INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA SUL-RIO-GRANDENSE - IFSUL, CAMPUS PASSO FUNDO
CURSO DE TECNOLOGIA EM SISTEMAS PARA INTERNET

GUILHERME ANTONIO BORGES

ARQUITETURA UBÍQUA PARA AMBIENTES RESIDENCIAIS

Profa. Me. Anubis Graciela De Moraes Rossetto

PASSO FUNDO, 2012

GUILHERME ANTONIO BORGES

ARQUITETURA UBÍQUA PARA AMBIENTES RESIDENCIAIS

Monografia apresentada ao Curso de Tecnologia em Sistemas para Internet do Instituto Federal Sul-Rio-Grandense, *Campus* Passo Fundo, como requisito parcial para a obtenção do título de Tecnólogo em Sistemas para Internet.

Orientador (a): Profa. Msc. Anubis Graciela De Moraes Rossetto

PASSO FUNDO, 2012

GUILHERME ANTONIO BORGES

ARQUITETURA UBÍQUA PARA AMBIENTES RESIDENCIAIS

Trabalho de Conclusão de Curso aprovado em ____/____/____ como requisito parcial para a obtenção do título de Tecnólogo em Sistemas para Internet

Banca Examinadora:

_____

Me. Anubis Graciela De Moraes Rossetto

_____

Esp. José Antônio Oliveira de Figueiredo

_____

Me. Élder Francisco Fontana Bernardi

_____

Me. Evandro Miguel Kuszera
(Coordenador do Curso)

PASSO FUNDO, 2012

Aos meus pais e ao meu irmão,
pelo apoio e compreensão
em todos os momentos.

AGRADECIMENTOS

Primeiramente gostaria de agradecer a Deus, por ter me guiado nesta caminhada.

Agradecendo aos meus pais, Carlos e Verone, e ao meu irmão, Lucas, por estarem presentes sempre em minha vida. Prometo que não vou deixa-los de lado, buscarei meus sonhos, mas trarei partes das minhas conquistas para junto de vocês! Amo vocês.

Agradeço ao meu bom e velho amigo Vinícius P. Lima, cuja presença somente engrandeceu meu espirito, cuja amizade só me trouce alegria, quero que saiba que você foi como um irmão mais velho. Também tenho a agradecer pelos meus amigos e irmãos Escoteiros, pelas inesquecíveis aventuras, experiências e por me mostrarem que mesmo tendo iniciativa, ou vontade de mudar o mundo, somente unidos somos fortes o suficiente para realizá-los, tudo isso sem nunca esquecer os verdadeiros motivos que nos mantiveram unidos: a amizade, o companheirismo e a fidelidade.

Agradeço aos meus amados colegas de faculdade pelas incríveis experiências e situações que estamos enfrentando, juntos. A minha orientadora Anubis pela confiança depositada, pelo excelente trabalho e dedicação. Aos demais professores que marcaram a minha vida de estudante.

Agradeço por toda a minha família, aos amigos que por diversos motivos tomamos caminhos diferentes, aos professores e colegas de musica, de artes marciais, das pessoas que conheci na Igreja, e o pessoal da Parceria. Cada um contribuiu um pouco para o que sou hoje, e sou muito grato por isso.

Por fim, existem pessoas que não estão mais junto de nós, mas que fizeram contribuições importantes para meu crescimento como pessoa e como estudante, dentre eles um professor que respeito e admiro muito deixou uma frase que por diversas vezes me inspirou: “Tudo o que fizermos com amor será bem feito” - Juliano Menegaz.

“Deixe o mundo um pouco melhor do que encontrou.”

Robert Stephenson Smyth Baden-Powell

RESUMO

Este trabalho apresenta uma arquitetura de software para o desenvolvimento de aplicações distribuídas ubíquas aplicadas ao ambiente residencial, tendo foco na solução de dois aspectos críticos destes tipos de aplicações: tratar a heterogeneidade dos diversos tipos de hardware existentes, sem necessitar reprogramar totalmente, e a utilização de uma forma de comunicação eficiente nos aspectos da heterogeneidade e eficaz na sua execução, capaz de tornar ubíqua a interação com o ambiente residencial. Assim, esse trabalho tem como objetivo propor uma arquitetura para sistemas ubíquos de ambientes residenciais que seja dinâmica e simples, com vistas a facilitar a adaptação para diferentes cenários. Serão apresentados o modelo conceitual da arquitetura proposta e os resultados obtidos com o protótipo desenvolvido.

Palavras-chave: Automação Residencial, Sistemas Distribuídos, Computação Ubíqua.

ABSTRACT

This work presents a software architecture for development of ubiquitous distributed application, applied to home environment, with focus on solving two critical aspects of these kind of applications: dealing with heterogeneity of various types of existing hardware, without need to completely rewrite a program, and use of a being efficient and effective communication with aspects of heterogeneity, able of making ubiquitous interaction with a home environment. Thus, this study aims to propose an architecture for dynamic and simple ubiquitous systems for home environments that to facilitate adaptation to different scenarios. What will be presented is the conceptual model of the proposed architecture and results obtained with the prototype.

Keywords: Home Automation, Distributed Systems, Ubiquitous Computing.

LISTA DE TABELAS

Tabela 1 - Diferentes formas de transparência em um sistema distribuído 17
Tabela 2 - Exemplos das características de microcontroladores nas versões 4, 8, 16 e 32 bits 26
Tabela 3 - Envelope de Requisição de Ação 44
Tabela 4 - Envelope de Requisição de Resposta 45
Tabela 5 - Referências do Protocolo de Interação contém 47
Tabela 6 - Permissões sobre o Serviço 52
Tabela 7 - Permissões sobre a Representação Lógica 52
Tabela 8 - Ações que a Comunicação Local Física do Protótipo pode executar 58

LISTA DE FIGURAS

Figura 1 - Relação Computação Móvel, Pervasiva e Ubíqua. 20
Figura 2 - Demonstração da comunicação de uma aplicação utilizando Web Service SOAP. 25
Figura 3 - Microcontrolador ATmega238P, da Atmel 27
Figura 4 - Arduino Duemilanove 28
Figura 5 - Shield Ethernet 28
Figura 6 - Exemplo de componentes internos e externos do ambiente residencial 31
Figura 7 - Representação conceitual das Camadas da Arquitetura do UBGEAR. 32
Figura 8 - Funcionamento da Comunicação Local Física 34
Figura 9 - Atividades das funcionalidades de Controle Físico 35
Figura 10 - Processo de interação entre as camadas intermediária e física 37
Figura 11 - Modelo Completo dos Dados XML 39
Figura 12 - Modelo de Coleta Completa de Dados 40
Figura 13 - Modelo de Coleta Simplificada de Dados 40
Figura 14 - Modelo de Coleta do Mapa de Componentes 41
Figura 15 - Modelo de Coleta de Configurações 42
Figura 16 - Comunicação Local do Servidor, processo ao receber dados da Camada Física. . 46
Figura 17 - Comunicação Local do Servidor, processo ao enviar dados da Camada Física. ... 46
Figura 18 - Processo da Interação ao receber o envelope da Comunicação Local Servidor. ... 48
Figura 19 - Processo da Interação ao enviar o envelope para a Comunicação Local Servidor. 48
Figura 20 - Atividade das Representações Lógicas do módulo de Serviço para o de Abstração 49
Figura 21 - Atividade das Representações Lógicas do módulo de Interação para o de Abstração 50
Figura 22 - Funcionamento do módulo de Serviços 51
Figura 23 - Arquitetura Proposta 56
Figura 24 - Estrutura proposta para a Camada Física 57
Figura 25 - Classe Componente utilizada na implementação da Camada Física 59
Figura 26 - Diagrama de Objetos contendo os dados iniciais dos componentes 59
Figura 27 - Valores de referência do Protocolo de Interação no nó de controle 60
Figura 28 - Estrutura proposta da Camada Intermediária 61
Figura 29 - Diagrama de classes proposto para a Comunicação Local Servidor 62

Figura 30 - Diagrama de classes proposto para a Interação 63
Figura 31 - Protocolo Default utilizado na proposta da Interação 64
Figura 32 - Diagrama de classes proposto para a Abstração 65
Figura 33 - Diagrama de classes proposto para o módulo de Serviço 66
Figura 34 - Diagrama de classes dos módulos de Usuários e de Comunicação Aplicação Servidor 67
Figura 35 - Estrutura proposta para a Camada Cliente 67
Figura 36 - Diagrama de classes Completo da Camada Intermediária 75
Figura 37 – XML contendo as configurações do nó de controle, pelo comando /c 76
Figura 38 – XML do Mapa de componentes presentes no nó de controle, pelo comando /m . 76
Figura 39 – XML resultado da coleta completa de dados do Componente 1, pelo comando /11ga 77
Figura 40 - XML resultado da coleta simplificada de dados do Componente 1, pelo comando /11gs 77
Figura 41 - XML resultado da coleta completa de dados do Componente 2, pelo comando /12ga 78
Figura 42 - XML resultado da coleta simplificada de dados do Componente 2, pelo comando /12gs 78
Figura 43 - XML resultado da coleta completa de dados do Componente 3, pelo comando /13ga 79
Figura 44 - XML resultado da coleta simplificada de dados do Componente 3, pelo comando /13gs 79

LISTA DE ABREVIATURAS E SIGLAS

ANSII - *American Standard Code for Information Interchange*, p. 68

CLF - Comunicação Local Física, p. 55

E/S – Entrada e Saída, p. 26

GHz – *Giga Hertz*, p. 68

GPS - *Global Positioning System*, p. 26

HTML - *HyperText Markup Language*, p. 24

IDE - *Integrated Development Environment*, p. 29

KB – *Kilo Byte*, p. 26

LAN - *Local-Area Network*, p. 16

LCD - *Liquid Crystal Display*, p. 26

LED - *Light-Emitting Diode*, p. 54

MAC - *Media Access Control*, p. 43

Mb - *Mega bit*, p. 63

MHz – *Mega Hertz*, p. 26

OO - Orientado a Objeto, p. 63

PC - *Personal Computer*, p. 64

PDAs - *Personal Digital Assistants*, p. 21

RFID - *Radio-Frequency Identification*, p. 33

RISC - *Reduced Instruction Set Computer*. P 28

PWM - *Pulse-Width Modulation*, p. 26

SD - *Secure Digital*, p. 35

SMS - *Short Message Service*, p. 51

SOAP - *Simple Object Access Protocol*, p. 24

UBGEAR - *Ubiquitous Generic Architecture*, p. 29

UML - *Unified Modeling Language*, p. 64

USART - *Universal Synchronous Asynchronous Receiver Transmitter.* p. 28

USB – *Universal Serial Bus*, p. 45

UTF - *Unicode Transformation Formats*, p. 34

XML - *eXtensible Markup Language*, p. 24

WAN - *Wide Area Network*. p. 30

SUMÁRIO

1 INTRODUÇÃO 13
1.1 MOTIVAÇÃO 14
1.2 OBJETIVOS 15
1.2.1 Objetivo Geral 15
1.2.2 Objetivos específicos 15

2 REFERENCIAL TEÓRICO 16
2.1 SISTEMAS DISTRIBUÍDOS 16
2.1.1 Transparência 17
2.1.2 Abertura 18
2.1.3 Escalabilidade 18
2.1.4 Concorrência 18
2.1.5 Segurança 18
2.1.6 Heterogeneidade 19
2.1.7 Tratamento de falhas 19
2.2 RELAÇÃO COMPUTAÇÃO MÓVEL, PERVASIVA E UBÍQUA 19
2.2.1 Computação móvel 20
2.2.2 Computação Pervasiva 21
2.2.3 Computação Ubíqua 21
2.3 AMBIENTES INTELIGENTES 22
2.3.1 Sistemas Domésticos 23
2.3.2 Automação Residencial 23
2.4 WEB SERVICES 24
2.5 MICROCONTROLADORES 25
2.5.1 Arduino 27

