolve os trabalhadores de campo propriamente ditos e os operadores de salas de controle. Para que tal trabalho funcionasse da maneira esperada, além das aplicações móveis para comunicação entre trabalhadores e operadores, o autor apresenta um protótipo de ambiente de comunicação para o monitoramento de equipamentos da rede elétrica que tem por objetivo detectar falhas para que assim ações de manutenção sejam efetuadas da maneira mais rápida possível.

Tal monitoramento é feito através de sensores que coletam informações acerca do equipamento utilizando-se técnicas de processamento de imagem. Tal trabalho reforça a necessidade de aplicações de monitoramento pervasivo no âmbito das redes elétricas, visto que muitos equipamentos necessitam ser monitorados constantemente com a finalidade de detectar falhas assim que elas ocorram, fazendo com que o conserto seja feito o mais breve possível, evitando maiores prejuízos.

3.6 Domínio da Energia Eólica

Em 2011, *Tian* [43] destacou a importância do monitoramento de ambientes e estruturas em usinas produtoras de energia eólica. Tendo isto como base, o autor propôs uma estratégia de manutenção baseada em condições (CBM-*condition based maintenance*) com o objetivo de reduzir os custos de manutenção em sistemas de geração desse tipo de energia. O método CBM proposto lida com informações de vários componentes separados das turbinas eólicas, sejam eles rolamentos, gerador, etc.

Neste mesmo enfoque, *Entezami* [10] desenvolveu uma aplicação de monitoramento do

sistema hidráulico de frenagem em turbinas eólicas baseado em dados de sensores de tensão e corrente elétrica, para detecção de falhas mecânicas nestas estruturas, e consequente tomada de decisão.

Na Figura 3.4, são ilustrados os sensores que capturam dados de tensão e de corrente relacionados ao sistema hidráulico de freio de uma turbina eólica, instalados no painel principal de controle. Já na Figura 3.5 apresenta-se a estrutura de uma turbina eólica e seus componentes.

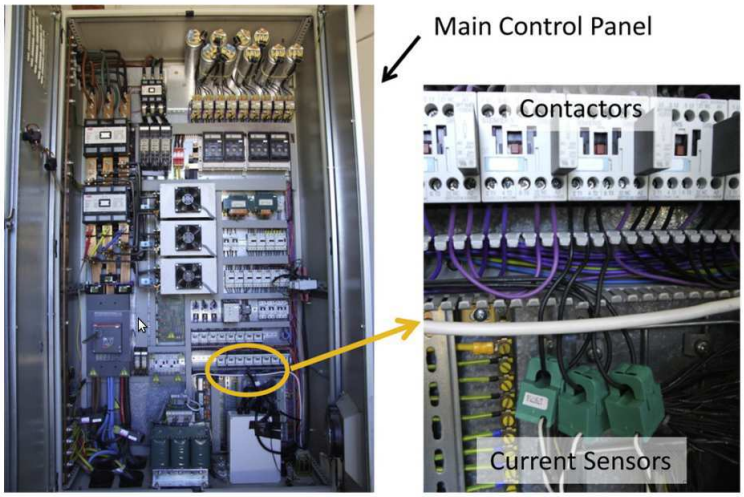


Figura 3.4: Sensores que capturam dados de tensão e de corrente elétrica relacionados ao sistema hidráulico de freio de uma turbina eólica [10]

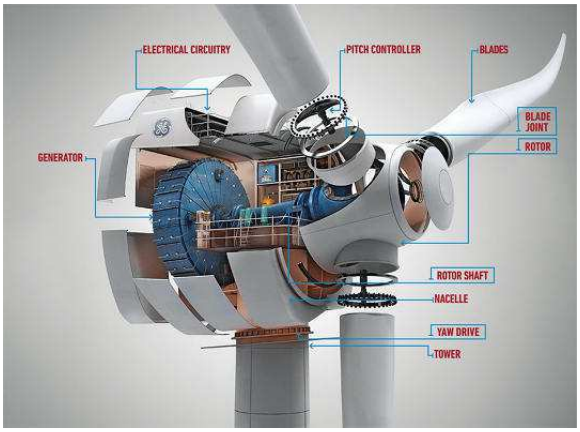


Figura 3.5: Estrutura de uma turbina eólica [35]

3.7 Domínio Industrial

Em 2008 *Rodriguez* [38] propôs um sistema para monitorar as condições de motores de indução. Ele usou um módulo de inteligência utilizando a lógica de *Fuzzi* [50] para obter o real estado atual do motor monitorado. Já *Garcia* [12] propõe um sistema de monitoramento para detectar problemas neste mesmo tipo de motores. Só que neste caso, ele utiliza um sistema de cálculo estatístico que serve de auxílio para o processamento dos dados oriundos dos sensores de temperatura e de vibração. Na Figura 3.6 apresenta-se o motor de indução utilizado por *Garcia* em sua pesquisa.

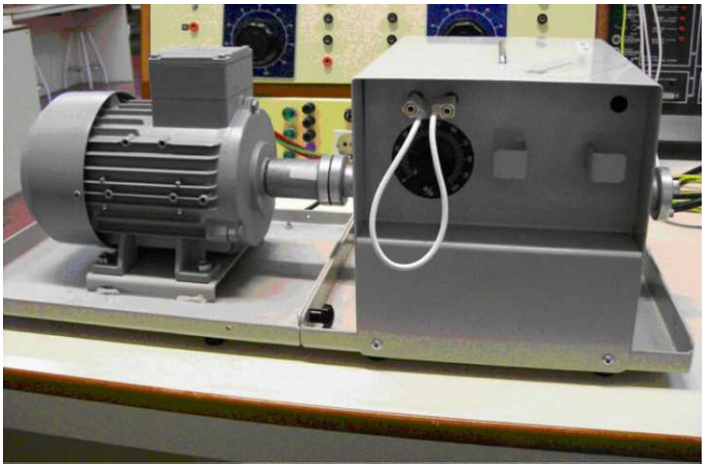


Figura 3.6: Motor de Indução [12]

3.8 Conclusão

Nesta seção foram apresentados os principais trabalhos na área do monitoramento pervasivo de ambientes, considerando domínios relevantes. As soluções propostas nestes trabalhos, embora resolvam total ou parcialmente os problemas para cada domínio em separado, somente se aplicam ao domínio em questão e não dão suporte para outros domínios distintos, ou seja, mesmo os trabalhos tendo características arquiteturais comuns entre si, as funcionalidades semelhantes entre eles tiveram que ser recriadas para cada uma destas aplicações, provocando um retrabalho desnecessário.

A diferença destes trabalhos para o trabalho aqui desenvolvido, é que através do Arcabouço que aqui apresenta-se, o desenvolvedor poderá criar aplicações para monitoramento

pervasivo de ambientes, independente do domínio em questão, visto que as particularidades e funcionalidades semelhantes entre os vários domínios existentes fazem parte da solução, ou seja, o Arcabouço esconde do desenvolvedor toda a complexidade existente para construção de aplicações deste tipo, oferecendo a este desenvolvedor ferramentas de desenvolvimento que tornam o trabalho mais simplificado, independente do domínio em questão.

Capítulo 4

Arcabouço para o desenvolvimento de aplicações para monitoramento pervasivo de ambientes

Neste capítulo, apresenta-se uma visão geral do Arcabouço que possibilita o desenvolvimento de aplicações de monitoramento pervasivo de ambiente. Os elementos vistos na Figura 4.1 são explicados em alto nível, de modo a mostrar o funcionamento, desde a coleta da medição e consequente envio para o Arcabouço, além dos disparos de alertas para ações de manutenção direcionados aos atuadores.

4.1 Visão Geral

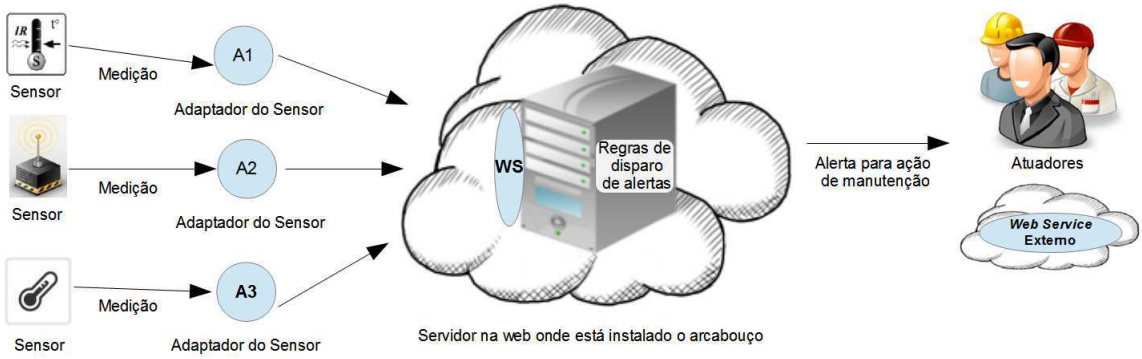


Figura 4.1: Visão Geral do Arcabouço

O Arcabouço é hospedado em um servidor na Internet, e oferece vários serviços e ferramentas através de seu *Web Service*, o qual é detalhado na Seção 4.3.2. Pode-se ver na Figura 4.1 que sensores de tipos diferentes podem ser usados para desenvolver aplicações que monitoram estruturas de domínios diversos.

Cada sensor envia suas medições para o Arcabouço através de um adaptador. Tal adaptador é responsável por conectar-se ao sensor, receber a medição e enviar ao servidor para que o Arcabouço de monitoramento hospedado neste possa processar a medição através das regras de diparo de alertas de manutenção previamente cadastradas para esse sensor.

Caso a medição viole a regra, um alerta é enviado aos atuadores que podem ser pessoas que são responsáveis em verificar a situação e atuar de modo a resolver o problema que possa existir na estrutura monitorada pelo sensor, ou atuadores automatizados, *e.g*, uma chamada a um *Web Service* que executa uma determinada ação previamente programada.

4.2 Arquitetura

Nesta seção, apresenta-se a arquitetura detalhada, tanto do Arcabouço quanto da *Camada de Sensoriamento* e da *Camada de Adaptadores*, mostrando o comportamento e o funcionamento de cada um dos componentes. O Arcabouço está dividido em vários módulos e camadas. A divisão e organização da arquitetura é ilustrada na Figura 4.2.

Inicialmente apresenta-se a *Camada de Sensoriamento* na qual estão presentes domínios do monitoramento pervasivo de ambientes cada um contendo seus respectivos sensores. Esta camada coleta a medição, e a envia para a *Camada Adaptadora* que trata e formata este dado. Logo em seguida, esta camada conecta-se ao Arcabouço e envia o esta medição através de uma requisição *POST* ao *Web Service* usando o método `addMeasurement(String measurement)`.

A partir deste ponto a camada *Receptora de medições* com o seu módulo *Receptor de medições* trata e encaminha a medição ao *Gerenciador de Regras* através do método `addRule(String rule)` da classe `RegisterRuleService` (módulo *Gerenciador de Regras*).

Logo em seguida na classe *InterpreterRuleService*, verifica-se se houve violação de alguma regra cadastrada.

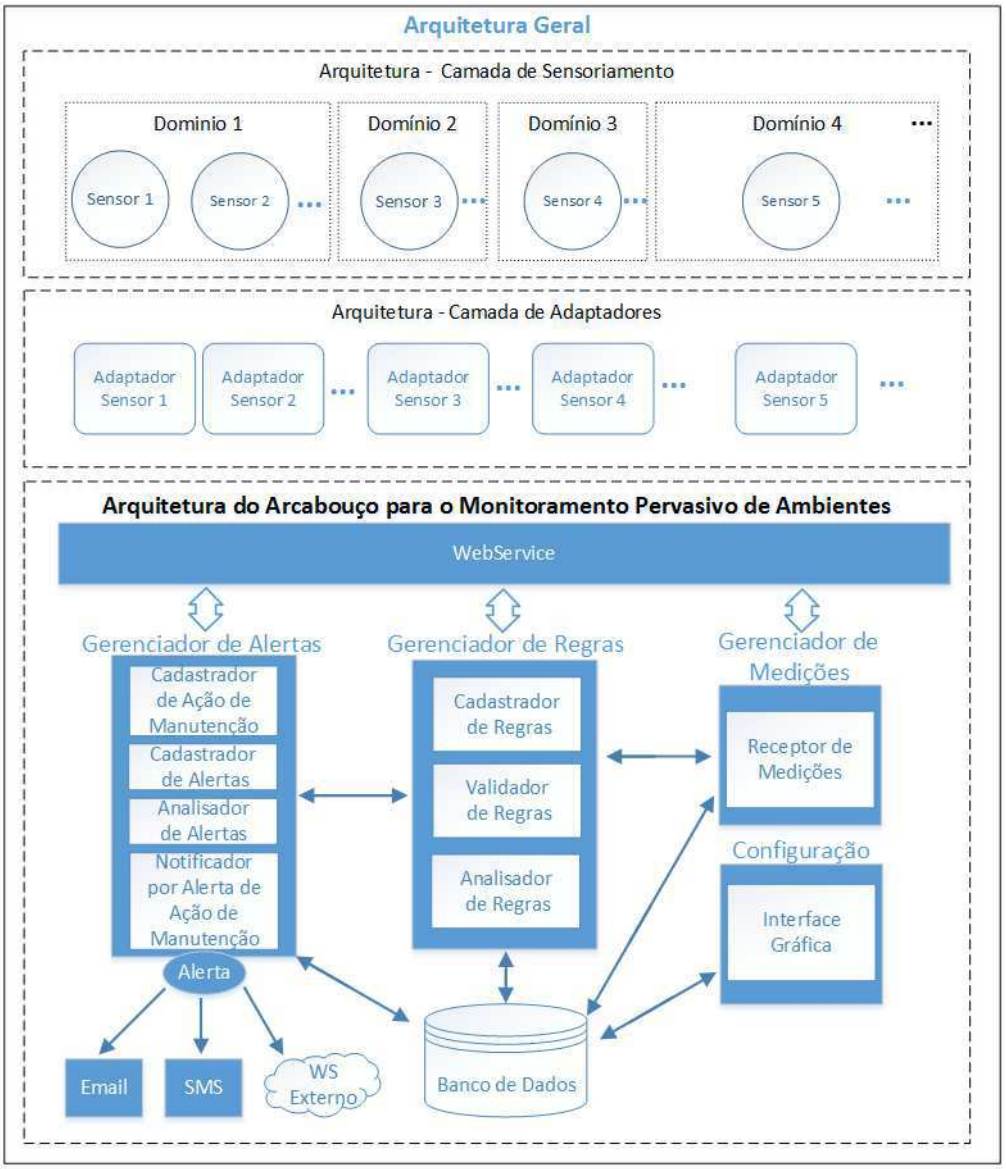


Figura 4.2: Arquitetura do Arcabouço para o desenvolvimento de aplicações para monitoramento pervasivo de ambientes e das camadas de sensoriamento e de adaptadores

Em caso positivo, o módulo *Gerenciador de Alertas* é acionado através de uma das implementações da interface `Notifier` no módulo *Analisador de Alertas*, salva o alerta no Banco de dados (através do módulo *Cadastrador de Alertas*) e dispara o alerta de ação de manutenção para os atuadores humanos, ou então para um *Web Service* externo qualquer que efetue uma ação de manutenção automatizada (através do módulo *Notificador por Alerta de ação de manutenção*). Na Tabela 4.1 é apresentada uma descrição das funcionalidades de cada camada do Arcabouço.