3 PROJETO UBGEAR 29
3.1 CENÁRIO DE APLICABILIDADE 29
3.2 ARQUITETURA CONCEITUAL 31
3.3 CAMADA FÍSICA 33
3.3.1 Comunicação Local Física 33
3.3.2 Controle Físico 34
3.4 COMUNICAÇÃO LOCAL ENTRE CLIENTE E SERVIDOR 36

3.4.1 Físico para Servidor 38
3.4.2 Servidor para Físico 44
3.5 CAMADA DE NÍVEL INTERMEDIÁRIO 45
3.5.1 Comunicação local Servidor 45
3.5.2 Interação 46
3.5.3 Abstração 49
3.5.4 Serviços 50
3.5.5 Usuários 51
3.5.6 Comunicação Aplicação Servidor 53
3.6 COMUNICAÇÃO DE APLICAÇÃO ENTRE CLIENTE E SERVIDOR 53
3.7 CAMADA CLIENTE 53
3.7.1 Comunicação Aplicação Cliente 53
3.7.2 Aplicação 53

4 DESENVOLVIMENTO DO PRÓTOTIPO 54
4.1 FERRAMENTAS UTILIZADOS 54
4.2 MATERIAIS UTILIZADOS 54
4.3 ARQUITETURA PROPOSTA 55
4.3.1 Camada Física 56
4.3.2 Camada Intermediária 61
4.3.3 Camada Cliente 67
4.3.4 Considerações sobre o desenvolvimento 68
4.3.5 Dificuldades enfrentadas 70

CONSIDERAÇÕES FINAIS 72

REFERÊNCIAS 73

ANEXOS E APÊNDICES 75

1 INTRODUÇÃO

Construir e viver em casas inteligentes não é um conceito novo, estando vivo há muito tempo na imaginação das pessoas. Apesar disso, somente recentemente o nível tecnológico chegou a um ponto satisfatório e economicamente viável para ser investido com sucesso no ambiente residencial. Resultado não somente da evolução do hardware ou do software, e sim de ambos. Enquanto o hardware faz a parte física, exibe o resultado a seus usuários e coleta dados do ambiente, o software faz a interação de todo esse hardware com o ambiente e o mundo, além de controlar as ações mecânicas e salvar os dados que são exibidos e coletados. Essa união vai rumo ao futuro, pois há muito que evoluir até chegar ao que WEISER (1991) afirmou: "estamos rumo à terceira era da computação, quando a tecnologia recua para o fundo de nossas vidas", a era da computação ubíqua.

Esses avanços tecnológicos associados ao novo contexto socioeconômico e ambiental do século XXI revelam inúmeras oportunidades para o desenvolvimento de sistemas inteligentes para o ambiente residencial. A integração desses sistemas tem mostrado um aumento considerável dos benefícios se comparados com os sistemas isolados, de eficiência limitada (BOLZANI, 2010, p. 16).

Já possuímos recursos físicos suficientes para construir tais ambientes, contudo a maioria desses recursos não tem uma forma padronizada e comum de comunicação, devido ao fato de que cada fabricante produz seus padrões. Os autores ARAÚJO e SIQUEIRA (2007) reforçam que essa falta de padrões comuns de comunicação e de representação de dados torna difícil e muitas vezes inviável a interação entre eles, possuindo, como BOLZANI citou no trecho acima: "eficiência limitada". Cabem aos softwares unir todas essas tecnologias, tornando-as mais eficientes, adicionando novas possibilidades e conectando-as ao mundo.

Por outro lado a indústria ainda é a principal utilizadora de sistemas automatizados. Os utiliza para facilitar seus processos, o que exige um alto custo de investimento. Também por sua influência muitos trabalhos de pesquisa sobre automação são focados na indústria, e esses conhecimentos também podem ser em grande parte aplicáveis na automação residencial. Contudo a viabilidade da implantação desses sistemas em residências era até pouco tempo "um mercado para poucos e abastados", como afirma BOLZANI (2010, p. 16).

No processo de pesquisa sobre arquiteturas, com a finalidade de tornar um sistema de automação residencial ubíquo, verificou-se, de maneira geral, uma falta de padronização e poucas arquiteturas de referência, quando encontradas demostraram ser teóricas e genéricas, ou ainda complexas em sua estrutura conceitual.

Baseando-se nestes elementos, este trabalho propõe o projeto de uma arquitetura genérica para sistemas ubíquos, buscando aplicá-la em ambientes residenciais, a fim de tratar a heterogeneidade dos muitos tipos de hardware existentes, sem precisar reprogramar todo o código fonte do sistema novamente e, a utilização de uma forma de comunicação eficiente no aspecto da heterogeneidade e eficaz na sua utilização, capaz de tornar a interação com o ambiente residencial ubíquo. Com isso, será oportunizado que os seus usuários possam acessar as informações em tempo real.

O trabalho apresenta em seu segundo capítulo os paradigmas e características dos temas abordados, através de consulta das principais literaturas da área, compondo o conhecimento inicial e necessário para o trabalho. Em seu terceiro capítulo aborda a arquitetura conceitual proposta pelo autor, composta por três camadas principais: físico, intermediário e cliente; representando respectivamente sistema embarcado, middleware e interface. No quarto capítulo será apresentado o desenvolvimento do protótipo e os resultados do protótipo desenvolvido, onde o middleware interage com os diversos sistemas embarcados e exibe os dados oriundos dessa interação para a interface, que por sua vez pode também enviar ordens para os sistemas embarcados desde que este possa executar alguma ação além de monitorar.

1.1 MOTIVAÇÃO

Hoje, ocorre uma considerável redução dos custos associados a sistemas embarcados. Esse fato aliado à grande diversificação de recursos em que eles podem interagir, abre um cenário favorável ao estudo de softwares para sistemas residenciais, que compõem uma rica área de pesquisa onde ainda há muito que se explorar.

Entre os maiores desafios desses softwares gerenciadores de hardware estão: tratar a heterogeneidade dos muitos tipos de hardware existentes sem precisar reprogramar todo o código fonte do sistema novamente e, a utilização de uma forma de comunicação eficiente e eficaz, capaz de tornar a interação com o ambiente residencial ubíquo. Bem empregados, esses softwares elevarão o conforto e a segurança da residência a um nível totalmente novo.

Desta forma, projetar uma arquitetura que aproveite essa redução de custo dos sistemas embarcados e procure tratar os desafios citados acima, formará uma base para que futuros trabalhos, tanto acadêmicos como profissionais, possam vir a utilizá-la.

1.2 OBJETIVOS

Esse trabalho tem como objetivo propor uma arquitetura para sistemas ubíquos para ambientes residenciais que seja dinâmica e simples, com vista a facilitar a adaptação para outros cenários.

1.2.1 Objetivo Geral

- Projetar uma arquitetura de sistema ubíquo aplicada em ambientes residenciais.

1.2.2 Objetivos específicos

- Estudar as principais referências da literatura sobre sistemas distribuídos;
- Propor um modelo conceitual para a arquitetura;
- Implementar o protótipo;
- Analisar os resultados do protótipo;
- Abrir outras áreas de pesquisas para novos projetos.

2 REFERENCIAL TEÓRICO

Neste capítulo definimos e delimitamos os paradigmas e características dos temas abordados, através de consulta das principais literaturas da área, compondo o conhecimento inicial e necessário para o trabalho.

2.1 SISTEMAS DISTRIBUÍDOS

A quantidade de melhorias tecnológicas que ocorreram nos últimos 50 anos podem ser dividas em dois principais eventos: o primeiro foi o desenvolvimento de microprocessadores de grande capacidade; o segundo foi a invenção das redes de computadores de alta velocidade, as redes locais, ou LANs (Local-area Networks). O resultado desses eventos é uma explosão de novas possibilidades e melhorias, os quais tornaram fácil a montagem de sistemas de computação compostos por grandes quantidades de computadores conectados por uma rede de alta velocidade. Esses sistemas são denominados redes de computadores ou sistemas distribuídos (TANENBAUM, STEEN, 2007).

Coulouris, Dollimore e Kinderg (2007, p. 16) afirmam que “o grande motivador para construir e usar sistemas distribuídos é proveniente do desejo de compartilhar recursos”. Esse desejo é alvo de inúmeras pesquisas em todo o mundo. Por ser uma área relativamente nova, há muito que se explorar. São exemplos comuns e amplamente difundidos entre os sistemas distribuídos: a Internet, intranets e redes baseadas em dispositivos móveis. Iara Augustin em sua tese de doutorado conceitua os sistemas distribuídos da seguinte maneira:

> Uma aplicação distribuída consiste em uma coleção de componentes, distribuídos sobre vários computadores (nodos) conectados via uma rede de computadores. Esses componentes interagem uns com os outros para trocarem informações ou acessarem serviços. Esta definição se aplica a ambos: sistema distribuído fixo ou móvel (AUGUSTIN, 2004. p. 26).

A partir dessa definição, a autora apresenta mais uma característica, a de que “Sistemas distribuídos podem ser fixos ou móveis”; além dessas subcategorias, há mais duas que serão abordadas diretamente nesse projeto: a computação ubíqua e a pervasiva. As particularidades de cada uma dessas áreas serão abordadas nas próximas seções.

Podemos encontrar redes de computadores, em fábricas, em veículos, em casas, sempre compostos por diversos dispositivos computacionais tais como: dispositivos móveis;

desktops; servidores em outros continentes; PDAs (Personal Digital Assistants). Todos com o mesmo propósito de compartilhar recursos, que podem ser: impressoras; scanners; imagens; vídeos; dados climáticos e dados de cadastro, bancos de dados, independente da distância física que separam os dispositivos dentre si.

Contudo, eles possuem características que são desafiadoras aos projetistas, uma vez que não são aplicações convencionais é necessário pensar de forma diferente, como Coulouris, Dollimore e Kinderg (2007) reforçam, tratando os principais desafios: necessidade de transparência, abertura, escalabilidade, concorrência, segurança, heterogeneidade e tratamento de falhas.

2.1.1 Transparência

Uma das características mais importantes que um sistema distribuído deve conter é ocultar o fato de que seus processos e recursos estão fisicamente distribuídos por vários computadores. Assim, deve propiciar uma visão única do sistema, em suma, exibir somente o resultado do processamento para o usuário que está utilizando o sistema.