Tabela 4.1: Principais componentes da arquitetura do Arcabouço

Camadas	**Descrição**
Camada de Sensoriamento	Parte do Arcabouço onde estão localizados os diversos sensores e seus domínios respectivos;
Camada Adaptadora	Composto pelos adaptadores para os sensores. São responsáveis por coletar as medições e enviá-las ao Arcabouço;
Web Service	*Web Service* que disponibiliza os métodos a serem usados pelo desenvolvedor para construir sua aplicação de monitoramento a partir do Arcabouço;
Módulo de Configuração	Módulo responsável pela configuração do Arcabouço. Tal configuração pode ser feita via a interface gráfica web do Arcabouço ou através de métodos disponibilizados pelos serviços;
Gerenciador de medições	Responsável por receber dados de medições para depois processá-los e/ou salvá-los;
Gerenciador de Regras	Responsável pelo Cadastro/Intepretação/Validação das regras;
Gerenciador de Alertas	Responsável pelo disparo de alertas de ações de manutenção caso alguma regra seja violada;
Banco de Dados	Responsável pelo armazenamento de todas as informações do Arcabouço.

Para ter-se uma visão mais detalhada do Arcabouço, é ilustrado na Figura 4.3 seu Diagrama de Classes.

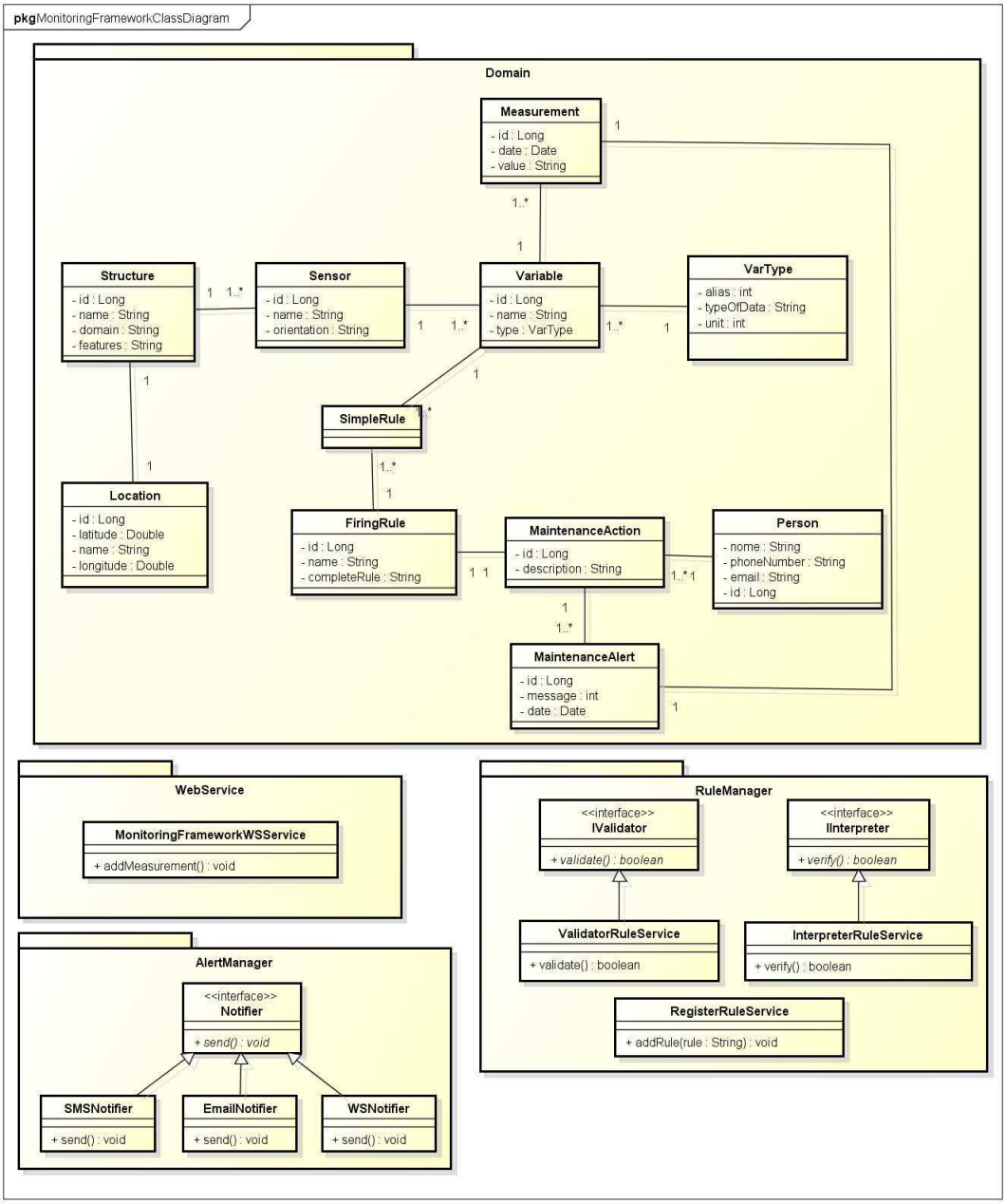


Figura 4.3: Diagrama de Classes Simplificado do Arcabouço

4.3 Módulos

4.3.1 Adaptador do sensor ao Arcabouço

Este componente tem a função de ser a ponte de ligação entre o sensor da *Camada de Sensoriamento* e o Arcabouço.

O adaptador é responsável por conectar-se ao sensor instalado na estrutura monitorada, receber as medições que são enviadas periodicamente por este, organizar os dados e enviá-los para o Arcabouço via *Web Service*. Tal adaptador é um software embarcado em qualquer dispositivo[1] que consiga coletar dados do sensor e que tenha conexão com a internet para conectar-se e usar o Arcabouço.

Vale lembrar que o adaptador não faz parte do Arcabouço, ele apenas é um componente de software que intermedia o envio de medições.

A ideia é que o adaptador seja implementado pelo desenvolvedor ou pelo fabricante que queira fornecer suporte ao uso de seus sensores no Arcabouço. Caso o adaptador ainda não exista, ele deve ser desenvolvido de modo a prover a comunicação entre sensor e Arcabouço.

A estrutura que contém as diretrizes básicas que deve-se seguir no desenvolvimento do adaptador é ilustrada na Figura 4.4 e são explicadas nos tópicos a seguir.

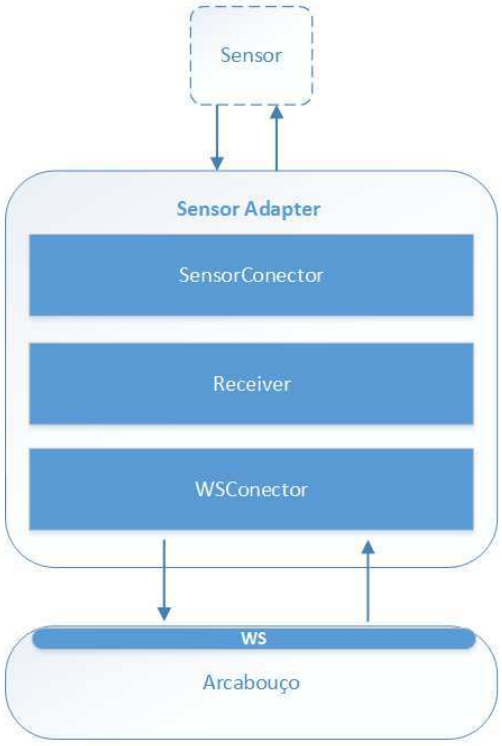


Figura 4.4: Diretrizes básicas para o desenvolvimento de um adaptador de sensor para o Arcabouço

- **Sensor** O sensor é responsável por fornecer dados/medições e disponibilizá-los perio-

[1]Exemplos: *Smartphones*, placa Arduíno [4] com *Ethernet Shield Arduíno* [3], Raspberry Pi [37]

dicamente ao adaptador (`Sensor Adapter`).

- **Sensor Connector** Este componente do adaptor é responsável por se conectar ao sensor desejado. Este componente, por exemplo, pode ser um conector de software via *Bluetooth* ou um programa rodando em uma placa Arduíno [4] que pode conectar-se com um sensor que coleta as medições.

  Vale ressaltar que o `Sensor Conector` pode ser desenvolvido em outra tecnologia, não se limitando às duas citadas. Após o sensor ser escolhido, este componente conecta-se ao sensor, e a partir deste ponto o adaptador deve estar pronto para receber as medições e repassá-las ao Arcabouço.

- **Receiver** Este componente tem a responsabilidade de receber o dado/medição do sensor, tratá-lo e deixá-lo compatível com o formato cadastrado no Arcabouço. Para que o mesmo funcione, previamente, a conexão do módulo anterior com o sensor tem que estar funcionando sem problemas.

- **WSConnector** Este componente é responsável por receber o dado/medição, conectar-se ao *Web Service* do Arcabouço e enviar a medição coletada. O `WSConnector` recebe esta medição do módulo `Receiver` e automaticamente conecta-se ao *Web Service* para após isso enviá-la ao Arcabouço.

4.3.2 Camada de acesso ao Arcabouço

Esta camada é composta por um *Web Service REST* o qual contém vários serviços de acesso ao Arcabouço [44]. Em outras palavras este conjunto de serviços formam a *API RESTful* que fornece várias funcionalidades.

Através destes é possível integrar o Arcabouço ao adaptador do sensor tendo como objetivo a criação de uma nova aplicação para o monitoramento pervasivo de ambientes, que possa ser desenvolvida pelo desenvolvedor, de modo a tornar possível o envio de medições, e o monitoramento no domínio escolhido.

Com acesso a estes serviços e às classes implementadas no Arcabouço, o desenvolvedor pode criar aplicações para o monitoramento pervasivo de ambientes nos domínios que necessite.

Tabela 4.2: Principais serviços *Web Service* disponibilizados pela *Camada de acesso ao Arcabouço*

Serviços	**Descrição**
addMeasurement	Responsável por receber as medições e enviar ao Arcabouço;
addStructure, getStructure, updateStructure	Métodos relacionados a criação, consulta e atualização de uma estrutura ou ambiente monitorado;
addLocation, getLocation, updateLocation	Relacionados a criação, consulta e atualização dos dados de localização geográfica de uma estrutura ou ambiente monitorado;
addSensor, getSensor, updateSensor	Relacionados a criação, consulta e atualização de dados do sensor instalado em uma estrutura ou ambiente monitorado.
addVariable, getVariable, updateVariable	Relacionados a criação, consulta e atualização de uma ou mais variáveis relacionadas ao sensor. No cadastro informa-se o nome da variável, o tipo de dado monitorado e a unidade de medida;
addFiringRule, getFiringRule, updateFiringRule	Relacionados a criação, consulta e atualização das regras relacionadas aos sensores e suas respectivas variáveis.
addMaintenanceAction, getMaintenanceAction, updateMaintenanceAction	Relacionados a criação, consulta e atualização de ações de manutenção, informando o devido tipo de alerta escolhido.

É fornecido o acesso às funções de enviar medições e também funções relativas a obtenção de dados cadastrados no sistema, como, estruturas monitoradas, localização, sensores, variáveis, tipos de dados monitorados, regras, ações de manutenção, alertas, atuadores etc. Os serviços são acessíveis através de requisições *POST* [44]. Estas entidades estão detalhadas na Figura 4.3, a qual retrata o diagrama de classes simplificado do Arcabouço.

O *Web Service* é representado pela classe `MonitoringFrameworkWSService` do pacote `WebService`. Na Tabela 4.2 apresentam-se os principais serviços disponibilizados pela *Camada de acesso ao Arcabouço*.

Embora o desenvolvedor possa ter acesso a todos estes serviços, o Arcabouço também oferece uma interface web de configuração (*Camada de Configuração*), através da qual o desenvolvedor pode configurar, cadastrar e atualizar todas as informações necessárias para o desenvolvimento de aplicações para o monitoramento pervasivo de ambientes, sejam dados de estruturas, sensores, etc.

4.3.3 Gerenciador de Regras

Dentro do campo do monitoramento pervasivo de ambientes, uma das funcionalidades mais importantes é o gerenciamento de regras para disparo de ações de manutenção [47]. Um bom gerenciador de regras tem que prover funcionalidades de *Inserção de regras*, *Validação da sintaxe e semântica das expressões que representam as regras* e *Interpretador de regras*.

Considerando esta abordagem, dentro do pacote *Rule Manager* (que é ilustrado na Figura 4.5) tem-se a classe `RegisterRuleService` que é responsável justamente pela inserção de novas regras para disparo de ações de manutenção relacionadas a uma estrutura monitorada através de sensores previamente instalados.

Nesta classe o método `addRule` é o responsável por receber um "String"que é uma expressão que representa a regra a ser cadastrada, e armazená-la na base de dados do Arcabouço.

Entretanto, antes de salvar, esta regra precisa ser validada, visto que ela pode estar mal escrita e não atender aos requisitos sintáticos e semânticos definidos pelo Arcabouço. Para uma regra estar bem escrita, ela tem que atender aos requisitos usados na implementação do método `validate()` da classe *ValidatorRuleService* que representa o *Validador de Regras*. É importante destacar que estes requisitos foram definidos previamente através da elaboração de uma Gramática (Código 4.1) que define o formato das expressões que representam as regras.

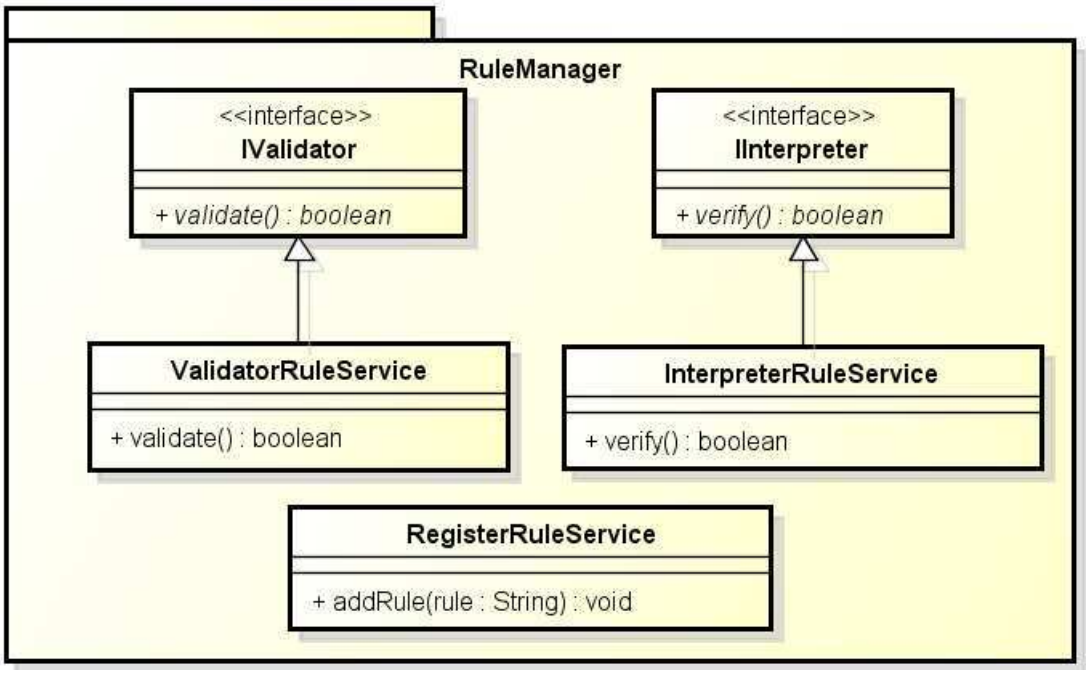


Figura 4.5: Diagrama de classes simplificado do módulo *Gerenciador de Regras*

Código Fonte 4.1: Gramática que define o formato das expressões que representam as regras

```
<Regra>:= <RegraSimples><operador><Regra> | <RegraSimples>
<RegraSimples> := ( <valor> <comparador> <variavel> <comparador> <valor>
    ) | ( <variavel> <comparador> <valor> )
<identificador> ::= <caracter><ident_aux>
<ident_aux> ::= <caracter><ident_aux>|<digito><ident_aux>| vazio
<caracter> ::= "a".."z" | "A".."Z"
<digito> ::= 0..9
<variavel> ::= <identificador>
<string> ::= <caracter> <String> | <digito> <String> | vazio
<inteiro> ::= <digito><inteiro>
<decimal> := <inteiro>.<inteiro>
<valor> := <decimal>
<comparador> := < | > | = | >= | <= | <>
<operador> := &    |      |      !     |     !=
```

Como apresentado no diagrama de classes na Figura 4.3, cada sensor tem a ele associada uma variável que representa a natureza e a forma do dado que é coletado pelo sensor. No caso do sensor de temperatura a variável "temperatura"é medida em Celsius. Cada variável tem um identificador único. Caso algum identificador especificado na regra não exista

ou não esteja associado a nenhuma variável, a mesma também é considerada inválida pelo Arcabouço.

Considerando estas características, pode-se ver que o Arcabouço fornece uma flexibilidade inovadora, que é a possibilidade de escrever regras que envolvam diferentes ambientes, diferentes sensores e tambem diferentes estruturas monitoradas. Para entender melhor tal afirmação, tem-se na Figura 4.6 um exemplo que ilustra tal funcionalidade.

Para melhor entendimento considere-se um exemplo em que o *Ambiente 1* é um *Depósito de grãos*, o *Ambiente 2* é um *ambiente externo* ao depósito, e que a *Estrutura 1* é um vão (que contém vários cilos) existentes no local e que a *Estrutura 2* é uma base onde fica instalado outro sensor de temperatura para verificar a condição externa.

Para esta situação pode-se escrever o seguinte exemplo de regra válida de disparo de ação de manutenção, `[tempS1E1 < 10 & umidS2E1 > 80 & tempS2E2 < 5]`, que em linguagem natural significa, *"Temperatura do vão que contém os grãos (**tempS1E1**) for menor que 10ºC e a umidade relativa do ar deste vão (**umidS2E1**) for maior que 80%, e a temperatura de fora do depósito (**tempS2E2**) for menor que 5ºC "*. Caso esta regra seja violada (esta verificação é efetuada pela classe `InterpreterRuleService` através do método `verify()`), um alerta pode ser enviado para o responsável, ou uma ação automatizada pode ser efetuada.

Em suma, o Arcabouço oferece uma maneira dinâmica para escrever regras, dando total flexibilidade de relacioná-las aos diferentes ambientes, estruturas e sensores associados à aplicação que o desenvolvedor queira construir.

4.3.4 Gerenciador de Alertas e de Ações de Manutenção

Na seção anterior abordou-se os aspectos e características relativas a criação, validação e interpretação de regras para disparo de ações de manutenção.

Tais ações de manutenção são disparadas como alertas. O módulo *Gerenciador de Alertas* é o responsável por gerenciá-los e enviá-los da maneira correta, no tempo certo e para o atuador especificado. A partir da Figura 4.2, estão ilustrados todos os principais componentes deste gerenciador. No componente *Cadastrador de Ações de Manutenção* são informadas a descrição da ação de manutenção a ser efetuada, e o atuador que será notificado visto que na tela de cadastro da regra já são informados os tipos de notificação de alerta (Email, Sms

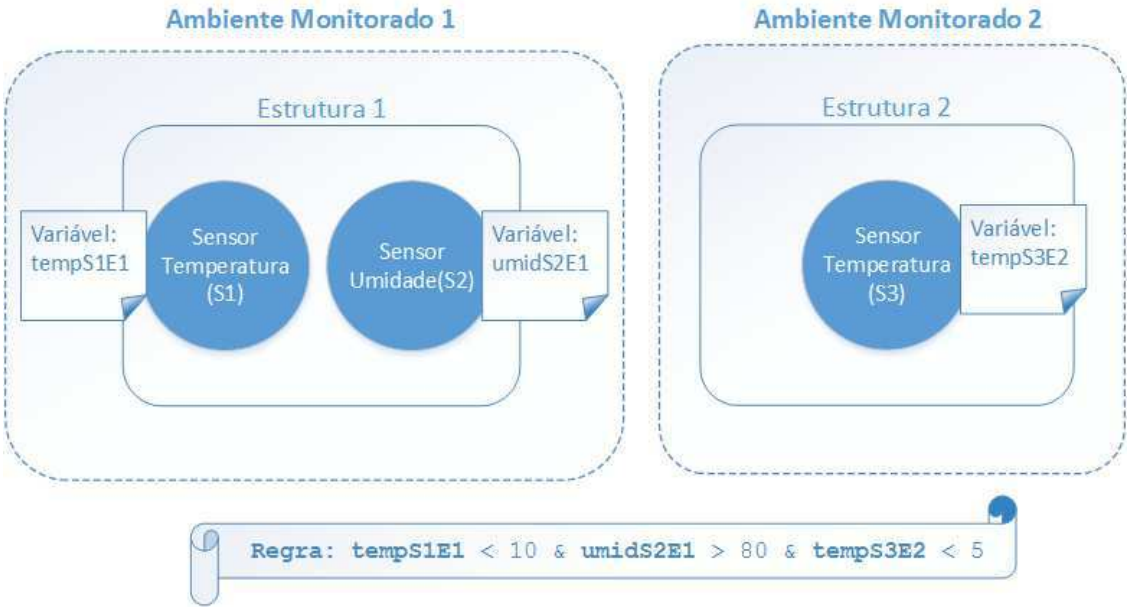


Figura 4.6: Funcionalidade a partir da qual é possível criar regras que lidam com ambientes e estruturas diferentes em conjunto

ou *Web Service* externo). Caso a regra seja violada um alerta é adicionado ao banco de dados, e o notificador correto é acionado.

Para ter-se um melhor entendimento desta funcionalidade, na Figura 4.7 é ilustrado o pacote `AlertManager` que apresenta a categorização dos tipos de notificação existentes. A *interface* `Notifier` especifica o contrato que todas as classes que a implementam tem que cumprir, em outras palavras, todas têm que implementar o método `send()` que é responsável por enviar o alerta para ações de manutenção, considerando todos os parâmetros relacionados.

O Arcabouço oferece três classes como implementação desta interface, as quais têm suas funcionalidades e características especificadas na listagem a seguir.

- `SMSNotifier`. Esta classe é a implementação responsável por enviar alerta em forma de SMS, para o telefone móvel do atuador.
- `EmailNotifier`. Esta classe é a implementação responsável por enviar alerta em forma de Email, para o atuador.
- `WSNotifier`. Esta classe é caracterizada não apenas por ser um alerta, mas também por fornecer a possibilidade do desenvolvedor fazer chamadas a um *Web Service*

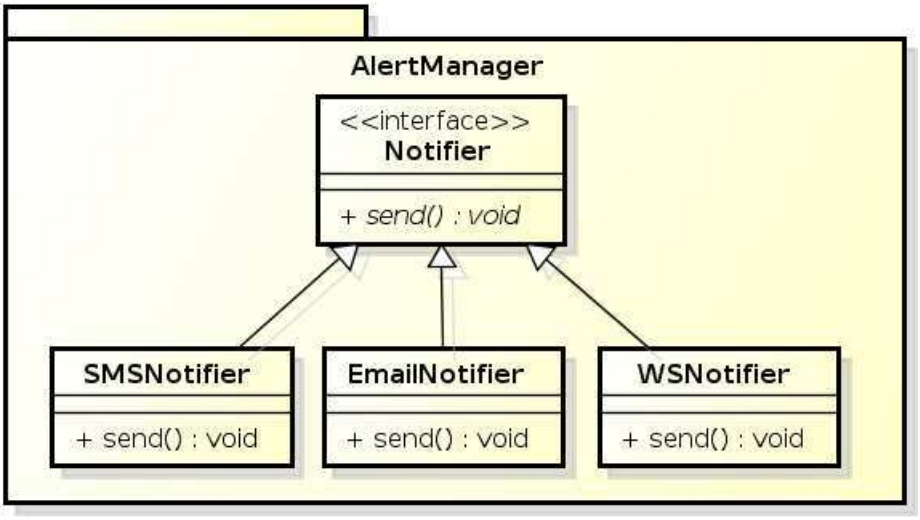


Figura 4.7: Diagrama de classes simplificado do módulo *Gerenciador de Alertas*

externo que tenha serviços responsáveis por efetuar uma ação de manutenção de forma automatizada.

Explicando mais detalhadamente a funcionalidade de notificação via *Web Service* externo, pode-se ver no Código 4.2, que para notificação a partir da classe `WSNotifier`, a implementação do método `send()` faz a chamada a um *WebService RESTFul* externo escolhido informando a URL do serviço.

Código Fonte 4.2: `WSNotifier`

```
package org.embeddedlab
class WSNotifier implements Notifier{
        def boolean send(...){
        WSClient.sendRequest("http://externalwebservice?doAnAction");
    }
}

//WSClient implementation
public class WSClient {
    public static String sendRequest(String requestString) throws
        Exception {
        URL url = new URL(requestString);
        URLConnection yc = url.openConnection();
        BufferedReader in = new BufferedReader(
```

```
                new InputStreamReader(
                yc.getInputStream()));
        String inputLine;
        String response = "";
        while ((inputLine = in.readLine()) != null) {
            response += inputLine;
            System.out.println(inputLine);
        }
        in.close();
        return response;
    }
}
```

Tal funcionalidade dá flexibilidade ao desenvolvedor de criar serviços externos que automatizem as ações de manutenção ou até mesmo usar serviços já existentes. Como exemplo, quando um motor monitorado super aquecer, fazendo com que a regra que delimita o limite de temperatura seja violada, um *Web Service* externo que desliga o motor pode ser chamado, assim evitando prejuízos maiores. Tal comportamento é ilustrado tanto na Figura 4.2 que apresenta a arquitetura, como também na Figura 4.1 que representa uma visão geral do Arcabouço. Na Figura 4.9 é apresentado um exemplo de email de alerta, e na Figura 4.8 um exemplo de SMS enviado pelo Arcabouço.

Concluindo, o Arcabouço oferece uma maneira flexível de envio de alertas, dando total liberdade para o desenvolvedor usar a estratégia de alerta de sua preferência.

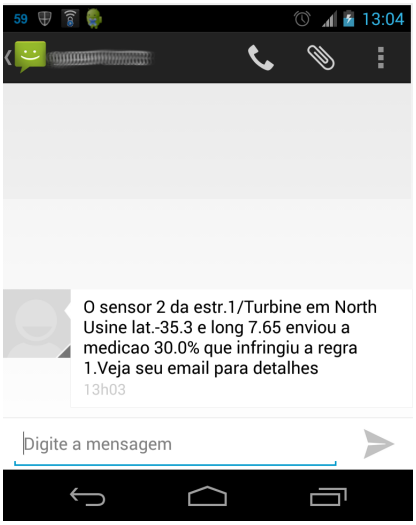


Figura 4.8: Exemplo de alerta via SMS enviado pela classe `SMSNotifier`

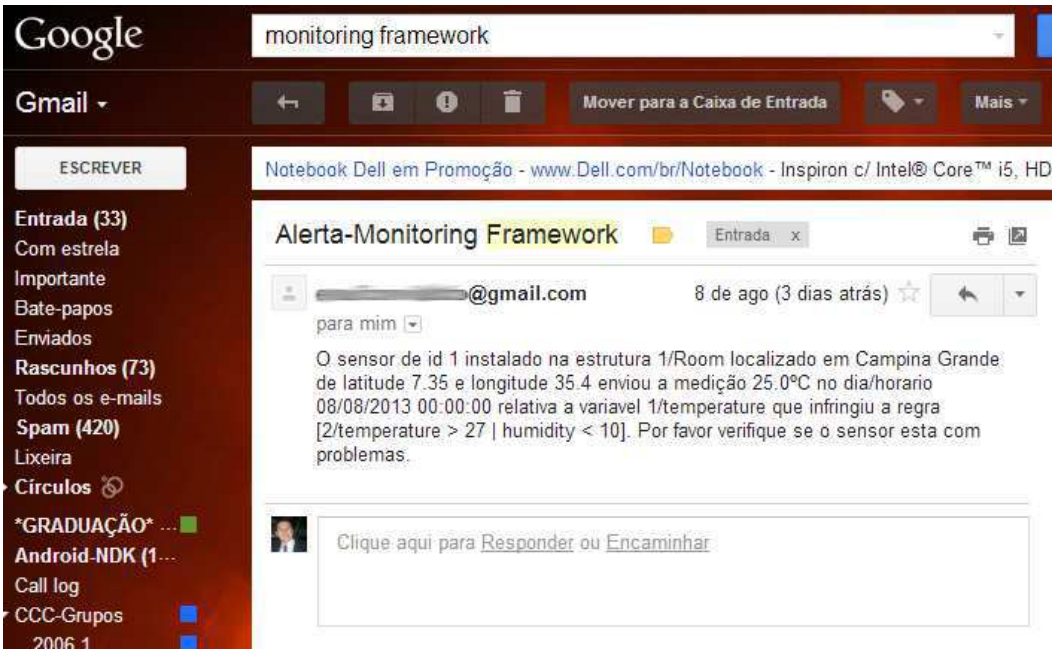


Figura 4.9: Exemplo de alerta de email enviado pela classe `EmailNotifier`

4.4 Configurando e usando o Arcabouço

A configuração do Arcabouço consiste em cadastrar todas as informações do ambiente, seja estrutura monitorada, sensores envolvidos, regras, ações de manutenção e tipos de alerta, etc. O desenvolvedor tem a liberdade de configurar o Arcabouço, seja através dos serviços

disponibilizados, os quais apresentou-se na Seção 4.3.2, ou através da *Interface Gráfica*.

Construir aplicações consiste em desenvolver uma solução que integre vários componentes, ferramentas, dispositivos etc, de modo que no final obtenha-se o resultado desejado. Para o desenvolvimento de aplicações com o Arcabouço aqui apresentado não é diferente, o desenvolvedor deverá usá-lo e plugá-lo a outros componentes de software, como o adaptador, para assim chegar ao resultado final.

Na Figura 4.10 é ilustrada a tela através da qual cadastra-se informações de estruturas, e na Figura 4.11 apresenta-se a tela para cadastro de um sensor.