O modelo de referência de processamento distribuído aberto, ISO/IEC 10746-1 apud (TANENBAUM, STEEN, 2007), define oito tipos de transparência relacionados a eventos que constantemente ocorrem com esse tipo de sistema, os quais o projetista do sistema deve saber para tratar a fim de manter a transparência. Tanenbaum e Steen (2007, p. 3) descrevem sete desses tipos, segundo eles os mais importantes, como mostra na Tabela 1.

Tabela 1 - Diferentes formas de transparência em um sistema distribuído

Transparência	Descrição
Acesso	Diferenças na representação de dados e no modo de acesso a um recurso
Localização	Lugar em que um recurso está localizado
Migração	Recurso pode ser movido para outra localização
Recolocação	Recurso pode ser movido para outra localização enquanto em uso
Replicação	Recurso é replicado
Concorrência	Recurso pode ser compartilhado por diversos usuários concorrentes
Falha	Falha e a recuperação de um recurso

Fonte: (TABENBAUN, STEEN, 2007).

2.1.2 Abertura

Um sistema distribuído é aberto quando ele pode ser estendido e reimplementado de várias maneiras, ou seja, são sistemas extensíveis (COULOURIS; DOLLIMORE; KINDBERD, 2007). Para tanto utiliza-se interfaces e protocolos padronizados, permitindo dois fornecedores diferentes coexistirem e trabalharem em conjunto. Também é necessário que seja portável, para que a aplicação possa ser executada, sem modificação, em outro sistema distribuído (TANENBAUM., STEEN, 2007).

2.1.3 Escalabilidade

Um sistema descrito como escalável é o que permanece eficiente quando há um aumento significativo no número de recursos e no número de usuários (COULOURIS; DOLLIMORE; KINDBERD, 2007). É preferível que o software não mude quando a escala aumentar, contudo esse é um resultado difícil de conseguir.

2.1.4 Concorrência

Toda aplicação distribuída tem possibilidade de conter vários clientes, logo é necessário gerenciar a concorrência dos recursos. Cada cliente deve ser tratado separadamente sem afetar o resultado e a integridade dos dados dos outros clientes. Para isso, em um ambiente concorrente, as operações devem ser sincronizadas de tal maneira que seus dados permaneçam consistentes. Para isso se utilizada técnicas, tais como: semáforo, semáforo mutex, spin-lock e mensagens (COULOURIS; DOLLIMORE; KINDBERD, 2007).

2.1.5 Segurança

Para a maioria dos usuários as suas informações são valiosas exigindo especialmente em sistemas distribuídos dar mais atenção no quesito segurança. Coulouris, Dollimore e Kinderg (2007) afirmam que a segurança de recursos de informação têm três componentes: a) confidencialidade: proteção contra exposição de pessoas não autorizadas; b) integridade: proteção contra alteração ou dano; e c) disponibilidade: proteção contra interferência com os meios de acesso aos recursos.

2.1.6 Heterogeneidade

Um dos maiores desafios dos sistemas distribuídos é justamente como ele irá interagir com a heterogeneidade de sistemas, hardwares, linguagens e redes, visto que cada fabricante ou programador utiliza padrões e algoritmos diferenciados. Para lidar com todas essas diferenças é adicionada uma camada de software intermediária, o Middleware, como definem Coulouris, Dollimore e Kinderg (2007, p. 29):

> O termo middleware se aplica a uma camada de software que fornece uma abstração de programação, assim como o mascaramento da heterogeneidade das redes, do hardware, de sistemas operacionais e linguagens de programação subjacentes.

Esta camada extra permite que haja uma visão única do sistema, ou seja, ela é uma das responsáveis pela transparência do sistema.

2.1.7 Tratamento de falhas

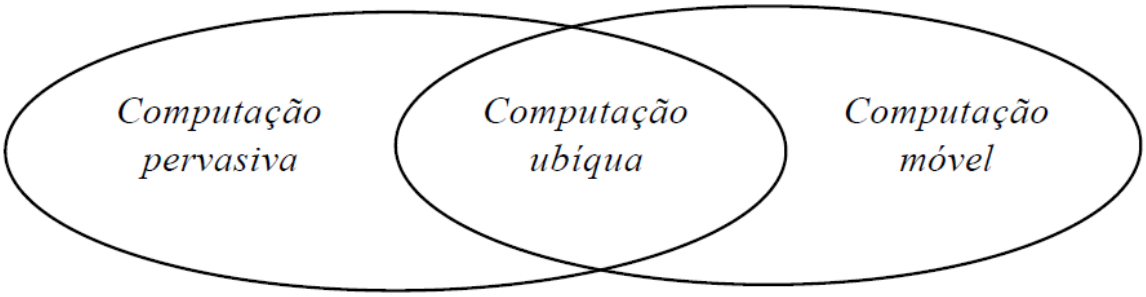
Às vezes computadores falham, seja na parte de hardware ou de software, o ideal é que de alguma forma essas falhas sejam tratadas. Para tanto são utilizadas diversas técnicas como: detecção de falhas, mascaramento de falhas, tolerância a falhas, recuperação de falhas e utilização de redundância. Contudo falhas em sistemas distribuídos são parciais, isto é, alguns componentes falham enquanto outros continuam funcionando, tornando o tratamento de falhas difícil de ser realizado (COULOURIS; DOLLIMORE; KINDBERD. 2007).

2.2 RELAÇÃO COMPUTAÇÃO MÓVEL, PERVASIVA E UBÍQUA

Enquanto a computação pervasiva se preocupa em se auto gerenciar e lidar com dispositivos embutidos no ambiente, que não são necessariamente móveis, a computação móvel trabalha com a premissa de que é possível acessar e utilizar serviços e recursos independente de estar em deslocamento ou não.

Do estudo dessas duas áreas surge a computação ubíqua, que é a relação entre computação pervasiva e móvel, ao qual une o conceito das duas áreas resultando na ideia em que o ambiente pervasivo é acessado em todo lugar a qualquer hora, passando a computação para o *background* da vida de seus usuários. A Figura 1 exibe a relação entre essas três áreas da computação.

Figura 1 - Relação Computação Móvel, Pervasiva e Ubíqua.



Fonte: ARAUJO, 2003.

2.2.1 Computação móvel

A computação móvel baseia-se no conceito de que os elementos computacionais utilizam recursos distribuídos através de conexões sem fio locais, trocando a localização durante o curso da execução, sem afetar o resultado (AUGUSTIN, 2000). Em outras palavras os usuários se deslocam fisicamente e os dispositivos computacionais o acompanham. Sendo carregados, esses aparelhos executam as mesmas funções que haviam sido programados independentes da sua localização, sem afetar o resultado do processamento durante esse deslocamento.

Diferente da computação convencional, a computação móvel possui diversas peculiaridades às quais o software e o hardware devem ser preparados para adaptar-se. Questões como a conectividade volátil, resultado da mobilidade da computação móvel, uma vez que todos os dispositivos computacionais móveis utilizam da conexão sem fio, onde casos de desconexão são triviais. Isso ocorre devido à possibilidade de saída da área de cobertura ou bloqueio do sinal por obstáculos físicos.

Além disso, devido à instabilidade das redes sem fio, diretamente influenciada pelo meio físico, a largura de banda e a latência são variáveis e, resultam em taxas de erros também instáveis, que por sua vez têm forte impacto sobre as propriedades do sistema.

> A volatilidade da conectividade – a variabilidade, em tempo de execução do estado de conexão ou desconexão entre os dispositivos e a qualidade do serviço entre eles – também tem forte impacto sobre as propriedades do sistema (COULOURIS, DOLLIMORE, KINDBERG, 2007, p. 572).

Essas características combinadas com a capacidade de acesso a redes e serviços utilizando conexões sem fio formam a computação móvel. São exemplos de dispositivos

móveis os smartphones, os tablets, os telefones móveis, os notebooks e os PDAs (*Personal Digital Assistants*).

2.2.2 Computação Pervasiva

Quando adicionados dispositivos de computação móvel e embutidos a um sistema distribuído, temos um cenário onde a instabilidade é o comportamento esperado: os sistemas distribuídos pervasivos ou a computação pervasiva. Nos quais costumam utilizar tecnologias que normalmente possuem as seguintes características: a) pequenas em tamanho; b) alimentadas por bateria; c) apresentam mobilidade; d) possuem conexão sem fio (TANENBAUM, STEEN, 2007).

A computação distribuída partilha a ideia de que o pervasivo é parte do entorno de seus proprietários nos quais o controle administrativo do sistema torna-se do próprio sistema ao invés do humano, como Tanenbaum e Steen (2007, p. 15) afirmam:

> Na melhor das hipóteses, os dispositivos podem ser configurados por seus proprietários; porém, quanto ao mais, eles precisam descobrir automaticamente seu ambiente e 'se encaixar' o melhor que puderem.

Sistemas domésticos, sistemas eletrônicos de tratamento de saúde e redes de sensores são alguns exemplos de sistemas distribuídos pervasivos. Neste trabalho, entre as inúmeras áreas existentes, abordamos mais especificamente a área de sistemas domésticos para segurança e controle de acesso, no qual envolvem conceitos de automação e casas inteligentes.

2.2.3 Computação Ubíqua

WEZER e BROWN definiram em seu artigo *"The coming age of calm technology"* escrito em 1996, o seguinte conceito sobre a computação ubíqua:

> O nome Computação Ubíqua refere-se à terceira onda da computação, que agora inicia. Primeiro vieram os mainframes, no qual eram compartilhados por diversas pessoas. Atualmente estamos na era da computação pessoal, onde a pessoa e a máquina se encaram desconfortavelmente através do desktop. Em seguida vem a computação ubíqua, ou a era da tecnologia calma, quando a tecnologia recua para o fundo de nossas vidas [Tradução Nossa].

Assim, a computação ubíqua se beneficia integrando a mobilidade com a funcionalidade pervasiva, em suma, qualquer dispositivo computacional, enquanto em movimento, adapta-se ao ambiente, autoconfigurando-se de acordo com a necessidade (ARAUJO, 2003).

2.3 AMBIENTES INTELIGENTES

A domótica, mais comumente chamada de residências inteligentes ou ambientes inteligentes, é uma ciência que estuda a relação do homem e a casa. Ela faz as pessoas imergir em ambientes computacionais ativos que revelam a necessidade do uso de técnicas mais sutis para gerenciar a complexidade e o dinamismo dos ambientes residenciais, o qual é repleto de dispositivos eletrônicos interligados por uma rede (BOLZANI, 2010).

A principal motivação de construir ambientes inteligentes é a capacidade de automatizar e responder a necessidades comuns encontradas no dia-a-dia do humano. VENTURI (2005, p. 19) aborda uma proposta integradora buscando dar resposta às necessidades do homem, agrupando em três grupos:

a) As necessidades de segurança estão relacionas com:
    - A qualidade do ar,
    - A prevenção de acidentes físicos e materiais,
    - A assistência à saúde,
    - A segurança contra intrusos.
b) As necessidades de conforto ambiental implicam na criação de um meio ambiente agradável:
    - Conforto térmico,
    - Conforto acústico,
    - Conforto visual,
    - Conforto olfativo,
    - Conforto espacial.
c) As necessidades de conforto de atividades vêm facilitar os hábitos cotidianos:
    - Para dormir,
    - Para alimentar-se,
    - Para cuidar-se,
    - Para manutenção (dos locais e dos materiais),
    - Para comunicar-se,

- Para divertir-se,
- Para trabalhar.