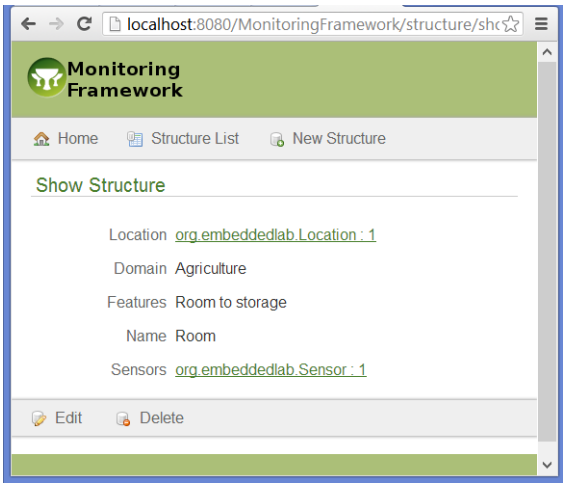


Figura 4.10: Tela com as informações de uma estrutura previamente cadastrada

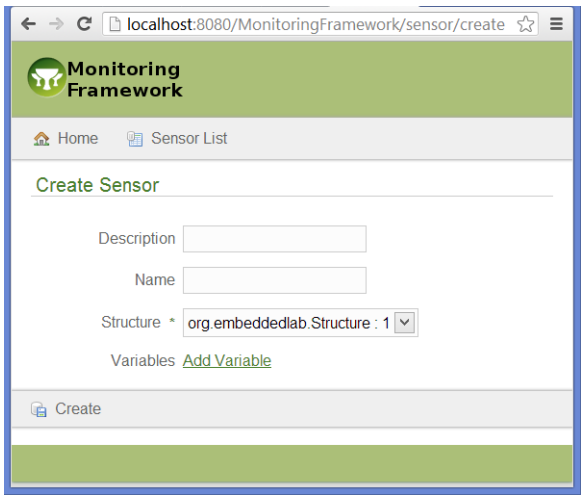


Figura 4.11: Tela de adição de sensores no Arcabouço

4.4.1 Diretrizes para a configuração e construção de aplicações para o monitoramento pervasivo de ambientes usando o Arcabouço

Para o desenvolvedor construir sua aplicação primeiro ele deve escolher os sensores com os quais vai trabalhar, após isso, ele deve cadastrar as informações a partir dos quais o Arcabouço estará pronto para ser usado no desenvolvimento da aplicação em questão. Após isso ele deve desenvolver o adaptador para integrar o Arcabouço à camada de sensores. A partir disso é necessário implementar a lógica de acesso ao *Web Service* do Arcabouço, configurando tal adaptador para encaminhar corretamente as medições coletadas através do método `addMeasurement` da classe `MonitoringFrameworkWSService`.

Para melhor entendimento, pode-se verificar o passo a passo para a construção de uma nova aplicação para o monitoramento pervasivo de ambientes usando o Arcabouço, através do diagrama de atividades ilustrado na Figura 4.12.

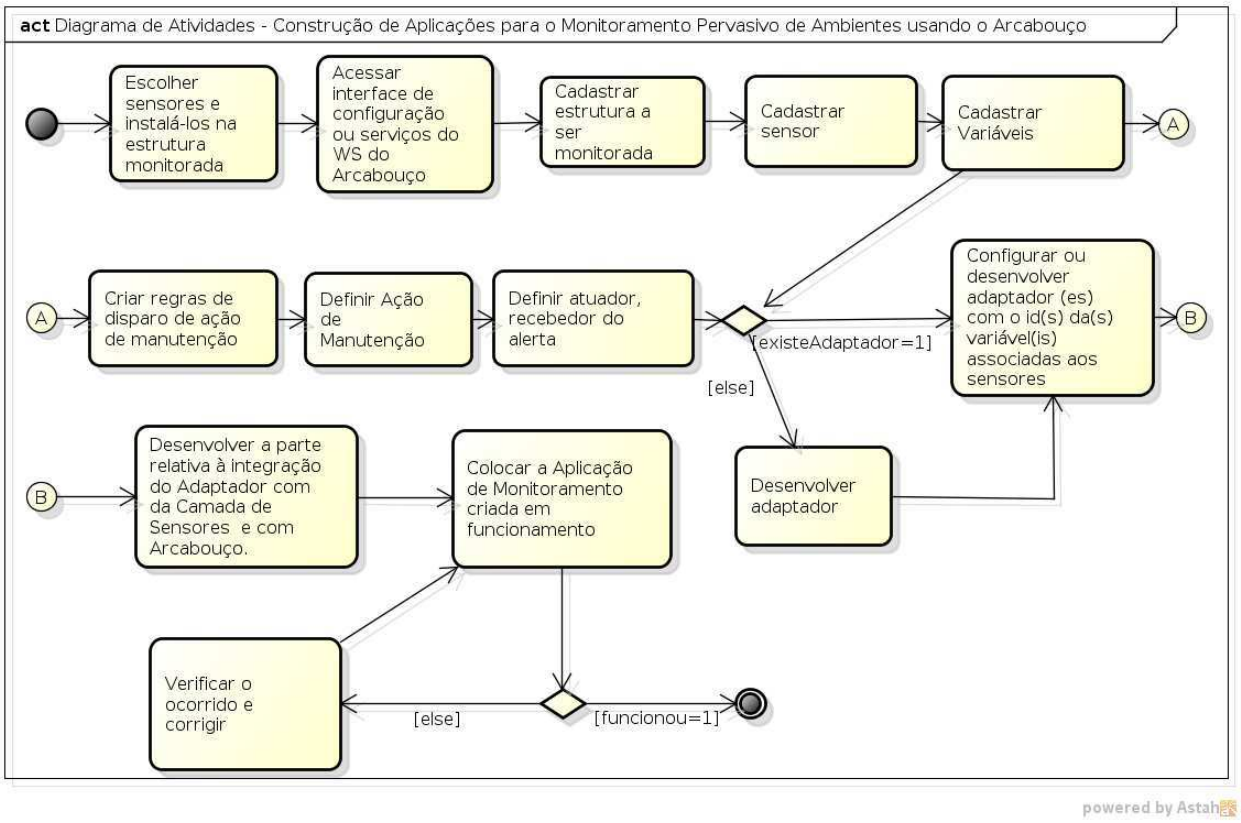


Figura 4.12: Diagrama de atividades para construção de aplicações para o monitoramento pervasivo de ambientes usando o Arcabouço

Uma aplicação desenvolvida utilizando o Arcabouço deve seguir determinadas diretrizes arquiteturais. As mesmas dizem respeito a estrutura e integração dos diversos componentes que devem compor a aplicação e a correlação entre os mesmos. Sendo assim, a aplicação quando pronta terá a arquitetura devidamente ilustrada pela Figura 4.13.

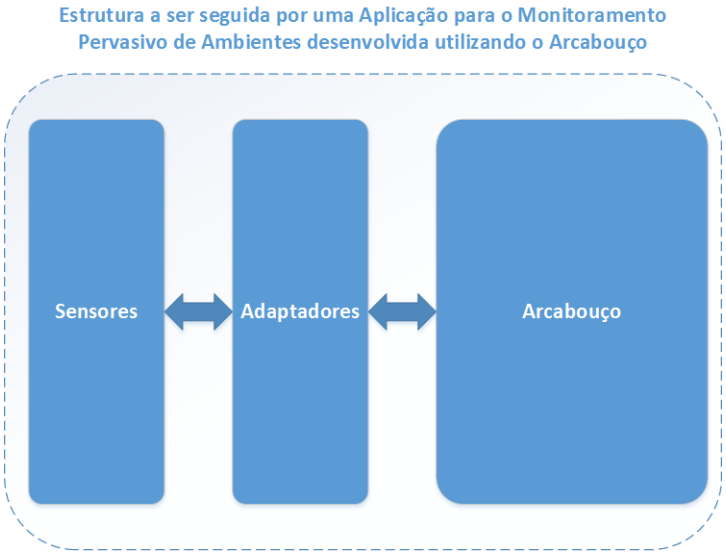


Figura 4.13: Arquitetura de uma nova aplicação para o monitoramento pervasivo de ambientes desenvolvida utilizando o Arcabouço

4.5 Extensibilidade do Arcabouço

Um Arcabouço deve ter como requisito a extensibilidade como forma de expandir suas funcionalidades principais dando mais liberdade ao desenvolvedor que o utiliza. Esta seção mostra os passos a serem seguidos para estender as partes partes do Arcabouço. A maioria delas são extensíveis a partir do uso de *interfaces*, o que reduz o acoplamento do código.

4.5.1 Classes de Verificação, Validação e Interpretação de Regras

A classe responsável pela validação de regras atualmente é a `ValidatorRuleService`, que implementa a *interface* `IValidator`. Esta interface agrega os métodos que definem o comportamento comum de todos validadores que por ventura venham a existir.

Uma nova classe validadora de regras deve implementar a *interface* `IValidator` e prover a implementação concreta de seus métodos, definindo seu código específico. O Có-

digo 4.3 mostra a interface `IValidator`. Já o Código 4.4 é um exemplo de classe que implementa uma nova forma de validar regras cumprindo o contrato desta *interface*.

Código Fonte 4.3: Interface `IValidator`

```
package org.embeddedlab
interface IValidator {
    def boolean validate(String rule);
}
```

Código Fonte 4.4: Um Exemplo de classe validadora de regras

```
public class NewRuleValidator implements IValidator {
    @Override
    def boolean validate(String rule) {
        //Code of validation
        return ...;
    }
}
```

Por outro lado, a classe responsável pela verificação de violação de regras é a `InterpreterRuleService`, que implementa a *interface* `IInterpreter`. Novas classes que sejam verificadoras de violação de regras devem implementar a interface *IInterpreter*, de modo a prover uma implementação concreta de seus métodos. Esta interface pode ser vista no Código 4.5. Já o Código 4.6 é um exemplo de classe que implementa uma nova forma de interpretar, ou seja, uma nova forma de verificar se uma regra foi violada, cumprindo o contrato da *interface IInterpreter*. A estrutura atual do pacote de regras é ilustrada na Figura 4.5.

Código Fonte 4.5: Interface `IInterpreter`

```
package org.embeddedlab

interface IInterpreter{
    def boolean verify(String rule);
}
```

Código Fonte 4.6: Um Exemplo de Classe Interpretadora de Regras

```
public class NewRuleInterpreter implements IInterpreter {
    @Override
    def boolean verify(String rule) {
        //Code to interpreter rule
        return ...;
    }
}
```

4.5.2 Classes relativas às notificações de ações de manutenção a partir de alertas

Atualmente o Arcabouço contém as classes, `EmailNotifier`, `SMSNotifier` e `WSNotifier` que são responsáveis pelo disparo de ações de manutenção a partir de alertas. Estas classes implementam a *interface* `Notifier`. Esta agrega o método que define o comportamento comum de todos notificadores de ações de manutenção através de alertas, que por ventura venham a existir.

Uma nova classe deste tipo deve implementar a interface `Notifier` e prover a implementação concreta de seus métodos, definindo seu código específico. O Código 4.7 mostra a *interface* `Notifier`. Já o Código 4.8 é um exemplo de classe que implementa uma nova forma de notificação.

Código Fonte 4.7: Interface `Notifier`

```
package org.embeddedlab
interface Notifier {
    def boolean send(...)
}
```

Código Fonte 4.8: Um Exemplo de Classe Interpretadora de Regras

```
class NewNotifier implements Notifier {
    public NewNotifier(){}
    @Override
    def boolean send(...){
        //Code ...
    }
}
```

Através da possibilidade de extensão destas classes o Arcabouço pode ser extendido adaptando-se as necessidades extras de customização do desenvolvedor.

Há que ressaltar que mesmo sem precisar alterar ou estender nada, ele pode desenvolver aplicações para o monitoramento pervasivo de ambientes sem se importar com o domínio em questão, apenas tendo o trabalho de usar as ferramentas providas para conectar a infraestrutura ou ambiente ao Arcabouço, criando assim aplicações para o monitoramento pervasivo de ambientes com menos esforço, menos complexidade e menos retrabalho.

4.6 Conclusão

Neste capítulo foi apresentada uma visão geral do Arcabouço para desenvolvimento de aplicações para o monitoramento pervasivo de ambientes.

Inicialmente foi apresentada uma visão geral, através da qual pode-se ter uma ideia global do funcionamento como um todo. Em seguida foi apresentada a arquitetura e o detalhamento de suas diversas camadas e módulos, *Adaptador do sensor ao Arcabouço*, *Camada de acesso ao Arcabouço*, *Gerenciador de Regras*, *Módulo Gerenciador de Alertas e de Ações de Manutenção* etc.

Logo em seguida foram apresentados os passos necessários para configurar e usar o Arcabouço para desenvolvimento de aplicações. E para concluir mostrou-se como extender os principais componentes existentes de modo prático e simples. A prova de conceito e a validação da Arcabouço foram realizadas por meio de três estudos de caso para domínios diferentes, os quais são discutidos no Capítulo 5.

Capítulo 5

Estudos de caso

Neste capítulo apresenta-se o desenvolvimento de três estudos de caso para demonstrar o uso do Arcabouço para desenvolvimento de aplicações para o monitoramento pervasivo de ambientes.

Primeiro apresenta-se a tecnologia e a estrutura utilizada para coleta de dados dos sensores e posteriormente é feita uma descrição do desenvolvimento do adaptador para os mesmos. Após isto, o capítulo apresenta uma descrição das três aplicações, os domínios e os sensores utilizados em cada uma, juntamente com a forma como interagem com o Arcabouço. Também são apresentados os passos que foram seguidos para construção de cada uma destas aplicações.

5.1 Tecnologia utilizada para conexão de sensores, coleta e envio de medições ao Arcabouço

Utilizando-se o Arcabouço, foram desenvolvidas aplicações em domínios distintos usando sensores diferentes. Para este estudo, foram utilizados sensores ligados ao microcontrolador Arduíno.

O Arduíno [4] é uma plataforma de prototipagem eletrônica de código aberto(*opensource*[1]) baseada na flexibilidade, com hardware e software fáceis de usar. O Arduíno é destinado a qualquer pessoa interessada em criar objetos ou ambientes

[1] http://opensource.org

interativos. Ele é uma placa que contém um microcontrolador que pode ser programado usando a linguagem de programação do Arduíno [4], e a partir desta, controlar qualquer item que esteja plugado.

Baseado nesta abordagem, ele pode perceber o ambiente através de uma variedade de sensores que podem ser conectados à placa através de suas portas analógicas e digitais. Apoiados por estas características, esta foi a plataforma escolhida para o desenvolvimento do software adaptador, e serviu como base para escolha de cada sensor utilizado neste trabalho.

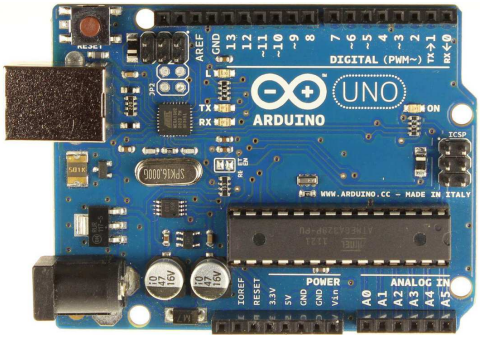


Figura 5.1: Arduíno Uno [4]

Utilizou-se a placa Arduíno (Figura 5.1) juntamente com sensores de domínios distintos, para o desenvolvimento da *Camada de Sensoriamento* e da *Camada Adaptadora* para cada estudo de caso. Tal tecnologia foi escolhida pelo fato de ser uma plataforma de prototipagem fácil de usar, além ter uma grande diversidade de sensores que se adequam às necessidades de pesquisadores e entusiatas da área de hardware e software.

Nos estudos de caso, o Arduíno foi responsável pela coleta de medições dos sensores (codigo relativo a coleta de medições). Já para conexão com o Arcabouço através de seu *Web Service* e consequente acesso ao método `addMeasurement` para envio de medições, foi utilizada uma placa *Ethernet Shield* [3] (código responsável pelo envio das medições) conectada ao microcontrolador. Esta placa é ilustrada na Figura 5.2.

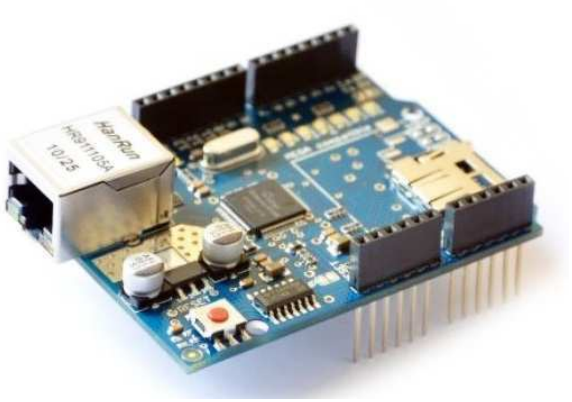


Figura 5.2: *Ethernet Shield* para Arduíno [3]

5.2 Casos desenvolvidos para validação do Arcabouço

Nesta seção apresenta-se os estudos de caso de aplicações para o monitoramento de ambientes pervasivos desenvolvidas usando o Arcabouço. A criação de cada aplicação consistiu no desenvolvimento de software e procedimentos de configuração necessários para criação e integração de suas 3 camadas: *Camada de Sensoriamento*, *Camada Adaptadora* e *Camada do Arcabouço*.

5.2.1 Aplicação para o monitoramento pervasivo de ambientes através de medições de corrente elétrica

Esta é uma implementação no domínio de Energia Elétrica, usando o Arcabouço descrito no capítulo anterior, que possui como objetivo monitorar medições de corrente elétrica de um determinado ambiente. Um cenário de aplicação da solução desenvolvida é apresentado a seguir:

"José tem vários equipamentos elétricos caros numa pequena empresa na qual trabalha, e necessita monitorá-los de modo a que ele seja avisado quando algum aumento ou diminuição brusca aconteça na corrente elétrica. Isso pode ocorrer na incidência de um curto circuito, ou então quando algum equipamento esteja com defeito, gerando na rede índices de corrente fora dos padrões."

A forma de monitorar este ambiente do José é através do monitoramento da corrente

elétrica. Para isso desenvolveu-se uma aplicação para este caso, retratando o ambiente, demonstrando o funcionamento do Arcabouço para o monitoramento pervasivo de ambientes através da variação de corrente elétrica.

Esta aplicação tem a seguinte arquitetura expressada na Figura 5.3, para fins de melhor entendimento e auxílio às subseções que explicam cada camada.

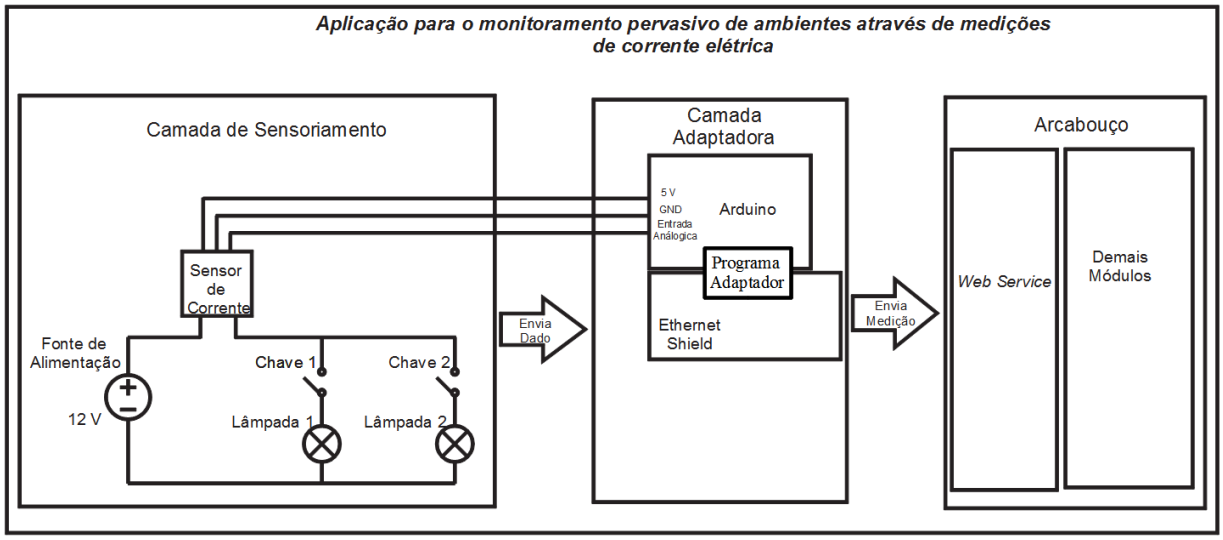
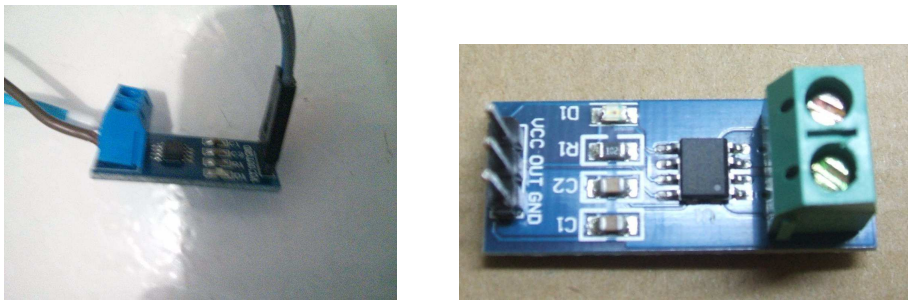


Figura 5.3: Arquitetura da aplicação para o monitoramento pervasivo de ambientes através de medições de corrente elétrica

Camada de Sensoriamento

Para conectar o sensor foi montado um ambiente simples para simular este caso. O sensor utilizado foi o *Sensor de corrente ACS712* para Arduíno retratado na Combinação de Imagens 5.4.

Este sensor de corrente opera como um tradutor que verifica um valor de corrente que circula em um circuito e transforma esse valor em uma informação para que seja enviada para o Arduíno. O sensor utilizado possui uma escala de de 30 A. O circuito montado para realização dos testes é ilustrado na Figura 5.3, e a montagem foi realizada utilizando uma fonte ATX, Arduíno, lâmpadas e uma placa *Ethernet Shield* pode ser visualizada na imagem 5.5.



(a) Sensor de corrente ACS712 no ambiente montato para a validação

(b) Sensor de corrente ACS712

Figura 5.4: Imagens do Sensor de corrente ACS712

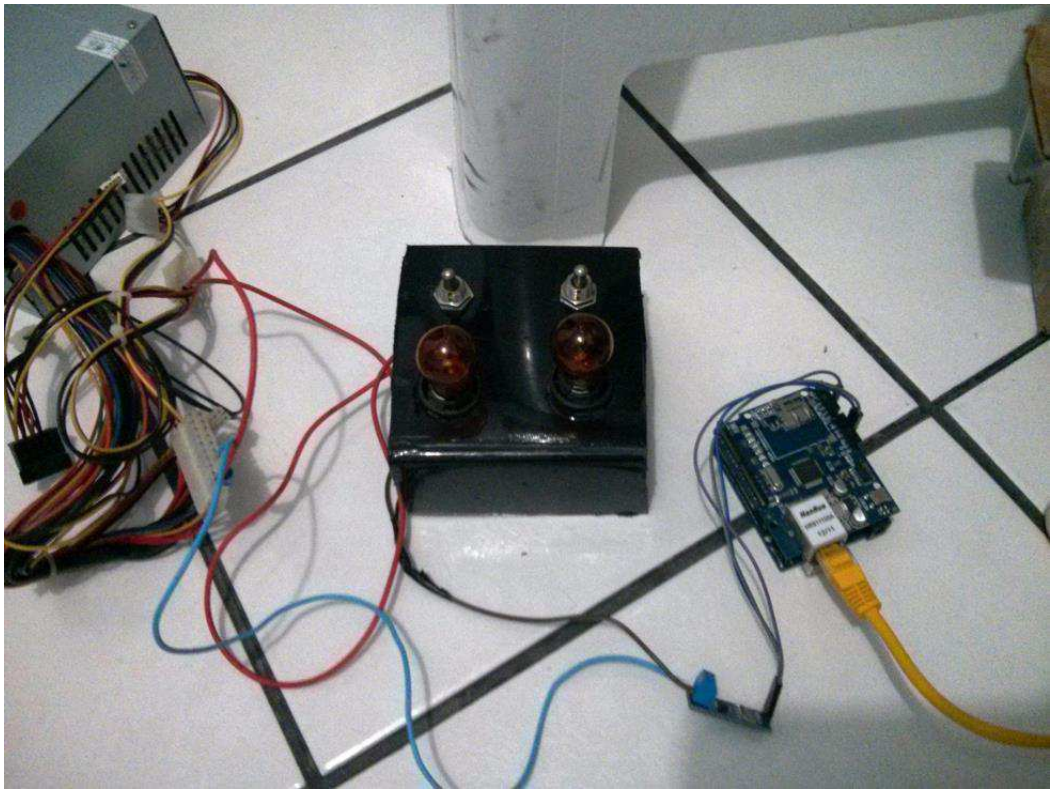


Figura 5.5: Ambiente montado para a validação com montagem real do circuito

Nesta montagem foram utilizadas duas lâmpadas de 21 *Watts* alimentadas através de uma fonte de corrente contínua de *12 V*. No instante que a `chave 1` é ligada, a `lâmpada 1` acende, possibilitando a passagem de corrente pelo circuito. A `lâmpada 2` é ligada pela `chave 2`. Através da operação das duas chaves pode-se variar o valor da corrente que circula no circuito. A corrente total que circula pelo circuito foi determinada em função da potência das lâmpadas de acordo com a expressão 5.1:

$$i = \frac{P}{V} \tag{5.1}$$

Onde: *P* é a potência elétrica da lâmpada; *V* é a tensão da fonte de alimentação; *i* corrente elétrica.

No instante que uma das lâmpadas está ligada a corrente que circula pelo circuito é determinada pela a expressão 5.2:

$$i = \frac{21}{12} = 1,75A \tag{5.2}$$

Por outro lado, quando as duas lâmpadas estão ligadas a corrente total que circula pelo circuito é determinada pela a expressão 5.3:

$$i = \frac{21 + 21}{12} = 3,5A \tag{5.3}$$

Lembrando que no intervalo em que as duas lâmpadas estão apagadas a corrente é zero. Utilizando esse circuito pode-se testar o funcionamento do Arcabouço em função da variação de corrente. Todo esse detalhamento é uma abordagem explicativa de como está organizado o experimento, detalhando todo o ambiente da camada de sensores para este caso.

Camada do Arcabouço

Para que o Arcabouço possa funcionar em conjunto com a *Camada Adaptadora*, é necessária a persistência na sua base de dados das informações relativas ao ambiente em questão. Para facilitar o trabalho, os dados foram cadastrados através da interface gráfica. Na Tabela 5.1 estão os dados informados. Já nas Figuras 5.6 e 5.7 são ilustradas a variável e a regra criada para o sensor de corrente elétrica desta aplicação.

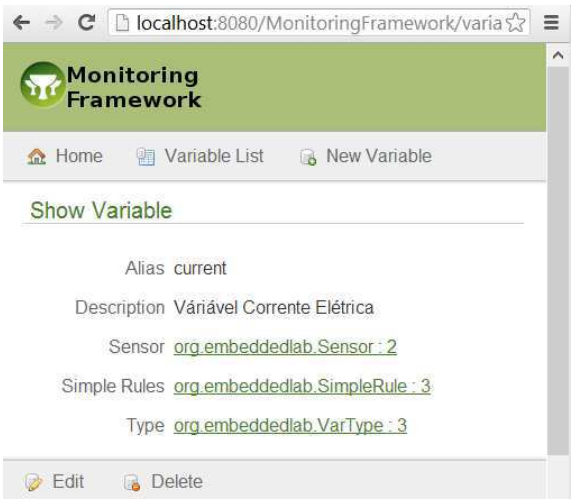


Figura 5.6: Tela que exibe a variável *current* no Arcabouço para o sensor de corrente elétrica.

Tabela 5.1: Dados informados ao Arcabouço na aplicação de monitoramento de medições de energia elétrica

Entidade	**Dados de Entrada**
Localização	`location(id: 1, name:"Campina Grande", latitude:-7.23", longitude:-35.88 ")`
Estrutura	`structure(id: 1, name:"Ambiente montado", domain:"Energia Elétrica", location: location)`
Sensor	`sensor(id:1, name: "Sensor de Corrente", structure: structure)`
Variável	`variable(alias:"current", sensor:1, vartype:(type: "double", unit: "A"))`
Regra	`firingRule(id: 1, completeRule: "current > 2", alertType: "sms", maintenanceAction:maintenanceAction)`
Ação de Manutenção	`maintenanceAction(description:"Verificar o ocorrido", personToBeNotified( name:"Romeryto", email:"romeryto@live.com", telefone:"558388888888"))`

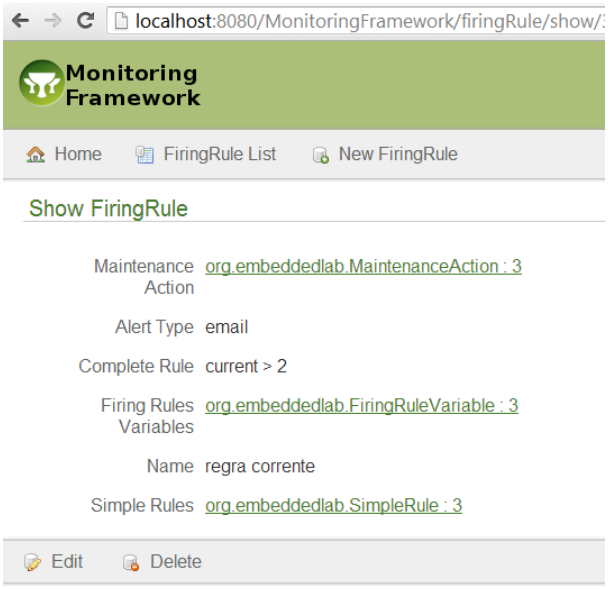


Figura 5.7: Tela que exibe a regra criada no Arcabouço para o sensor de corrente elétrica ACS712

A partir deste momento, com o Arcabouço devidamente configurado, o próximo passo foi o trabalho relativo a sua integração com a *Camada Adaptadora*.

Camada Adaptadora

Com a camada de sensores previamente configurada, foi desenvolvida a *Camada Adaptadora* com o objetivo de coletar as medições de corrente elétrica e enviá-las ao Arcabouço. Tal adaptador para o sensor de corrente escolhido foi desenvolvido usando a própria linguagem de programação do Arduíno. O desenvolvimento deste foi dividido em duas partes: Código Coletor de medições 5.1 e o Código 5.2 que se conecta e envia a medição coletada do sensor de corrente para o Arcabouço, através da placa *Ethernet Shield*.