2.3.1 Sistemas Domésticos

Os sistemas domésticos são softwares que gerenciam os procedimentos lógicos dos ambientes inteligentes, em suma, contém o controle de todos os dados e o mais importante: integram aparelhos eletrônicos típicos, como equipamentos de áudio e vídeo, aparelhos de televisão, smartphones, PDAs, eletrodomésticos e outros equipamentos de uso pessoal em um único sistema através da rede doméstica.

De acordo com Tanenbaum e Steen (2007) é um desafio importante para tal sistema ter a capacidade de ser completamente configurável e auto gerenciável. Pois, não se pode esperar que usuários finais estejam dispostos ou sejam capazes de manter um sistema distribuído doméstico ligado e em funcionamento se seus componentes forem propensos a erros, por mau uso de seus usuários, como acontece com muitos dispositivos existentes hoje.

Como já citado acima, o sistema é um software que gerencia e integra os aparelhos. Contudo diversas aplicações devem ser feitas sob medida a cada residência, esse ponto se utiliza automação residencial, enquanto o software possui uma forma de abstrair os dados e comandos dos dispositivos computacionais presentes no meio residencial a automação tratará o hardware com a programação para a máquina, em suma, a união dessas áreas possibilita a interação e gerência do sistema doméstico sobre o dispositivo automatizado.

2.3.2 Automação Residencial

A automação residencial é um ramo da automação predial especializada no controle de operações no âmbito doméstico. Em geral, coletam-se informações sobre o ambiente por meio de sensores, analisam-se seus parâmetros e tomam-se decisões segundo um programa específico. Essas decisões podem disparar ações que, por sua vez, podem alterar o estado de atuadores[1] que modificam o ambiente, reduzindo a necessidade da intervenção humana (BOLZANI, 2010).

---

[1] Atuadores segundo MARCIEIRA (2006) são mecanismos gerenciais destinados a executar uma ação, tais como: ligar um motor, fechar uma válvula ou movimentar uma esteira.

Acionar a luz ao haver movimento, acionar sirenes e filmagens caso portas ou janelas sejam abertas indevidamente, desligar a bomba da caixa d'agua quando estiver cheia e acionar o ar condicionado quando certa temperatura for alcançada são exemplos do que pode ser feito.

A automação residencial pode possuir um forte laço de benefícios com os sistemas domésticos, pois ela permite abstrair, armazenar e controlar os dados oriundos dos sensores que captam as informações do ambiente. Além disso, através dessa parceria, é possível pensar novos cenários, tais como poder interagir com os equipamentos e eletrodomésticos presentes no ambiente residencial a partir de qualquer local onde haja conexão com a rede. Além disso, pode ser empregado o uso de ferramentas como Web Services e Sockets, ou ainda páginas Web, o que agrega mais um conceito: a computação ubíqua.

2.4 WEB SERVICES

Web Services surgiram com intuito de utilizar o protocolo HTTP para acessar remotamente métodos, permitindo que diferentes aplicações de diferentes plataformas possam acessar esses serviços e trocar informações, para tanto utiliza um padrão de comunicação. Um dos mais aceitos e utilizados é o protocolo SOAP (Simple Object Access Protocol).

No presente projeto, Web Services tem o importante papel de conectar a aplicação doméstica com a Internet, disponibilizando métodos para serem acessadas por qualquer aplicação que utilize o protocolo SOAP. O que possibilita maior interação entre plataformas distintas.

O SOAP interage através da arquitetura cliente-servidor de forma assíncrona com a internet, a maior vantagem da sua utilização é justamente a capacidade de comunicar diferentes plataformas de programação, ou seja, independente da linguagem utilizada no software, se ele souber interpretar SOAP, as mensagens serão trocadas e interpretadas. Para tanto define um esquema XML[2] (Extensible Markup Language) para representar o conteúdo das mensagens de requisição (COULOURIS, DOLLIMORE, KINDBERG T, 2008, p. 677).

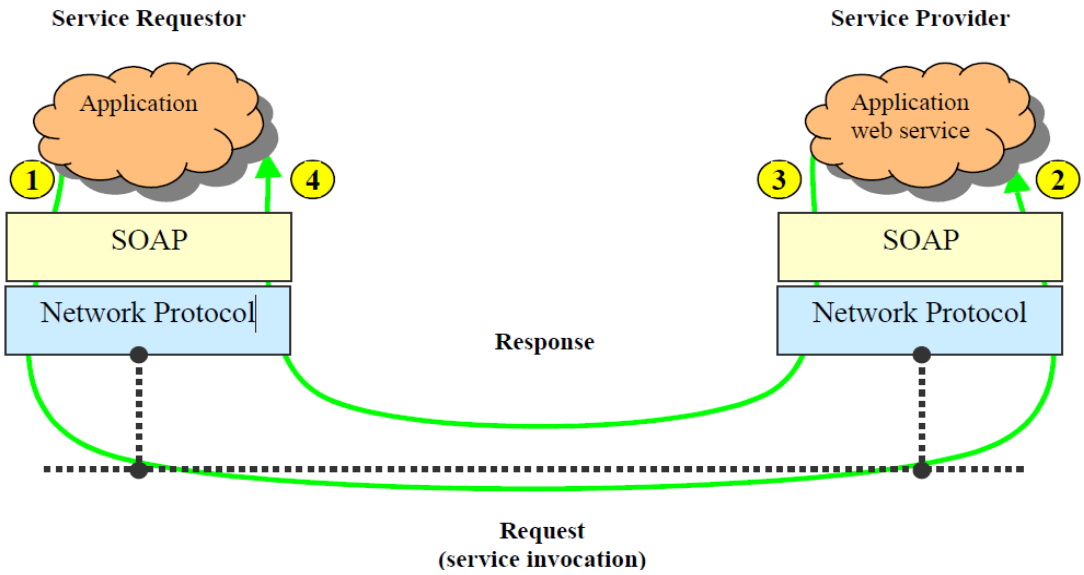
O uso de esquemas XML é mais lento e necessita alocar mais espaço que arquivos binários. Contudo, segundo os autores, a sua utilização se justifica pelo fato de apresentar

[2] O XML é um formato textual auto descritivo, rotulado com strings de marcação (tags), relacionadas à estrutura do texto que englobam, em contraste com a HTML (COULOURIS G., DOLLIMORE J., KINDBERG T., 2008)

formato legível para seres humanos, o que facilita a construção de mensagens simples e a depuração das mais complexas.

A Figura 2 ilustra essa interação, onde a marcação 1 representa a requisição feita pelo solicitante do Serviço (na imagem descritor por *Service Requestor*), no qual envia a requisição (Request - *Service Invocation*) para o Provedor do Serviço (na imagem *Service Provider*, Marcado com o número 2); após receber essa requisição o Provedor envia sua resposta (*Response*) – Marcação 3 na figura – para o Solicitante do Serviço – Marcação 4 na figura. Durante esse processo é utilizado o SOAP para invocar os métodos necessários para fazer as requisições e poder enviar as respostas que por sua vez utilizaram um protocolo de rede para encaminhar essas mensagens.

Figura 2 - Demonstração da comunicação de uma aplicação utilizando Web Service SOAP



Fonte: KREGER, 2001, p. 14.

2.5 MICROCONTROLADORES

Os microcontroladores são computadores embutidos em uma grande variedade de aparelhos diferentes com capacidade de gerenciar e manipular suas interfaces. Para TANENBAUM (2007) os microcontroladores são considerados computadores pequenos, mas completos, compostos por processador, memória e capacidade de E/S (entrada e saída) o qual permite detectar botões e interruptores do aparelho além de controlar suas luzes, monitores, sons e motores.

Por possuírem a característica de terem baixo custo, pequenos em tamanho, consumirem pouca energia, terem pouca memória, baixa capacidade de processamento e conseguirem monitorar e controlar componentes acoplados neles, os fabricantes de muitos aparelhos eletrônicos utilizam microcontroladores em seus produtos, tais como, em eletrodomésticos, aparelhos de comunicação, periféricos de computadores, equipamentos médicos, equipamentos para entretenimento e brinquedos. Desta forma, os microcontroladores estão fortemente presentes na vida cotidiana das pessoas, ocultos nos diversos aparelhos eletrônicos digitais.

Os microcontroladores são classificados em versões de 4, 8, 16 e 32 bits, os quais possuem notáveis diferenças entre suas características de hardware. A Tabela 2 apresenta um exemplo das características de cada uma dessas versões baseado nos *datasheets* de alguns microcontroladores disponibilizados pelos fabricantes: Microchip (http://www.microchip.com/), Texas Instruments (http://www.ti.com/) e Atmel (http://www.atmel.com/).

Tabela 2 - Exemplos das características de microcontroladores nas versões 4, 8, 16 e 32 bits

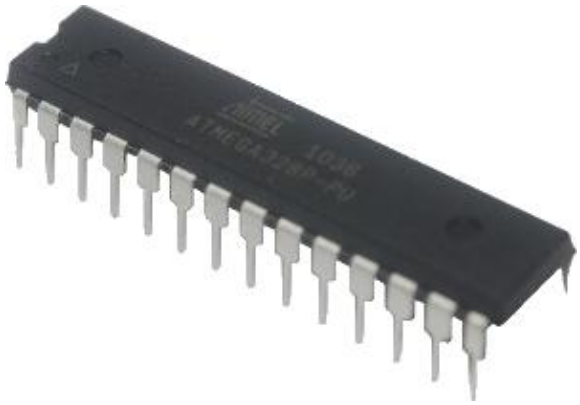
Descrição	Versão 4 bits	Versão 8 bits	Versão 16 bits	Versão 32 bits
Microcontrolador	ATAR080	Atmega328P	MSP430F2419	PIC32MX775F256H
Família	MARC4	8-bit AVR	MSP430	PIC32MX7xx
Fabricante	Atmel	Atmel	Texas Instruments	Microchip
Tecnologia	RISC	RISC	RISC	RISC
Terminais	20 pinos	28 pinos	80 pinos	64 pinos
Flash (KB)	2	32	120	256
SRAM (KB)	256 x 4	2	4	64
Max I/O Pins	12 pinos	23 pinos	64 pinos	53 pinos
Frequência Máxima (MHz)	4	20	16	80
Vcc (Volts)	1.8 - 6.5	1.8 - 5.5	1.8 - 3.6	2.3 - 3.6
PWM	-	6 canais PWM	-	5 canais PWM
USART	-	1 porta serial	2 portas seriais	6 portas seriais

Fonte: ATMEL, MICROCHIP e TEXAS, 2012.

O grande problema de utilizar microcontroladores é a conhecimento necessário em programação para hardware e eletrônica para conseguir fazê-lo interagir com dispositivos, uma vez que o *chip* que contêm o sistema embarcado do microcontrolador não funciona sem

que haja um circuito eletrônico que dê suporte. A Figura 3 exibe o formato externo do microcontrolador ATmega328P do fabricante Atmel.