Código Fonte 5.1: Código coletor de medições do sensor de corrente elétrica plugado ao Arduíno

```
#include <SPI.h>
#include <Ethernet.h>
//inicialização das variáveis
#define A0 A0 //interface analógica do Arduíno de onde os dados de
    corrente serão lidos
...
```

```
EthernetClient client;
int variableSensorIDOfMonitoringFramework = 3; //id da variável
    cadastrada no Arcabouço

//Código que coleta a medição do sensor de corrente
int getCurrentMeasurement(){
 int sensorValue = analogRead(A0);
 //O código a seguir diz para o Arduíno que o valor lido pelo sensor deve
     ser transformado de 0 a 1023 para -30A a +30A.
 int outputValue = map(sensorValue, 0, 1023, -30, 30);
 return outputValue;
}
```

Código Fonte 5.2: Código que integra a *Camada de Sensoriamento* ao *Arcabouço* através do envio de medições de corrente elétrica

```
//Código que conecta-se e envia a medição para o Arcabouço
void sendMeasurement(char* currentMeasurement){
  if (client.connect("http://localhost:8080/MonitoringFramework/
      monitoringFrameworkService/", 8080)) {
    Serial.println("connected");
    // Faz a requisição:
    client.println("POST addMeasurement?idVariable="+
        variableIDOfMonitoringFramework +"&value="+currentMeasurement);
    client.println("http://localhost:8080");
    client.println("Connection: close");
    client.println();
  }
  else {
    Serial.println("connection failed");
  }
}
```

Em linguagem Arduíno, através do método `sendMeasurement`, este programa adaptador conecta-se ao *Web Service* do Arcabouço e envia a medição de corrente coletada em `getCurrentMeasurement()`, fazendo uma requisição *POST* ao serviço `addMeasurement`.

Aplicação em funcionamento

A partir deste ponto o objetivo foi verificar se realmente o Arcabouço funciona como proposto para este domínio. Utilizando-se o ambiente montado, e considerando todo o desenvolvimento e integração feita entre as três camadas explicitadas nas subseções anteriores (Figura 5.3), resultando numa nova aplicação para o monitoramento pervasivo de ambientes, verificou-se o comportamento do Arcabouço à medida que o mesmo recebia medições.

Na Subseção 5.2.1, foi apresentada a forma de funcionamento deste ambiente e o que ocorre com o valor da corrente de acordo com a quantidade de luzes ligadas. Com base nisso, foi feito exatamente o explicado, quando ligou-se a `chave 1` e `lâmpada 1` acendeu, o valor da corrente elétrica, foi para 1,75 A, porém nenhum disparo de alerta para ação de manutenção foi enviado, visto que, a regra cadastrada (ver Figura 5.7) não foi violada, ou seja, o valor não ultrapassou *2 A*. A partir deste ponto ligou-se a `chave 2` e a `lâmpada 2` acendeu, então o valor da corrente elétrica subiu para *3,5A* o que violou a regra. Como esta regra cadastrada prevê, um alerta via SMS foi enviado para a pessoa responsável (ver Figura 5.9). Um email também foi enviado[2], este pode ser visto na Figura 5.8. Assim, comprovou-se que a aplicação desenvolvida com o Arcabouço para este domínio funciona como o esperado, resolvendo o problema do José.

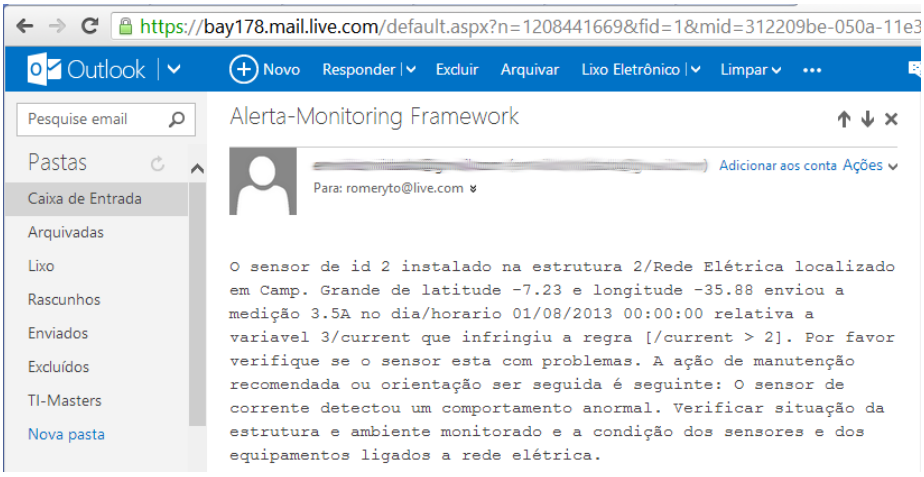


Figura 5.8: Email de alerta relativo sensor de corrente elétrica ACS712

[2]O envio de email é feito sempre, independente do notificador escolhido

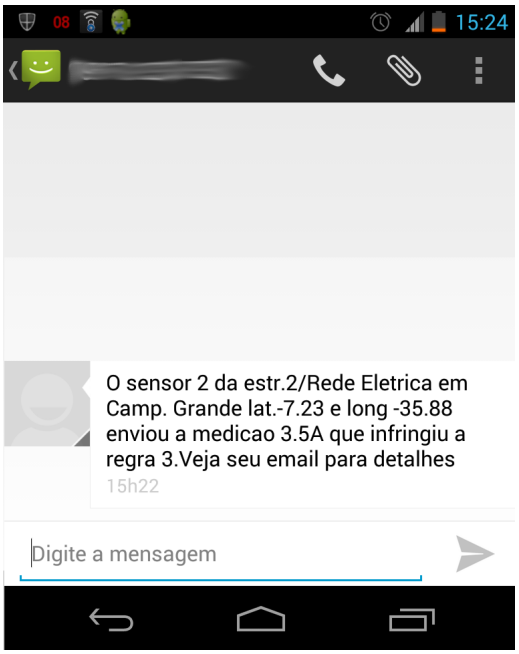


Figura 5.9: SMS de alerta relativo sensor de corrente elétrica ACS712

5.2.2 Aplicação para o monitoramento pervasivo de ambientes através de medições de temperatura e umidade

Esta segunda aplicação é uma implementação que pode aplicar-se a vários domínios, usando o Arcabouço descrito no capítulo anterior. Temperatura e umidade são informações que são necessárias em vários domínios, dependendo de qual é monitorado. Baseando-se nestes aspectos, um cenário de aplicação da solução desenvolvida para o domínio agrícola é apresentado a seguir:

"Maria Josefina tem um galpão onde são guardados grãos da produção das roças da cooperativa da qual seu avô é presidente. Como ela é muito atenta, viu logo que os grãos estavam estragando quando a temperatura ou a umidade relativa do ar no local ficavam muito altas nos tempos de verão. Maria necessita monitorar o ambiente de modo a que quando tal situação comece ela possa resguardar os grãos, seja controlando a temperatura do ambiente, e/ou efetuando procedimento para diminuir a umidade."

A forma de monitorar este ambiente de Maria é através do monitoramento da temperatura e da umidade do local. Com base neste caso, desenvolveu-se nesta pesquisa uma aplicação para monitoramento pervasivo deste tipo ambiente, usando o Arcabouço.

Esta aplicação tem a seguinte arquitetura expressada na Figura 5.10, para fins de melhor entendimento e auxílio às subseções seguintes explicam cada camada.

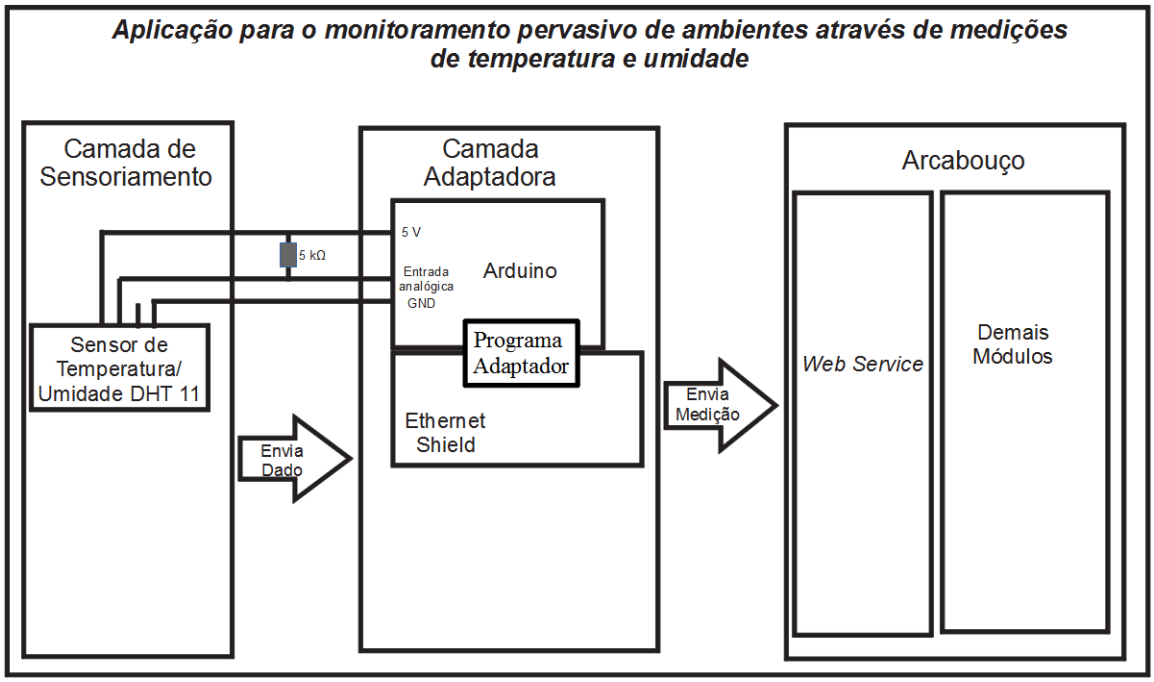


Figura 5.10: Arquitetura da aplicação para o monitoramento pervasivo de ambientes através de medições de temperatura e umidade.

Camada de Sensoriamento

Na camada de sensores montou-se um ambiente simples para simular tal caso. O sensor utilizado foi o *Sensor de DHT 11* para Arduíno retratado na combinação de imagens 5.11. Este sensor opera na escala "Celsius (ºC)"para temperatura e "porcentagem(%)"para umidade relativa do ar.

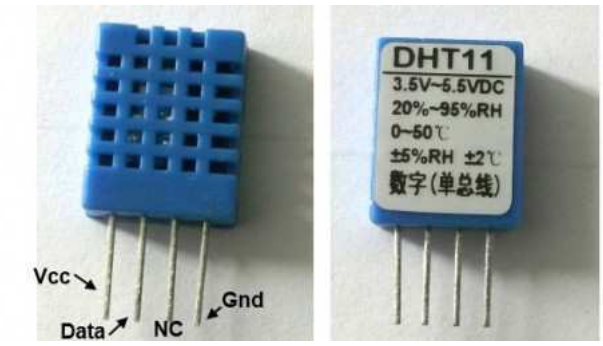


Figura 5.11: Sensor DHT 11

Este sensor trabalha com duas variáveis ao mesmo tempo, temperatura ambiente e umidade relativa do ar. É um sensor muito utilizado em conjunto com o Arduíno.

Camada do Arcabouço

Como na aplicação anterior, os dados de entrada para o Arcabouço foram cadastrados através da interface gráfica. Na Tabela 5.2 apresentam-se estes dados. Já na Figura 5.12 é ilustrada a regra criada para para o sensor DHT 11 desta aplicação.

Tabela 5.2: Dados informados ao Arcabouço na aplicação de monitoramento de medições de temperatura e umidade

Entidade	**Dados de Entrada**
Localização	`location(id: 1, name:"Campina Grande", latitude:-7.23", longitude:-35.88 ")`
Estrutura	`structure(id: 1, name:"Galpão de Grãos", domain:"Agrícola", location: location)`
Sensor	`sensor(id:1, name: "Sensor DHT11", structure: structure)`
Variáveis	`variable1(alias:"temperature", sensor:1, vartype:(type: "double", unit: "°C")) variable2(alias:"humidity", sensor:1, vartype:(type: "double", unit: "%"))`
Regra	`firingRule(id: 1, completeRule: "temperature >= 20.00 | humidity > 55.00", alertType: "email", maintenanceAction:maintenanceAction)`
Ação de Manutenção	`maintenanceAction(description:"Verificar o ocorrido", personToBeNotified( name:"Romeryto", email:"romeryto@live.com", telefone:"558388888888"))`

A partir deste momento, com o Arcabouço devidamente configurado, o próximo passo foi o trabalho relativo a sua integração com a *Camada Adaptadora*.

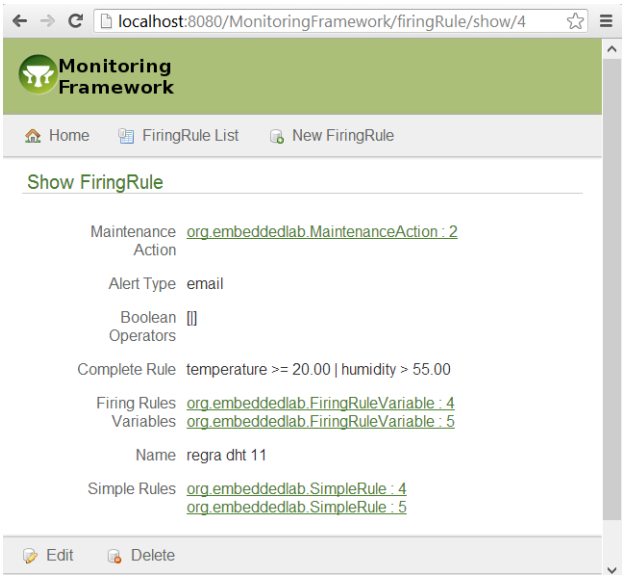


Figura 5.12: Tela que exibe a regra criada no Arcabouço para o sensor DHT 11.

Camada Adaptadora

Com a camada de sensores previamente configurada, foi desenvolvida a *Camada Adaptadora* com o objetivo de enviar as medições de temperatura e de umidade coletadas do sensor DHT 11. Tal adaptador para o sensor escolhido também foi desenvolvido usando a própria linguagem de programação do Arduíno. Para esta validação foi montado um ambiente que é ilustrado na Figura 5.13.

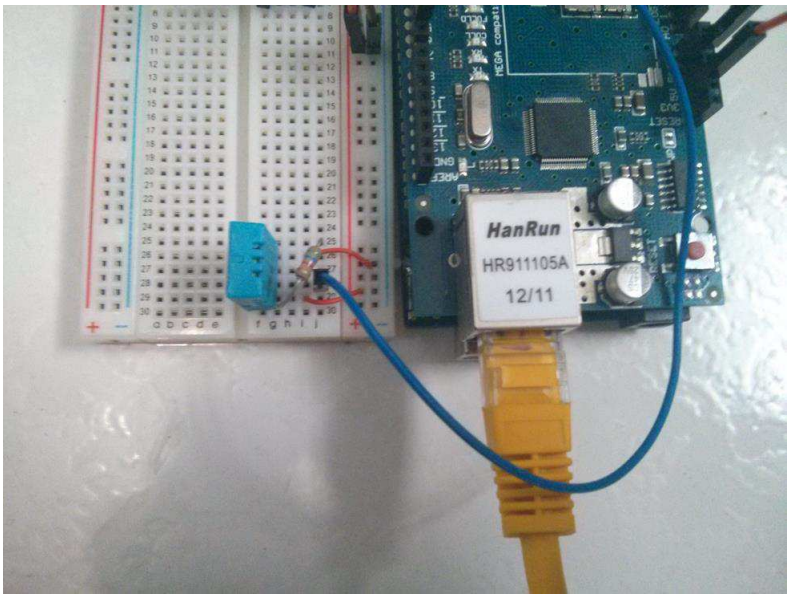


Figura 5.13: Ambiente montado com o Arduíno, *Ethernet Shield* e sensor DHT 11 .

O desenvolvimento desta camada foi dividido em duas partes: Código Coletor de Medições (`getDHT11Measurement(type)`) [3], e o Código 5.3 que conecta-se e envia a medição coletada do sensor para o Arcabouço, através da placa *Ethernet Shield*.

Código Fonte 5.3: Código adaptador que integra a *Camada de Sensoriamento*(sensor DHT 11) ao *Arcabouço* através do envio de medições de temperatura e humidade

```
//Código que conecta-se e envia a medição para o Arcabouço
void sendMeasurement(char* dht11Measurement, int idVariable){
  if (client.connect("http://localhost:8080/MonitoringFramework/
     monitoringFrameworkService/", 8080)) {
    Serial.println("connected");
    // Faz a requisição:
    client.println("POST addMeasurement?idVariable="+ idVariable +"&value
       ="+dht11Measurement);
    client.println("http://localhost:8080");
    client.println("Connection: close");
    client.println();
  }
  else {
    Serial.println("connection failed");
  }
}
```

Em linguagem Arduíno, através do método `sendMeasurement`, este programa adaptador conecta-se ao *Web Service* do Arcabouço e envia as medições de temperatura ou umidade (método `getDHT11Measurement(type)`), fazendo uma requisição *POST* ao serviço `addMeasurement`. Em termos arquiteturais (Figura 5.10), a medição é coletada da *Camada de Sensores*, enviada ao Arduíno e, através da placa *Ethernet Shield*, é enviada ao Arcabouço.

Aplicação em funcionamento

A partir deste ponto o objetivo foi verificar se realmente o Arcabouço funciona como proposto para este domínio. Utilizando-se o ambiente montado através do Arduíno e a placa

[3] Este método não foi representado através de códigos devido a sua complexidade e grande tamanho

Ethernet Shield ilustrada na Figura 5.10, e considerando todo o desenvolvimento e integração feita entre as três camadas explicitadas nas subsções anteriores, resultando numa nova aplicação para o monitoramento pervasivo de ambientes, verificou-se o comportamento do Arcabouço à medida que o mesmo recebia medições.

Com o ambiente em funcionamento, diminuiu-se a temperatura na região próxima ao sensor. Depois esperou-se a temperatura medida passar de 20 ^{o}C. Quando tal evento ocorreu, e como a regra cadastrada prevê, um alerta via email foi enviado para a pessoa responsável(ver Figura 5.14). Assim, comprovou-se que a aplicação desenvolvida com o Arcabouço para este domínio funciona como o esperado, resolvendo o problema da Maria Josefina.

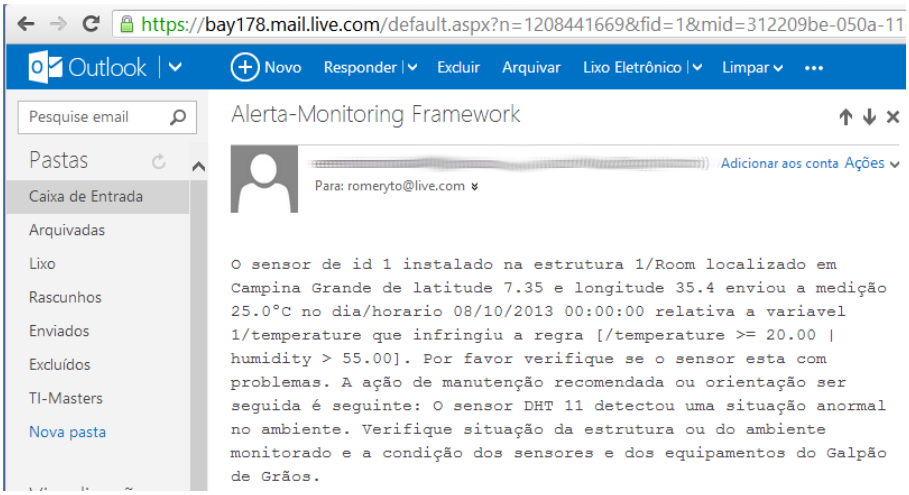


Figura 5.14: Email de alerta relativo sensor DHT 11.

5.2.3 Aplicação para o monitoramento pervasivo de ambientes através de medições de distância com sensor ultrassônico

O Monitoramento de medições de distância através de sensores ultrassônicos, visando manutenção de equipamentos sejam industriais são cada vez mais necessários, e estão em pleno uso no mercado. Os mesmos são usados para monitoramento de turbinas, válvulas, compressores, tubos, entre outros equipamentos [34]. Baseando-se nestes aspectos, um cenário de aplicação da solução desenvolvida para o domínio industrial é apresentado a seguir:

"Paula necessita monitorar a distância e posição que um determinado componente interno de uma bomba de água tem. Este componente quando em funcionamento fica em bas-

tante movimento e com o tempo vai folgando (pode ser um rolamento, um eixo, etc.). O que acontece é que se ele folgar ou se deslocar demais, ficará longe da posição ideal e poderá danificar o equipamento. Assim Paula teve a idéia de monitorar a distância do componente. Ou seja, quando esta distância aumentasse demais, Paula deveria saber previamente, para assim efetuar a manutenção e evitar problemas posteriores."

A forma de monitorar este ambiente da Paula pode ser através do monitoramento da distância do componente até um determinado referencial. Ou seja, pode-se instalar um sensor ultrassônico preso na bomba d'água e apontado para o componente, de modo que se ele sair da distância especificada, um alerta de disparo para ação de manutenção seja enviado prontamente. Com base neste caso, desenvolveu-se nesta pesquisa uma aplicação para monitoramento pervasivo deste tipo de ambiente, usando o Arcabouço.

Esta aplicação tem a seguinte arquitetura expressada na Figura 5.15, para fins de melhor entendimento e auxílio às subseções que explicam cada camada.

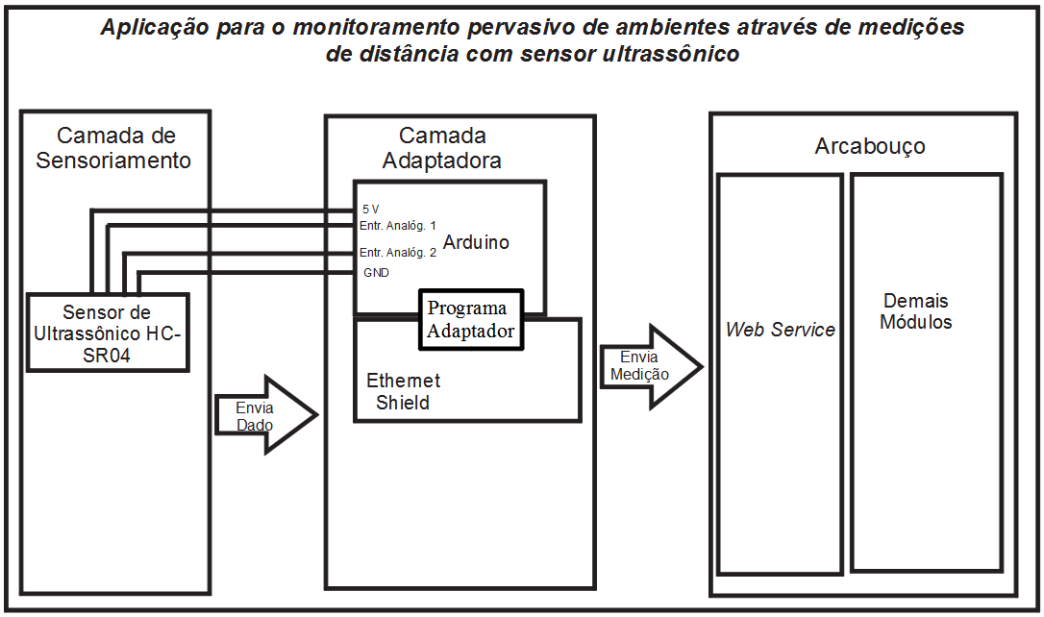


Figura 5.15: Arquitetura da aplicação para o monitoramento pervasivo de ambientes através de medições de distância usando sensor ultrassônico.

Camada de Sensoriamento

Nesta camada de sensores montou-se um ambiente simples para simular tal caso. O sensor utilizado foi o *Sensor Ultrassônico HC-SR04* para Arduíno retratado na Figura 5.16. Este

sensor opera na escala dos "cm", sendo assim ideal para esta aplicação.

Figura 5.16: Sensor ultrassônico HC-SR04.

Camada do Arcabouço

Semelhantemente às duas aplicações anteriores, os dados de entrada para o Arcabouço foram cadastrados através da interface gráfica. Na Tabela 5.3 apresente estes dados. Já na Figura 5.17 é ilustrada a regra criada para para o sensor ultrassônico HC-SR04 desta aplicação.

Figura 5.17: Tela que exibe a regra criada no Arcabouço para o sensor ultrassônico HC-SR04.

Tabela 5.3: Dados informados ao Arcabouço na aplicação de monitoramento de medições distância.

Entidade	**Dados de Entrada**
Localização	`location(id: 1, name:"Campina Grande", latitude:-7.23", longitude:-35.88 ")`
Estrutura	`structure(id: 1, name:"Bomba d'água", domain:"Industrial", location: location)`
Sensor	`sensor(id:1, name: "Sensor UT HC-SR04", structure: structure)`
Variáveis	`variable1(alias:"distance", sensor:1, vartype:(type: "double", unit: "cm"))`
Regra	`firingRule(id: 1, completeRule: "distance > 30", alertType: "email", maintenanceAction:maintenanceAction)`
Ação de Manutenção	`maintenanceAction(description:"Verificar o ocorrido", personToBeNotified( name:"Romeryto", email:"romeryto@live.com", telefone:"558388888888"))`

A partir deste momento, com o Arcabouço devidamente configurado, o próximo passo foi o trabalho relativo a sua integração com a *Camada Adaptadora*.

Camada Adaptadora

A *Camada Adaptadora* desta aplicação foi desenvolvida com o objetivo de enviar as medições de distância coletadas pelo sensor ultrassônico HC-SR04. Tal adaptador para o sensor escolhido também foi desenvolvido usando a própria linguagem de programação do Arduíno. Para esta validação foi montado um ambiente que é ilustrado na Figura 5.18.

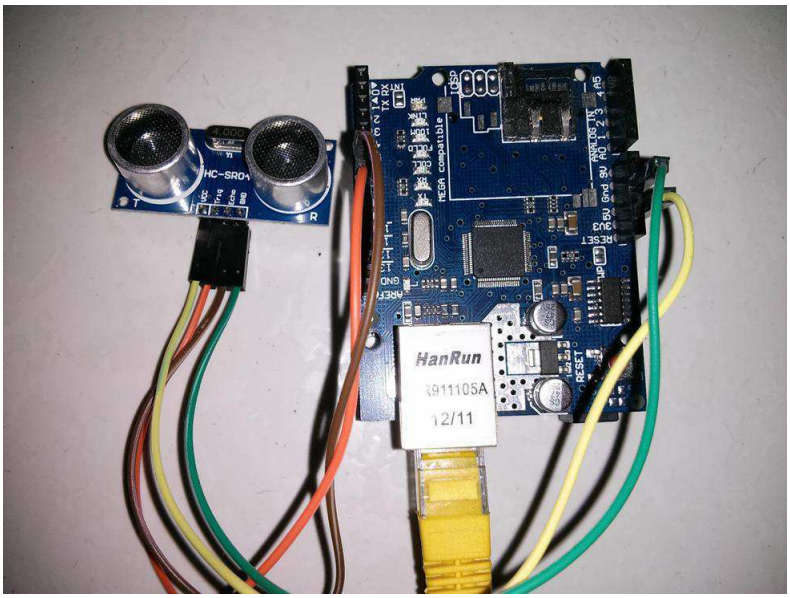


Figura 5.18: Ambiente montado com o Arduíno, *EthernetShield* e sensor ultrassônico HC-SR04.

O desenvolvimento desta camada foi dividido em duas partes: Código Coletor de Medições(`getUTMeasurement`) 5.4, e o Código 5.5 que se conecta e envia a medição coletada do sensor para o Arcabouço, através da placa *EthernetShield*.

Código Fonte 5.4: Código coletor de medições do sensor ultrassônico plugado ao Arduíno

```
//Código que coleta a medição do ultrassônico
char* getUTMeasurement(){
 /* As variáveis trigPin/echoPin são usadas para determinar a distância(
    em "cm"
 do sensor até um objeto mais próximo, a partir de envios de ondas
    ultrassônicas*/
 digitalWrite(trigPin, LOW);
 delayMicroseconds(2);
 digitalWrite(trigPin, HIGH);
 delayMicroseconds(10);

 digitalWrite(trigPin, LOW);
 duration = pulseIn(echoPin, HIGH);
 //Calcula a distância (em cm) baseado na velocidade do som.
 distance = duration/58.2;
```

```
 return distance;
}
```

Código Fonte 5.5: Código que integra a *Camada de Sensoriamento* ao Arcabouço através do envio de medições de distância

```
//Código que conecta-se e envia a medição para o Arcabouço
void sendUTMeasurement(char* utMeasurement){
  if (client.connect("http://localhost:8080/MonitoringFramework/
     monitoringFrameworkService/", 8080)) {
    Serial.println("connected");
    // Faz a requisição:
    client.println("POST addMeasurement?idVariable="+
       variableIDOfMonitoringFramework +"&value="+utMeasurement);
    client.println("http://localhost:8080");
    client.println("Connection: close");
    client.println();
  }
  else {
    Serial.println("connection failed");
  }
}
```

Em linguagem Arduíno, através do método `sendUTMeasurement`, este programa adaptador conecta-se ao *Web Service* do Arcabouço e o envia a medição de distância (obtida e calculada no método `getUTMeasurement()`), fazendo uma requisição *POST* ao serviço `addMeasurement`. Em termos arquiteturais (Figura 5.15), a medição é coletada da *Camada de Sensores*, enviada ao Arduíno e, através da placa *EthernetShield*, é enviada ao Arcabouço.

Aplicação em funcionamento

Utilizando-se o ambiente montado através do Arduíno e a placa *EthernetShield* ilustrada na Figura 5.15, e considerando todo o desenvolvimento e integração feita entre as três camadas explicitadas nas subsções anteriores, resultando numa nova aplicação para o monitoramento pervasivo de ambientes, verificou-se o comportamento do Arcabouço à medida que o mesmo recebia medições de distância do sensor ultrassônico.

Com o ambiente em funcionamento, escolheu-se um referencial plano para apontar o sensor, e a distância foi sendo aumentada até ultrapassar os 30 cm. Quando tal evento ocorreu, e como a regra cadastrada prevê, um alerta de email enviado para a pessoa responsável (ver Figura 5.19). Assim, comprovou-se que a aplicação desenvolvida com o Arcabouço para este domínio funciona como o esperado, resolvendo o problema da Paula.