Figura 3 - Microcontrolador ATmega238P, da Atmel



Fonte: WEBTRONICO, 2012.

Para minimizar o problema relativo à complexidade dos microcontroladores e do processo de desenvolvimento, há hardwares pré-prontos e IDEs que auxiliam nestas tarefas. Um projeto que fornece essas facilidades é o Arduino.

2.5.1 Arduino

O princípio do Arduino é tornar mais fácil a manipulação de hardwares e softwares embarcados, pois disponibiliza todos os recursos físicos para que o microcontrolador funcione, além de oferecer uma IDE e bibliotecas que facilitam o desenvolvimento do software. Como MARGOLIS, 2011 afirma: "O Arduino torna possível que qualquer pessoa com interesse, mesmo pessoas sem nenhuma experiência em programação ou eletrônica, consiga utilizar essa tecnologia rica e complexa".

É possível o fazer perceber e responder ao toque, sons, infravermelho, temperatura, peso, luz e posição, além de configurar as rotinas que ele vai responder. Isso aliado ao fato de possuir um hardware e software *open-source*, possibilita o desenvolvimento individual dos próprios dispositivos que podem ser utilizados tanto para consumo pessoal, como para comercializar os dispositivos criados.

Para tanto, além de dispor da placa Arduino que contêm o microprocessador Atmel AVR, como ilustra a Figura 4, ele utiliza uma linguagem de programação padrão baseada em C++, dispõem de uma IDE e extensões de hardwares pré-prontos, chamados de Shields, que são interfaces de hardware que podem ser acopladas no Arduino para interagir com algum

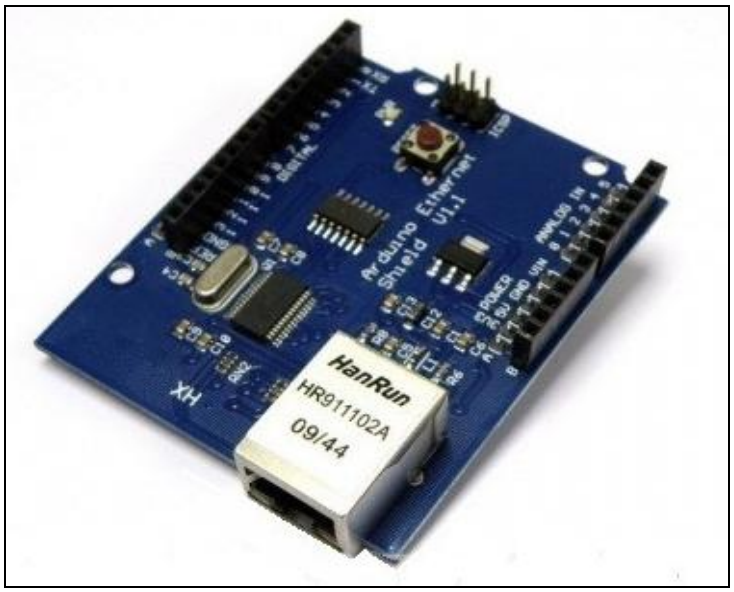
dispositivo ou realizar comunicação (a Figura 5 ilustra uma Shield Ethernet). Assim como MCROBERTS (2011, p. 24) afirma:

> O Arduino também pode ser estendido utilizando os shields (escudos), que são placas de circuito contendo outros dispositivos (por exemplo, receptores GPS, displays de LCD, módulos de Ethernet etc.), que você pode simplesmente conectar ao seu Arduino para obter funcionalidades adicionais. Os shields também estendem os pinos até o topo de suas próprias placas de circuito, para que você continue a ter acesso a todos eles.

Figura 4 - Arduino Duemilanove

Fonte: WEBTRONICO, 2012.

Figura 5 - Shield Ethernet



Fonte: WEBTRONICO, 2012.

3 PROJETO UBGEAR

O projeto UBGEAR (*Ubiquitous Generic Architecture* - Arquitetura Ubíqua Genérica) propõe a construção de uma arquitetura para sistemas ubíquos aplicada em ambientes que necessitem de um middleware para interagir remotamente. Nas seções posteriores serão descritos um cenário de aplicabilidade da proposta, o seu modelo conceitual da arquitetura, a descrição individual dos módulos e as propostas de modelos de comunicação e interação entre as camadas que a compõem.

.

3.1 CENÁRIO DE APLICABILIDADE

Esta seção tem por objetivo apresentar um cenário de aplicabilidade da proposta que será detalhada, a fim de melhor abordar os problemas e situações que se pretende tratar na arquitetura proposta. Entre os exemplos de aplicações pode-se citar a que segue abaixo.

Em uma residência existem diversos equipamentos atuadores os quais seus moradores acionam manualmente, tais como: aquecedor, ar-condicionado, a iluminação dos cômodos, o irrigador do jardim, as cortinas, o alarme e as câmeras de segurança. Muitas vezes esses equipamentos podem ser acionados em situações previsíveis, como: o aquecedor é ligado quando estiver frio para o morador presente no peça da casa; sempre que o usuário entrar em uma peça sem luminosidade as lâmpadas serão ligadas e quando ele sair as lâmpadas serão desligadas; e sempre que houver muito sol em cima dos aparelhos eletrônicos as cortinas serão fechadas e sempre quando amanhecer e o morador levantar da cama são abertas as cortinas. Todas essas ações podem ser automatizadas através de sistemas residenciais.

Para tanto é necessário que o sistema capte informações para saber se este evento previsível deve ou não executar, desta forma utilizam-se sensores para coletar essas informações, como nos casos: para saber que temperatura há nas peças da casa será necessário ter sensores de temperatura distribuídos pela residência; para ativar/desativar a iluminação automaticamente é necessário ter sensores para saber onde a pessoa está na casa ou se ela entrou ou saiu do cômodo; e sensores de intensidade luminosa para saber se a luz do Sol está incidindo diretamente, fechando assim a cortina.

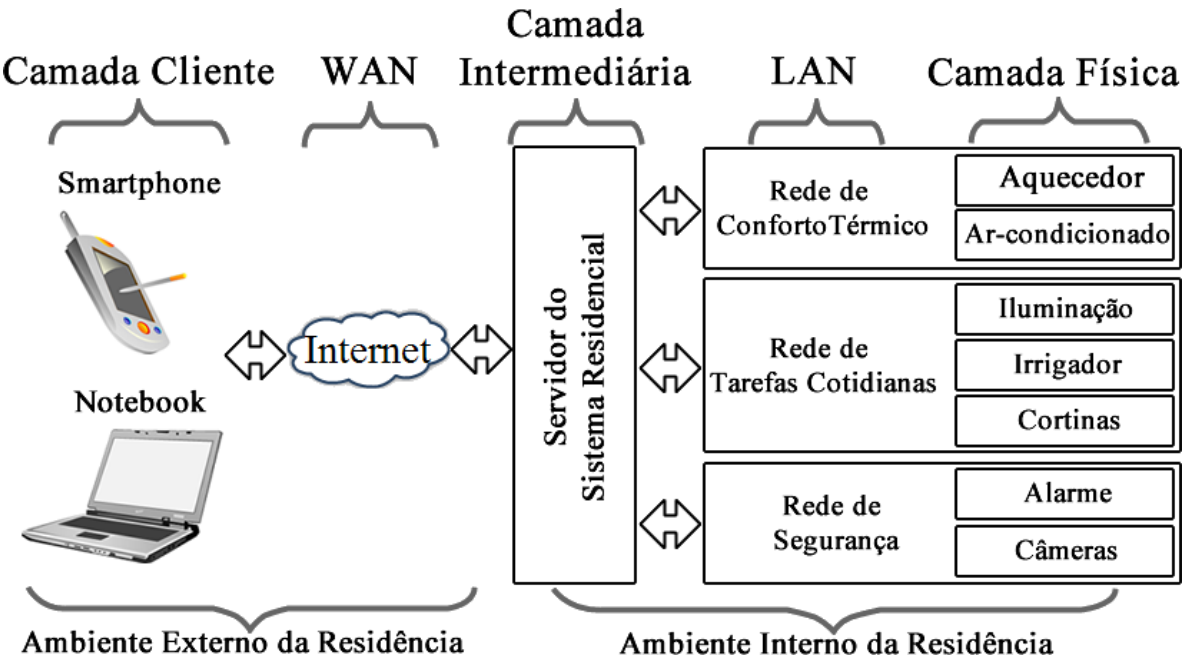
Desta forma esses sistemas provem mais conforto aos seus moradores, automatizando ações cotidianas, aumentando a segurança com o uso de equipamentos que controlam o acesso e alertam sobre intrusos e oferece aos moradores economia de energia devido ao fato

de gerenciar os elementos que estão em funcionamento, desligando-os e ligando-os conforme a necessidade.

Uma vez automatizada a residência ela é considerada um ambiente pervasivo, auto gerenciado, contudo se o morador não estiver em sua residência não vai ter certeza do que está acontecendo nela, além disso, muitas vezes o morador quer ser informado sobre alguns eventos particulares na residência, mesmo estando em outros locais. Para tanto, conectar o sistema residencial na internet possibilitará que o seu dono saiba tudo o que está acontecendo na casa, mesmo que esteja longe dela e em movimento, sendo avisado sobre determinados eventos e podendo controlar muitos dos equipamentos remotamente. Por exemplo: a) o dono foi viajar de férias e quer ver através de seu notebook se o sistema de segurança está ativado, que as plantas estão devidamente irrigadas e as janelas fechadas; b) se alguma janela ou porta for aberta indevidamente o dono receberá uma mensagem pelo celular podendo a partir do celular acessar as câmeras da casa para ver o que está ocorrendo, podendo tomar as devidas providências; c) quando o dono entrar em sua residência será reconhecido e automaticamente o sistema de conforto térmico ajustará a temperatura para um valor agradável a seu dono.

A Figura 6 ilustra todos das as camadas que compõem este cenário, onde: a camada cliente representa por onde o usuário cliente interage com o sistema residencial, na imagem representadas por um Smartphone e um Notebook; a WAN é o meio por onde o usuário envia as informações do ambiente externo da residência para o interno, no caso a Internet; a camada intermediária é responsável por conter o sistema de gerência da residência; a LAN é o meio em que o servidor do sistema residencial interage com os equipamentos da camada física; a camada física gerência os equipamentos físicos permitindo que eles sejam controlados pelo servidor do sistema residencial (na imagem os equipamentos descritos são os mesmos citados no exemplo acima).

Figura 6 - Exemplo de componentes internos e externos do ambiente residencial



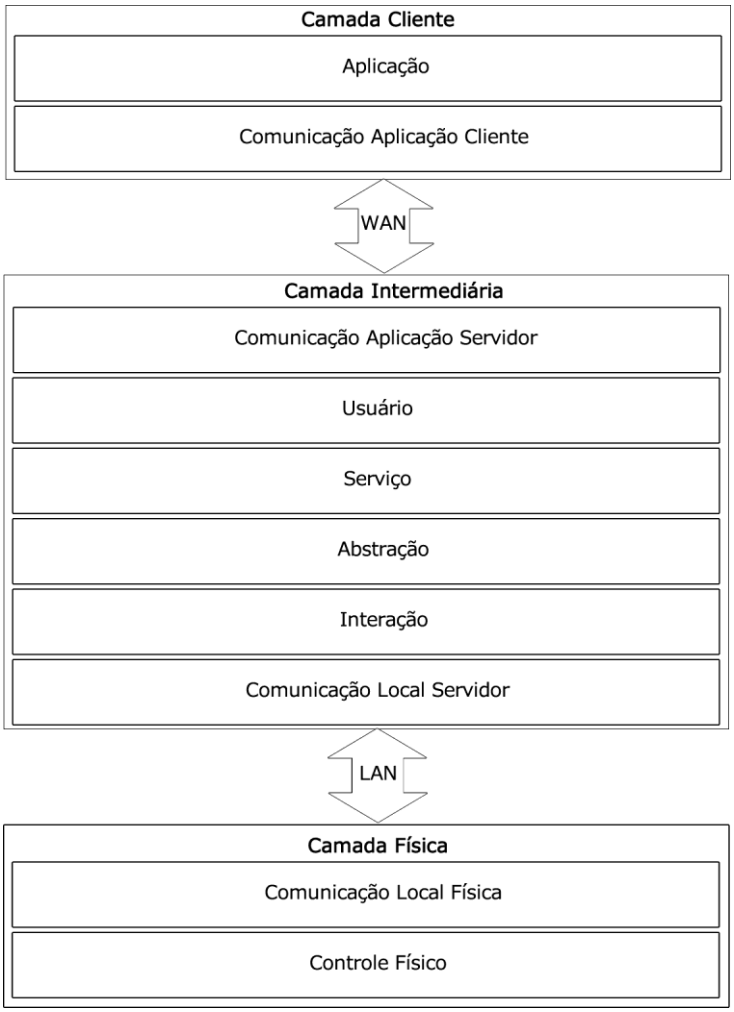
Fonte: do Autor.

3.2 ARQUITETURA CONCEITUAL

A arquitetura conceitual do projeto se baseia em três camadas: cliente, intermediário e físico; nos quais representam as funcionalidades do cliente, do middleware e do controle do ambiente físico respectivamente. Como apresenta a Figura 7, as camadas possuem módulos de software com funções específicas, as quais serão detalhadas nas seções posteriores, sendo elas:

- Camada Cliente, composta pelos módulos: Aplicação e Comunicação Aplicação Cliente;
- Camada Intermediária, composta pelos módulos: Comunicação Aplicação Servidor, Usuário, Serviço, Abstração, Interação e Comunicação Local Servidor; e
- Camada Física, composta pelos módulos: Comunicação Local Física e o Controle Físico.

Figura 7 - Representação conceitual das Camadas da Arquitetura do UBGEAR.



Fonte: do Autor.

A principal motivação do uso desta arquitetura se baseia em sua representação simplificada que permite ser adaptada para qualquer tipo de ambiente de forma flexível, possibilitando, de acordo com a necessidade, centralizar ou descentralizar o processamento de condição e decisão das funcionalidades dos nós de controle através do Servidor. Como exemplo pode-se citar os casos: a) O controle físico possui um micro controlador que gerência um sensor de interrupção magnético e uma sirene de alarme, quando o sensor for ativado o micro controlador liga a sirene de alerta, após avisa o servidor sobre o ocorrido. Este é um caso onde o processamento é descentralizado, pois a decisão encontra-se na própria camada física, não necessitando, assim, de intervenção do serviço para ativar a sirene. b) em uma residência a camada física é composta por 20 microcontroladores distribuídos em todos os cômodos de uma casa, os quais cada um gerência um conjunto de sensores de interrupção magnético postos em janelas/portas e uma sirene, quando um desses sensores for ativado, todas as sirenes de todos os micro controladores devem ser ativadas simultaneamente, além

disso o dono da casa deve receber uma mensagem por SMS sobre o ocorrido. Neste caso o processamento da decisão deverá ser centralizado no serviço, pois exige mais processamento para executar todas essas funções e ainda exige ter acesso a rede global para finalizar o envio do SMS, não cabendo ao nó de controle a responsabilidade total de processar todas essas funções.

3.3 CAMADA FÍSICA

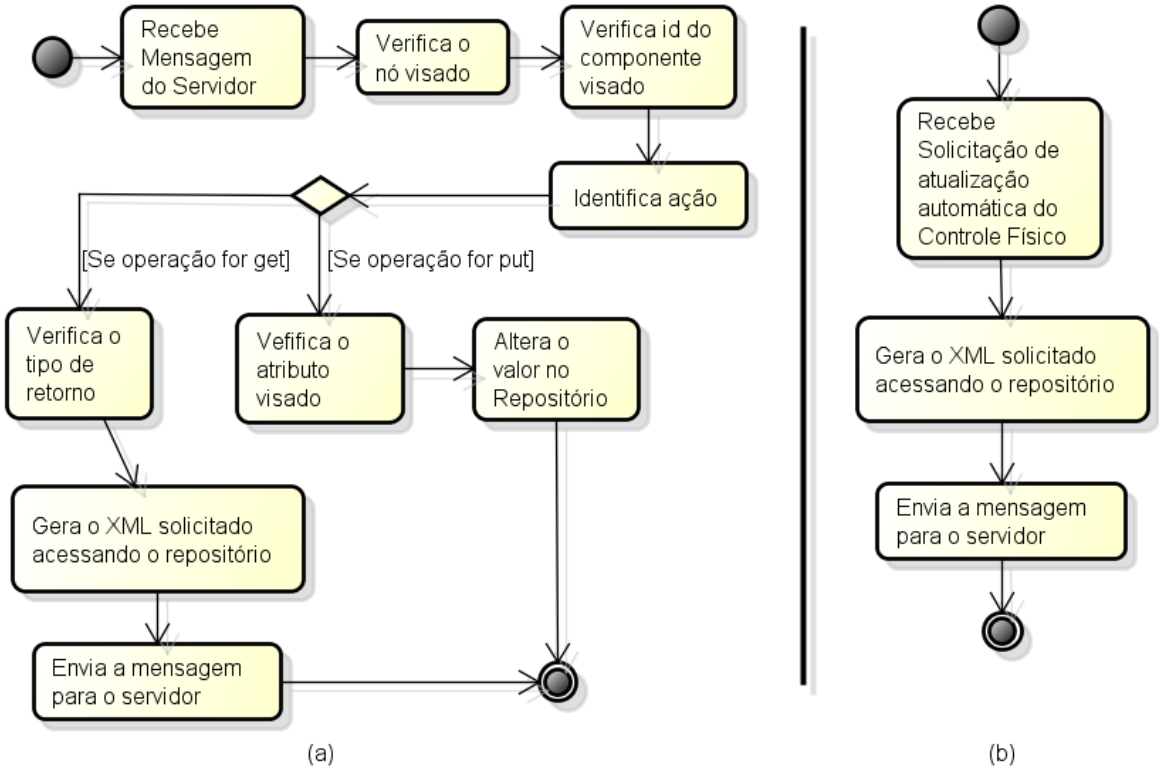
Este nível é composto pelos módulos de software que interagem com o hardware, gerenciando as funções e coletando os dados. Nela se incluem todas as instruções que lidam diretamente com os sensores e atuadores, tais como: controladores, câmeras, leitores de cartão, motores, foto células, sensores de temperaturas, RFID (*Radio-Frequency Identification*), entre outros. Dentro desta camada pode-se separá-la em duas funcionalidades principais: comunicação local física e controle físico. BOLZANI (2010, p. 84) nomeia como nó de controle o hardware e software responsável por gerenciar um ou mais dispositivos, função desempenhada pela camada física do presente projeto.

3.3.1 Comunicação Local Física

Faz a comunicação com a camada intermediária, configurado o protocolo para traduzir os envelopes de requisição oriundos da comunicação local servidor e codificar os envelopes de resposta para a camada intermediária utilizando, por exemplo, formas de transmissão: Ethernet; Bluetooth; Serial, ZigBee; ou infravermelho. Além de receber envelopes do servidor e responder a ele, a comunicação local do hardware pode fazer o envio automático de atualizações, medida utilizada para agilizar certos casos onde é necessária uma resposta imediata do servidor. A representação gráfica com os processos de funcionamento deste módulo está nos diagramas de atividade da Figura 8, onde o diagrama 8a apresenta a atividade ao receber mensagem do servidor e o diagrama 8b atividade ao receber solicitação do controle físico.

Para realizar a comunicação com o servidor, o nó de controle gera modelos XML contendo os dados dos respectivos componentes, expressando seu funcionamento. Esses modelos de XML podem ser encontrados na seção 3.2.2 - Comunicação entre camadas intermediária e baixa.

Figura 8 - Funcionamento da Comunicação Local Física

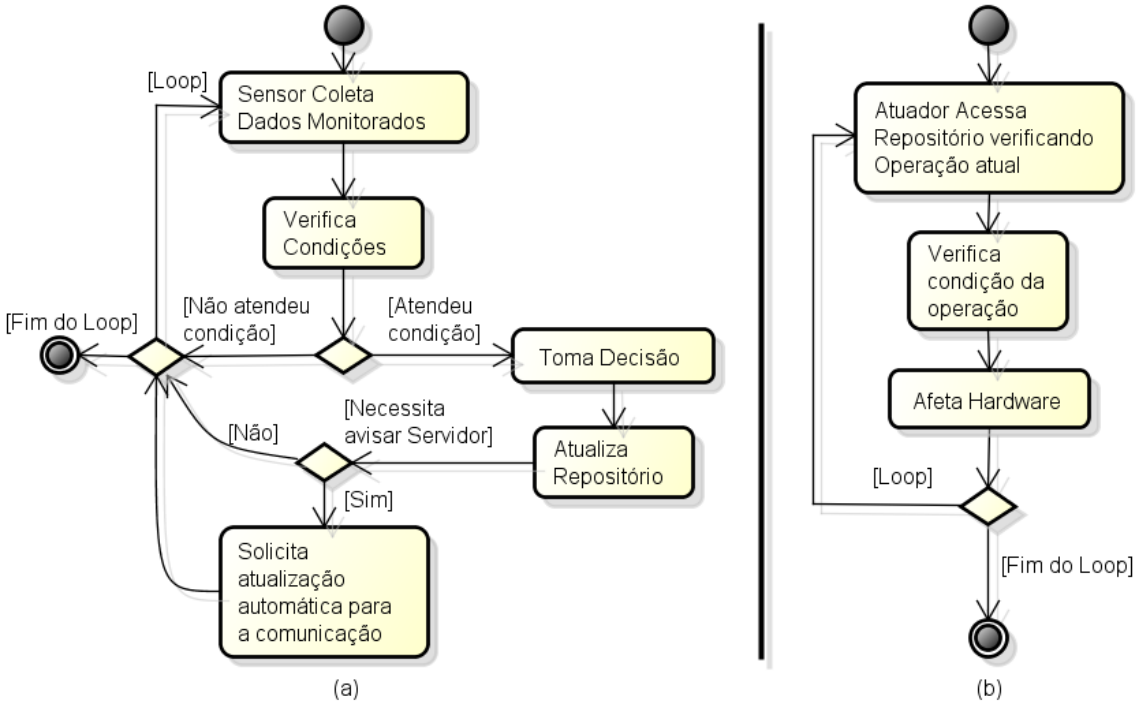


Fonte: do Autor.