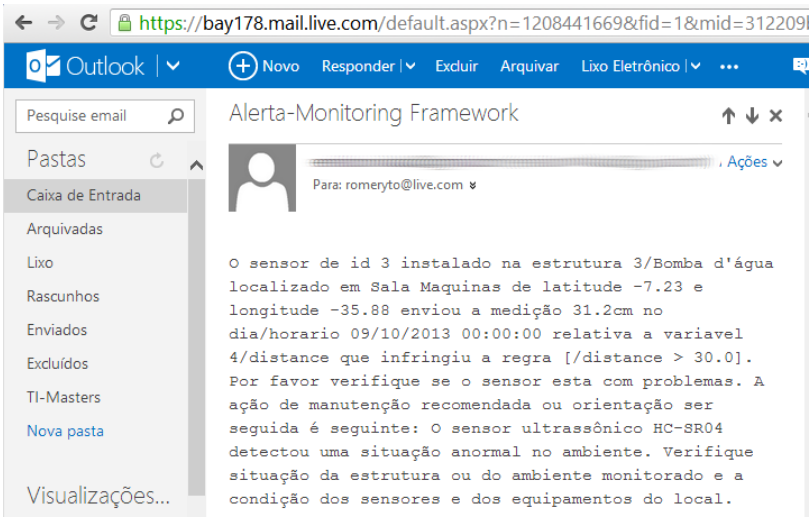


Figura 5.19: Email de alerta relativo ao monitoramento de medições de distância do sensor ultrassônico.

5.3 Conclusão

Neste capítulo foi apresentado um mecanismo de validação do Arcabouço desenvolvido expondo o desenvolvimento de três estudos de caso cada um com seu próprio domínio. O objetivo desses estudos de caso foi demonstrar como o Arcabouço pode ser utilizado na criação de aplicações para o monitoramento pervasivo de ambientes, independente do domínio em questão, bastando usar as ferramentas e funcionalidades fornecidas pelo mesmo.

Capítulo 6

Considerações Finais

Atualmente existem diversos trabalhos na área do monitoramento pervasivo de ambientes tendo como base a manutenção preventiva em suas infraestruturas implantadas. Embora as soluções propostas nestes trabalhos resolvam total ou parcialmente os problemas para cada domínio em separado, estas se aplicam somente ao domínio em questão e não dão suporte para outros domínios distintos, ou seja, mesmo os trabalhos tendo características arquiteturais comuns entre si, todas as funcionalidades semelhantes tiveram que ser recriadas para cada uma destas aplicações, provocando retrabalho desnecessário.

A contribuição deste trabalho é um Arcabouço onde o desenvolvedor poderá criar aplicações para monitoramento pervasivo de ambientes independente do domínio em questão, visto que as particularidades arquiteturais e funcionalidades semelhantes entre os vários domínios existentes são parte da solução aqui apresentada, ou seja, o Arcabouço esconde do desenvolvedor grande parte da complexidade existente na construção de aplicações deste tipo, evitando assim retrabalho e perda de tempo que antes existiam. Para isto o desenvolvedor necessita apenas desenvolver um adaptador, que efetue a coleta de medições do sensor escolhido e as envie para os serviços do Arcabouço, visto que não é necessário se preocupar com o desenvolvimento de funcionalidades relativas às regras de disparo para ações de manutenção, persistência e gerenciamento de medições, nem com o envio de alertas, já que todas estas funcionalidades já são entregues prontas para o uso.

Com o objetivo de validar o trabalho, desenvolveu-se três estudos de caso utilizando o Arcabouço, cada qual em um domínio diferente. O primeiro estudo de caso desenvolvido para o domínio da Energia Elétrica teve como objetivo a criação de uma aplicação para o

monitoramento pervasivo de ambientes através de medições de corrente elétrica. A partir do sensor ACS712 o adaptador envia ao Arcabouço medições de corrente que podem variadas através de um circuito elétrico montado com uma fonte atx e um conjunto de duas lâmpadas. Já o segundo estudo de caso foi desenvolvido para o domínio agrícola e teve como objetivo a criação de uma aplicação para o monitoramento pervasivo de ambientes através de medições de temperatura e umidade relativa do ar. Através do sensor DHT 11 o adaptador responsável envia as medições de temperatura e umidade coletadas ao Arcabouço. Por fim, o terceiro estudo de caso foi desenvolvido para o domínio industrial e teve como objetivo a criação de uma aplicação para o monitoramento pervasivo de ambientes através de medições de distância com sensor ultrassônico. Através do sensor HC-SR04, o adaptador correspondente envia as medições de distância para o Arcabouço.

Para todas as aplicações o Arcabouço recebe a medição, armazena-a e verifica se as regras previamente cadastradas são violadas. Em caso positivo, um alerta (SMS, Email ou *Web Service*) para ação de manutenção é enviado ao atuador responsável. Assim, comprovou-se que cada aplicação desenvolvida com o Arcabouço funciona como o esperado, resolvendo os problemas mostrados nos estudos de caso, para cada domínio em questão.

Uma das limitações deste trabalho está no fato do mesmo não oferecer suporte a variáveis mais complexas (como matrizes), visto que a maioria absoluta dos sensores para monitoramento usam variáveis simples como um simples número decimal. Como trabalho futuro, pode-se implementar esta nova funcionalidade, facilitando mais ainda o trabalho do desenvolvedor.

Outro trabalho futuro é o melhoramento do módulo de regras, de modo a dar suporte à criação de regras mais complexas que considerem o uso de funções matemáticas, visto que em vários domínios o estudo do valor de uma medição não é feito de forma atômica, e sim a partir de um conjunto de operações matemáticas que transformem tal medição numa informação que possa ser usada como paramêtro seguro de decisão.

Bibliografia

[1] Isabelle Alvarez and Sylvie Huet. Automatic diagnosis of engine of agricultural tractors: The bed experiment. *Biosystems Engineering*, 100(3):362–369, July 2008.

[2] Ricardo de A. Araújo, Adriano L. I. Oliveira, and Sergio Soares. A shift-invariant morphological system for software development cost estimation. *Expert Systems with Applications*, 38(4):4162–4168, April 2011.

[3] Arduino.cc. Arduino ethernet shield specification. `http://arduino.cc/en/Main/ArduinoEthernetShield`, July 2013. [Online; accessed July-2013].

[4] Arduino.cc. Official arduino website. `http://arduino.cc`, July 2013. [Online; accessed July-2013].

[5] M. Bocciolone, A. Caprioli, A. Cigada, and A. Collina. A measurement system for quick rail inspection and effective track maintenance strategy. *Mechanical Systems and Signal Processing*, 21(3):1242–1254, April 2007.

[6] P. Chardsutthi, K. Achariyasombat, and S. Adsavakulchai. E-training for private bus preventive maintenance. *2010 International Conference on Education and Management Technology*, pages 524–527, November 2010.

[7] A.M. Cheriyan, Z. Kalbarczyk, R.K. Iyer, A.O. Jarvi, T.M. Gallagher, and K.L. Watkin. Pervasive embedded real time monitoring of eeg and spo2. In *Pervasive Computing Technologies for Healthcare, 2009. PervasiveHealth 2009. 3rd International Conference on*, pages 1–4, 2009.

[8] Soonhwang Choi, Sooyong Park, and Vijayan Sugumaran. A rule-based approach for

estimating software development cost using function point and goal and scenario based requirements. *Expert Systems with Applications*, 39(1):406–418, January 2012.

[9] Tore Dyba and Torgeir Dingsoyr. Empirical studies of agile software development: A systematic review. *Information and Software Technology*, 50(9-10):833–859, August 2008.

[10] M. Entezami, S. Hillmansen, P. Weston, and M.Ph. Papaelias. Fault detection and diagnosis within a wind turbine mechanical braking system using condition monitoring. *Renewable Energy*, 47:175–182, November 2012.

[11] Rebiha Fourar-Belaifa, Francis Fleurat-Lessard, and Zouaoui Bouznad. A systemic approach to qualitative changes in the stored-wheat ecosystem: Prediction of deterioration risks in unsafe storage conditions in relation to relative humidity level, infestation by sitophilus oryzae (l.), and wheat variety. *Journal of Stored Products Research*, 47(1):98–105, January 2011.

[12] Luis A. García-Escudero, Oscar Duque-Perez, Daniel Morinigo-Sotelo, and Marcelo Perez-Alonso. Robust condition monitoring for early detection of broken rotor bars in induction motors. *Expert Systems with Applications*, 38(3):2653–2660, March 2011.

[13] Varuna Godara. *Strategic Pervasive Computing Applications: Emerging Trends*. Information Science Reference - Imprint of: IGI Publishing, 2010.

[14] Network Working Group. Internet message format. `http://tools.ietf.org/html/rfc2822`, April 2001.

[15] W3C Working Group. Web services architecture. `http://www.w3.org/TR/ws-arch`, February 2004.

[16] Randolph W. Hall. Scheduling and facility design for transit railcar maintenance. *Transportation Research Part A: Policy and Practice*, 34(2):67–84, February 2000.

[17] Erich Hau. *Wind turbines: fundamentals, technologies, application, economics*. Springer, 2006.

[18] Chin-Yu Huang and Jung-Hua Lo. Optimal resource allocation for cost and reliability of modular software systems in the testing phase. *Journal of Systems and Software*, 79(5):653–664, May 2006.

[19] Internet Engineering Task Force IETF. Sms-short message service. `http://www.ietf.org/rfc/rfc5724.txt`, January 2010.

[20] Jian-guang Jia, Zun-wen He, Jing-ming Kuang, and Hong-bo Yan. Application of wireless sensor network in monitor system for grain depots. In *Proceedings of the 5th International Conference on Wireless communications, networking and mobile computing*, pages 1–5. IEEE, 2009.

[21] Jianguang Jia, Jingming Kuang, Zunwen He, and Mu. Mechatronics and automation, 2009. icma 2009. international conference on. In *Design of monitor system for grain depots based on wireless sensor network*, pages 2318–2323. IEEE, 2009.

[22] Chinmaya Kar and A. R. Mohanty. Monitoring gear vibrations through motor current signature analysis and wavelet transform. *Mechanical Systems and Signal Processing*, 20(1):158–187, January 2006.

[23] Nicole J. Kessissoglou. Integrating vibration and oil analysis for machine condition monitoring. *Practicing Oil Analysis Magazine*, 5:98–105, 2003.

[24] R. Khodabakhshian and M. Shakeri. Prediction of repair and maintenance costs of farm tractors by using of preventive maintenance. *International Journal of Agriculture Sciences*, 3(1):39–44, 2011.

[25] Young-Duk Kim, Yeon-Mo Yang, Won-Seok Kang, and Dong-Kyun Kim. On the design of beacon based wireless sensor network for agricultural emergency monitoring systems. *Computer Standards & Interfaces*, pages 1–12, May 2011.

[26] Naoto Kume and Mikko J. Rissanen. Towards ambient communication support for power grid maintenance and repair. *Procedia Computer Science*, 5:98–105, January 2011.

[27] Anita Lee, Chun Hung Cheng, and Jaydeep Balakrishnan. Software development cost estimation: Integrating neural network with cluster analysis. *Information & Management*, 34(1):1–9, August 1998.

[28] Li Lijuan and Minchai Hao. The mathematical model of food storage safety monitoring and control system. In *Computer Application and System Modeling (ICCASM), 2010 International Conference on*, volume 13, pages V13–591–V13–594. Ieee, October 2010.

[29] Changyu Liu, B. Iung, Luqing Ye, and G. Morel. An innovative hydraulic turbine governor for predictive maintenance of hydropower plant. In *Decision and Control, 2002, Proceedings of the 41st IEEE Conference on*, volume 265, pages 925–930. New York, 2002.

[30] Emerson Loureiro, Glauber Ferreira, Hyggo Almeida, and Angelo Perkusich. Pervasive computing: What is it anyway? *Ubiquitous and Pervasive Knowledge and Learning Management: Semantics, Social Networking and New Media to Their Full Potential - IGI Global*, pages 1–34, 2007.

[31] E. Martinez, F. Sanz, S. Pellegrini, E. Jimenez, and J. Blanco. Life cycle assessment of a multi-megawatt wind turbine. *Renewable Energy*, 34(3):667–673, March 2009.

[32] Ram B. Misra. Global it outsourcing: Metrics for success of all parties. *Journal of Information Technology Cases and Applications*, 6(3):21–34, July 2004.

[33] Dept. of Electrical Engineering of University of Delaware. Rfc822: Standard for arpa internet text messages. `http://www.w3.org/Protocols/rfc822`, August 1982.

[34] Alf Püttmer. New applications for ultrasonic sensors in process industries. *Proceedings of Ultrasonics International and World Congress on Ultrasonics (WCU)*, 44, Supplement 1:e1379–83, December 2006.

[35] Inc. Pure Energy Systems Network. Wind turbines's structure. `http://peswiki.com/energy/Directory:Wind`, July 2013. [Online; accessed July-2013].

[36] Julien Rabatel, Sandra Bringay, and Pascal Poncelet. Anomaly detection in monitoring sensor data for preventive maintenance. *Expert Systems with Applications*, 38(6):7003–7015, June 2011.

[37] Raspberrypi.org. Official raspberry pi website. `http://www.raspberrypi.org/`, August 2013.

[38] Pedro Vicente Jover Rodríguez, Marian Negrea, and Antero Arkkio. A simplified scheme for induction motor condition monitoring. *Mechanical Systems and Signal Processing*, 22(5):1216–1236, July 2008.

[39] D. Saha and A. Mukherjee. Pervasive computing: a paradigm for the 21st century. *IEEE Computer, IEEE Computer Society Press*, 36(3):25–31, 2003.

[40] N. Tandon, G.S. Yadava, and K.M. Ramakrishna. A comparison of some condition monitoring techniques for the detection of defect in induction motor ball bearings. *Mechanical Systems and Signal Processing*, 21(1):244–256, January 2007.

[41] N.A. Thakkar, J.A. Steel, and R.L. Reuben. Rail-wheel interaction monitoring using acoustic emission: A laboratory study of normal rolling signals with natural rail defects. *Mechanical Systems and Signal Processing*, 24(1):256–266, January 2010.

[42] Zhigang Tian and Tongdan Jin. Maintenance of wind turbine systems under continuous monitoring. In *Reliability and Maintainability Symposium (RAMS), 2011 Proceedings - Annual*, volume 1, pages 1–6. IEEE, 2011.

[43] Zhigang Tian, Tongdan Jin, Bairong Wu, and Fangfang Ding. Condition based maintenance optimization for wind power generation systems under continuous monitoring. *Renewable Energy*, 36(5):1502–1509, May 2011.

[44] Sameer Tyagi. Restful web services. `http://www.oracle.com/technetwork/articles/javase/index-137171.html`, August 2006.

[45] Mohammed F. Wahby and Saleh A. AL-Suhaibani. Repair and maintenance cost models for agricultural equipment in saudi arabia. *American Society of Agricultural and Biological Engineers*, page 7, 2002.

[46] Mark Weiser. The computer for the 21st century. *Scientific American*, 265(3):94–104, January 1991.

[47] Pingle Yang, Yalei Yang, and Yunfeng Lou. A business activity real-time monitoring platform based on rule engine. *Procedia Engineering - CEIS 2011*, 15:3744–3748, January 2011.

[48] B Yoshimi. On sensor frameworks for pervasive systems. *Proceedings of the Workshop on Software Engineering for Wearable and Pervasive Computing at the 22nd International Conference on Software Engineering*, 5:98–105, 2000.

[49] Jiehan Zhou, E. Gilman, M. Ylianttila, and J. Riekki. Pervasive service computing: Visions and challenges. In *Computer and Information Technology (CIT), 2010 IEEE 10th International Conference on*, number Cit, pages 1335–1339. Ieee, June 2010.

[50] H.J. Zimmermann. *Fuzzy Set Theory-And Its Applications*. Kluwer Academic Publishers, 2001.