3.3.2 Controle Físico

É responsável pela comunicação com os sensores e atuadores, fazendo respectivamente, a leitura dos dados dos sensores e o encaminhamento de ordens para os atuadores. Também, exercem a função de interpretar as ordens recebidas pela comunicação, executando a ação referente a essa ordem no hardware. Internamente ele possui três funcionalidades principais: a primeira lida com as condições dos dados monitorados; a segunda é responsável pelas decisões encadeadas pelas condições e; a terceira é responsável por armazenar as preferências e dados de controle dos primeiros dois módulos além de conter a descrição dos metadados que dão significado aos dados que interagem com o hardware. A Figura 9 apresenta através de dois diagramas as atividades das funcionalidades de controle físico, onde a Figura 9a apresenta a atividade do Atuador e a Figura 9b apresenta a atividade do Sensor, estas representam a interação entre as principais funcionalidades, sendo respectivamente condições, decisões e configurações.

Figura 9 - Atividades das funcionalidades de Controle Físico



Fonte: do Autor.

Há ainda outra funcionalidade que pode ser adicionada ao módulo de controle físico, que é desejável, contudo não obrigatória, no qual permite que o nó de controle não fique obsoleto enquanto não há comunicação, ou seja, o ideal é que o software embarcado possua instruções para lidar com os dispositivos ligados a ele enquanto a comunicação com a camada intermediária estiver indisponível. Em outras palavras, ainda haverá um serviço básico em funcionamento enquanto essa conexão não for reestabelecida. Tais medidas como continuar gravando em um cartão SD os dados monitorados pelos sensores e encaminhar decisões para os atuadores conforme as condições impostas ao ler os dados monitorados. Os nós de controle podem agir de três formas: totalmente dependente; parcialmente dependente ou; independente do Servidor.

Totalmente dependente do Servidor: o hardware envia seus dados ao servidor e age de acordo com as respostas e ordens oriundas dele. Neste caso o Servidor terá completo controle do processamento e das ações do hardware, contudo se houver problemas na comunicação o hardware ficará ocioso, no pior dos casos poderá ficar postergado indefinidamente até que a comunicação se reestabeleça. Podemos citar o caso hipotético em que há o uso de câmeras de vigilância no qual somente podem ser acessadas pelo sistema,

também podemos citar o caso em que se compra um hardware de algum fabricante que não possui um programa configurado para agir sem que haja ordem externa;

Parcialmente dependente do Servidor: Caso em que o hardware necessita da interação com o servidor para funcionar, contudo se houver falha na comunicação o hardware ainda estará em funcionamento. Um exemplo é quando o dispositivo de hardware não possui processamento ou memória suficiente para executar alguma operação específica que não compromete o total funcionamento do hardware, é o caso de dispositivos que não possuam memória própria suficiente para salvar seus dados no decorrer do tempo;

Independente do Servidor: Independente de haver comunicação com o servidor ou não, o dispositivo continuará a fazer todas as funções que exerce. Podemos fazer a analogia em que o servidor exerce a função de observador do hardware, em outras palavras, o hardware age e depois comunica a ação para o servidor, se não houver possibilidade de comunicação ele irá executar suas funções normalmente. Um exemplo disso é um portão com trava magnética acessada por senha, no qual é preciso registrar o horário e o usuário que está passando pelo portão, neste caso o sistema interno nunca pode parar de funcionar, por motivos de segurança. Se em algum momento a comunicação for interrompida ele continuará a guardar os logs em memória própria e a exercer suas rotinas normais. Quando reestabelecer a comunicação, o servidor compara sua lista de logs com as do dispositivo, resgatando os dados que antes não puderam ser acompanhados devido ao problema de comunicação.

3.4 COMUNICAÇÃO LOCAL ENTRE CLIENTE E SERVIDOR

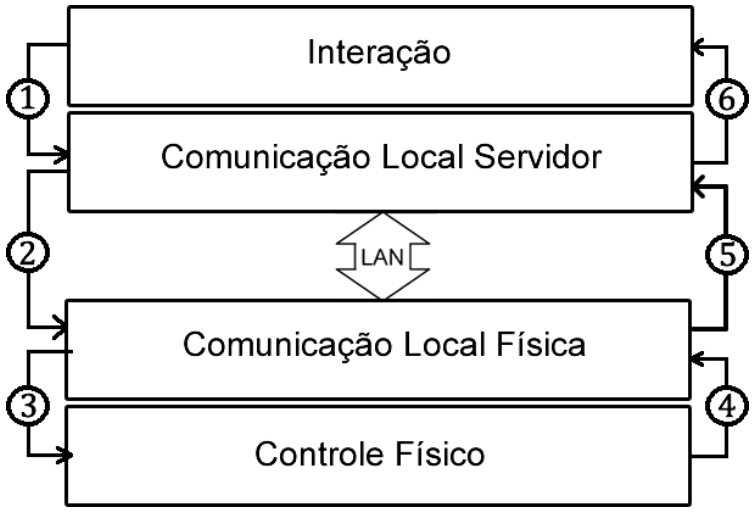
Para conseguir integrar o maior número de hardwares possíveis com softwares é necessário que haja um padrão de representação dos formatos de dados que faça o meio de campo entre eles. Pensando nisso utilizou-se nesse trabalho modelos em XML para integrá-los, pois é necessário um padrão fácil de ser utilizado para a troca de informações entre diversas aplicações, plataformas e ambientes de rede (BASSO, 2002). O uso desses modelos em XML em conjunto com a adoção do protocolo de interação permite que a arquitetura consiga ser heterogenia em relação ao hardware, para tanto a Figura 10 representa como ocorre esse processo, onde:

1 – Quando o módulo de interação recebe uma ordem para executar uma ação, este cria um envelope de requisição e preenche os dados codificando-os pelo protocolo de interação correspondente a comunicação local física do nó de destino. Após, envia o envelope para o módulo de comunicação local servidor;

2 – A comunicação local servidor recebe o envelope e encaminha-o, através da LAN, para a comunicação local física referente ao nó de controle destinado;

3 – Ao receber o envelope, a comunicação local física examina o seu conteúdo verificando quem é o destinatário e que tipo de ação deve ser executada, então encaminha esta ação para o controle físico do dispositivo de destino do envelope;

Figura 10 - Processo de interação entre as camadas intermediária e física



Fonte: do Autor.

4 – O controle físico recebe a ação e a executa, se for uma ação de modificação de dado, o processo cessa aqui, se for uma ação para retorno de dados, é gerado um XML do dispositivo e passado para a comunicação local física;

5 – O módulo de comunicação local física envelopa o XML e o envia para a comunicação local do servidor.

6 – A comunicação local do servidor recebe o envelope e o encaminha para o módulo de interação que extrai o XML do envelope, verifica a versão do XML e então o traduz com o protocolo de interação para ser possível atualizar a representação lógica do dispositivo físico monitorado.

Esse processo de interação entre as duas camadas só é possível por que tanto o módulo de interação quanto a comunicação local física utilizam o mesmo protocolo de interação para codificar e decodificar as mensagens.

Desta forma é possível avaliar para a comunicação local, entre a camada intermediária e física em suas situações: Físico para o Servidor; Servidor para o Físico. O primeiro caso, realizado do físico para o intermediário, utiliza a geração de esquemas em XML com

elementos pré-definidos os quais o Servidor irá interpretar. O segundo é realizado do nível intermediário para o nível físico, contendo as informações em um formato simplificado e objetivo para que o nó de controle consiga interpretar. Os dois protocolos são descritos com maiores detalhes nas próximas seções.

3.4.1 Físico para Servidor

Para realizar a interação da camada física para o servidor é utilizado modelos em XML que, independente da forma de comunicação, serão interpretados pelo servidor. Como há uma limitação de alguns hardwares em relação ao modo que transporta os dados de seu tipo de comunicação, pode ocorrer que um XML seja muito extenso para ser enviado de uma só vez, pensando nisso é necessário a utilização de versionamento para esses modelos. Isso permite também que casos que utilizem novos modelos de dados possam ser adicionados em novas versões quando houver necessidade, não necessitando alterar os modelos já existentes que hipoteticamente estão fechados e em uso. Esta seção apresenta duas versões (1.0 e 1.1) de modelos XML, bem como a descrição com breve explicação sobre o conteúdo de seus elementos.

3.4.1.1 Modelo Versão 1.0

O modelo 1.0 se preocupa em lidar com XMLs maiores, tendo como objetivo realizar somente uma conexão para enviar todos os dados para o servidor.

Modelo Completo dos Dados XML: O modelo de dados em formato XML apresentado abaixo foi construído com intuito de facilitar a identificação por parte do servidor dos dados fornecidos pelo nó de controle sobre seu endereço, protocolo que utiliza para identificar os metadados e os dos dados de cada componente. Em suma, esta estrutura de XML possui os dados básicos para o sucesso da interpretação dos dados, possibilitando futuramente aumentar o número de dados que o compõem. A Figura 11 exibe um exemplo de modelo completo dos dados XML.

Figura 11 - Modelo Completo dos Dados XML

```
<?xml version="1.0" encoding="UTF-8"?>
<root version="1.0">
    <config>
        <model>1.0</model>
        <serie>TCC</serie>
        <mac>xx-xx-xx-xx-xx-xx</mac>
        <ip>xxx.xxx.xxx.xxx</ip>
        <protocol>default</protocol>
        <nodes>1</nodes>
        <devices>3</devices>
    </config>
    <components>
        <metadata>
            <node>1</node>
            <activity>s</activity>
            <id>1</id>
            <status>t</status>
            <device>2</device>
            <datatype>d</datatype>
            <data>21.2304384393</data>
            <about>Temperatura em graus célsius</about>
            <min>-30</min>
            <max>150</max>
        </metadata>
        <metadata>..Componente dois..</metadata>
        <metadata>..Componente três..</metadata>
    </components>
</root>
```

Fonte: do Autor.

Modelo de Coleta de Completa de Dados XML: Este modelo permite que sejam transportados todos os dados de uma única da representação lógica para o servidor. A Figura 12 exibe um modelo de coleta completa de dados.

Figura 12 - Modelo de Coleta Completa de Dados

```
<?xml version="1.0" encoding="UTF-8"?>
<root version="1.0">
    <config>
        <ip>10.5.99.191</ip>
    </config>
    <metadata>
        <node>1</node>
        <activity>s</activity>
        <id>1</id>
        <status>t</status>
        <device>2</device>
        <datatype>b</datatype>
        <data>t</data>
        <about>Reed Switch</about>
        <min>0</min>
        <max>1</max>
    </metadata>
</root>
```

Fonte: do Autor.

Modelo de Coleta Simplificada de Dados XML: Este modelo permite que seja transportado somente os dados necessários para atualização da representação lógica no servidor, isto diminui o tamanho do envelope que será transportado para o servidor, buscando agilizar a comunicação e economizar processamento, em comparação ao modelo em que é coletado todos os dados do componente. A Figura 13 exibe um exemplo de modelo de coleta simplificada de dados.

Figura 13 - Modelo de Coleta Simplificada de Dados

```
<?xml version="1.0" encoding="UTF-8"?>
<root version="1.0">
    <config>
        <ip>10.5.99.191</ip>
    </config>
    <metadata>
        <node>1</node>
        <activity>115</activity>
        <id>1</id>
        <status>116</status>
        <datatype>98</datatype>
        <data>t</data>
    </metadata>
</root>
```

Fonte: do Autor.

3.4.1.2 Modelo Versão 1.1

O modelo 1.1 é focado em casos que é necessário o uso de envelopes de comunicação menores, devido ao fato do método de comunicação físico não suportar um volume grande de informações enviadas de uma só vez. Assim, ele utiliza os modelos em XML de Coleta do Mapa de Componentes, Coleta de Configurações em adição aos modelos de Coleta Simplificada e Completa de Dados, oriunda do modelo 1.0.

Modelo de Coleta do Mapa de Componentes XML: Este modelo permite que sejam transportados os dados de mapeamento dos componentes dentro do nó de controle, após o servidor receber essa informação ele requisita as configurações e os dados individuais de cada componente. Usa-se caso seja muito grande o Modelo Completo dos Dados XML. A Figura 14 exibe um modelo de mapa de componentes.

Figura 14 - Modelo de Coleta do Mapa de Componentes

```
<?xml version="1.0" encoding="UTF-8"?>
<root version="1.1">
    <map>
        <node>1</node>
        <id>1</id>
    </map>
    <map>
        <node>1</node>
        <id>2</id>
    </map>
    <map>
        <node>1</node>
        <id>3</id>
    </map>
</root>
```

Fonte: do Autor.

Modelo de Coleta de Configurações XML: Este modelo permite que sejam transportados todos os dados de configuração da representação lógica no servidor. A Figura 15 exibe um modelo de coleta de configurações.

Figura 15 - Modelo de Coleta de Configurações

```
<?xml version="1.0" encoding="UTF-8"?>
<root version="1.1">
    <config>
        <model>1.0</model>
        <serie>TCC</serie>
        <mac>84-85-88-16-0-36</mac>
        <ip>10.5.99.191</ip>
        <protocol>default</protocol>
        <nodes>1</nodes>
        <devices>3</devices>
    </config>
</root>
```

Fonte: do Autor.

3.4.1.3 Descrição dos Metadados

Os metadados são as informações, configurações ou descritores referentes a cada componente presente no nó de controle, eles são representados dentro dos XML através dos elementos. Por medidas de segurança apenas os metadados Status, Data dos atuadores e o Status dos sensores podem ser alterados. Em suma podemos separá-los em duas classificações: padrão e opcionais.

Elementos Padrão:

- `<root version="1.0">`: Elemento raiz dos modelos XML, o atributo versão dentro deste elemento indica que versão de modelo de coleta está sendo utilizado, no caso podendo ser 1.0 ou 1.1;
- `<model>`: Contém o valor referente ao modelo de implementação atual em que o nó de controle se encontra;
- `<serie>`: Contém o número de série que o nó de controle foi marcado, essa informação, juntamente com o modelo, ajuda a identificar quem fez essa implementação e se ela já é conhecida pelo sistema;
- `<activity>`: Indica qual atividade o dispositivo opera, ou seja, indica se o dispositivo é um sensor ou um atuador;
- `<id>`: É a informação que separa logicamente os componentes dentro do nó de controle;

- <status>: Indica se o dispositivo está em funcionamento ou não, tanto para sensores como atuadores, representa que ele está ou não sendo monitorado, por motivos que podem ser desde ordem do servidor até por erro ou mau funcionamento detectado pelo próprio tratamento de erros do nó de controle;
- <device>: Indica qual o tipo de componente. O valor deste metadado é diretamente relacionado ao protocolo de interação que o nó de controle utiliza, pois através dos dois o servidor identifica que tipo de componente ele está lidando, tomando assim as providências específicas. Exemplo: temperatura, infravermelho, interruptor, câmera, motor, LED e lâmpada;
- <datatype>: Indica o tipo de dado que é utilizado (Double, Int, String, Boolean, Char, Streaming, entre outros), sendo um complemento do metadado device, descrito acima. No caso dos sensores indica o tipo de dado que resulta de seu monitoramento e no caso do atuador indica o comportamento que ele deve tomar (Operação). Exemplo: Double para temperatura, pois utiliza a grandeza Célsius; Integer para Motor, para alterar sua velocidade e; Streaming para uma câmera, pois utiliza imagens como dado;
- <data>: É o valor referente ao tipo de dado citado acima;
- <mac>: É o valor de endereço MAC atribuído ao nó de controle. Ele pode ser utilizado para fazer conferência de dados e para permissões;
- <ip>: É o endereço lógico, caso o nó de controle utilize um protocolo de Ethernet para comunicar-se com o Servidor;
- <nodes>: quando uma camada de comunicação gerência as mensagens de mais que um nó de controle é necessário informar esse número para que seja escolhido o nó correto, por padrão esse valor é 1 (um);
- <devices>: Indica o valor referente ao número de componentes que o nó de controle contém;
- <node>: Identifica o nó que o dispositivo se encontra;
- <protocol>: É a referência ao protocolo de interação que está codificado os dados do nó de controle, através da referência deste protocolo a camada intermediária identifica o real significado do valor codificado. Seu uso se justifica pelo fato de possibilitar um maior número de formas de tradução de dados, além de possibilitar utilizar referências com tamanhos menores. Exemplo: para o protocolo X um sensor de temperatura é representado pelo caractere ‘1’, esse caractere é transportado mais rapidamente que a cadeia de caracteres “Temperatura”. Além desse fato há a

possibilidade que para um protocolo Y a referência temperatura seja representada pelo caractere ‘t’. O que ocasiona a possibilidade de adaptar referências já utilizadas por fabricantes de componentes, que não precisarão reimplementar totalmente seus códigos.

Elementos Opcionais:

- `<about>`: Possui uma descrição sobre o componente;
- `<min>`: É o valor mínimo que o dado referente pode assumir;
- `<max>`: É o valor máximo que o dado referente pode assumir.

3.4.2 Servidor para Físico

A comunicação do fluxo servidor para o físico é efetuada de forma dedicada, pois além do servidor conhecer o protocolo de interação que o nó de controle utiliza, poucos dados são necessários para requisitar as ações de execução.

A comunicação local cliente de cada nó de controle executa ações de requisição, para tanto se utiliza envelopes de requisição, que contém o endereço da interface de comunicação para destino, a identificação do nó e do componente ao qual o envelope é enviado e a ação que será executada. Esta ação determina o tipo de mensagem que há no conteúdo do envelope: envelopes de requisição de resposta utiliza-se a ação GET, representada pelo caractere ‘g’; envelopes de requisição de ação utiliza-se a ação PUT, representada pelo caractere ‘p’. A partir do conhecimento da ação há outras informações específicas que compõem o envelope, como demonstram as Tabelas 3 e 4, onde a Tabela 3 é referente ao conteúdo do envelope de requisição de ação e a Tabela 4 é referente ao conteúdo do envelope de requisição para resposta.

Tabela 3 - Envelope de Requisição de Ação

Informações	Descrição
Endereço	Informação do endereço da interface de comunicação destinada
Identificador do Nó	Informação que identifica qual nó é visada a ação
Identificação do Componente	Informação que identifica qual componente é visada a ação
Ação	Informa que a ação realizada será de PUT
Atributo visado	Informa qual atributo vai ser afetado pela ação
Valor	Informa o novo valor assumido pelo atributo visado

Fonte: do Autor.

Os únicos atributos que podem ser visados no envelope de requisição de ação são os dados dos atuadores e o status de funcionamento do componente, mais especificamente a operação do funcionamento dos atuadores e a informação se o componente deve ou não ser monitorado.

No caso do envelope de requisição de resposta os atributos visados podem assumir dois tipos de valores que determinam se serão retornados todos os dados ou se vão ser retornados apenas os dados simplificados.

Tabela 4 - Envelope de Requisição de Resposta

Informações	Descrição
Endereço	Informação do endereço da interface de comunicação destinada
Identificador do Nó	Informação que identifica qual nó é visada a ação
Identificação do Componente	Informação que identifica qual componente é visada a ação
Ação	Informa que a ação realizada será de GET
Atributos visados	Informa quais campos serão retornados ao servidor

Fonte: do Autor.

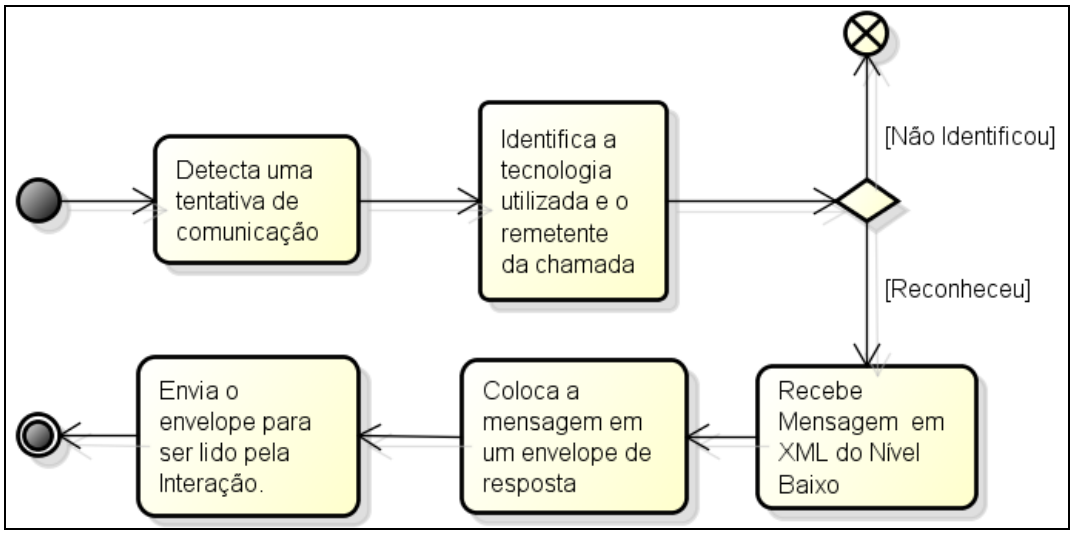
3.5 CAMADA DE NÍVEL INTERMEDIÁRIO

Essa é a camada responsável por tornar transparente a utilização dos dispositivos computacionais, concentrando a maior parte das informações e decisões que afetam a residência. Em suma, são tratados os protocolos que unem a camada física com a cliente, abstraindo os dispositivos físicos em representações lógicas, além de abstrair as funcionalidades para serem interpretadas por ambos os níveis, agindo de certa forma como o "*Device Driver*" do Sistema Operacional.

3.5.1 Comunicação local Servidor

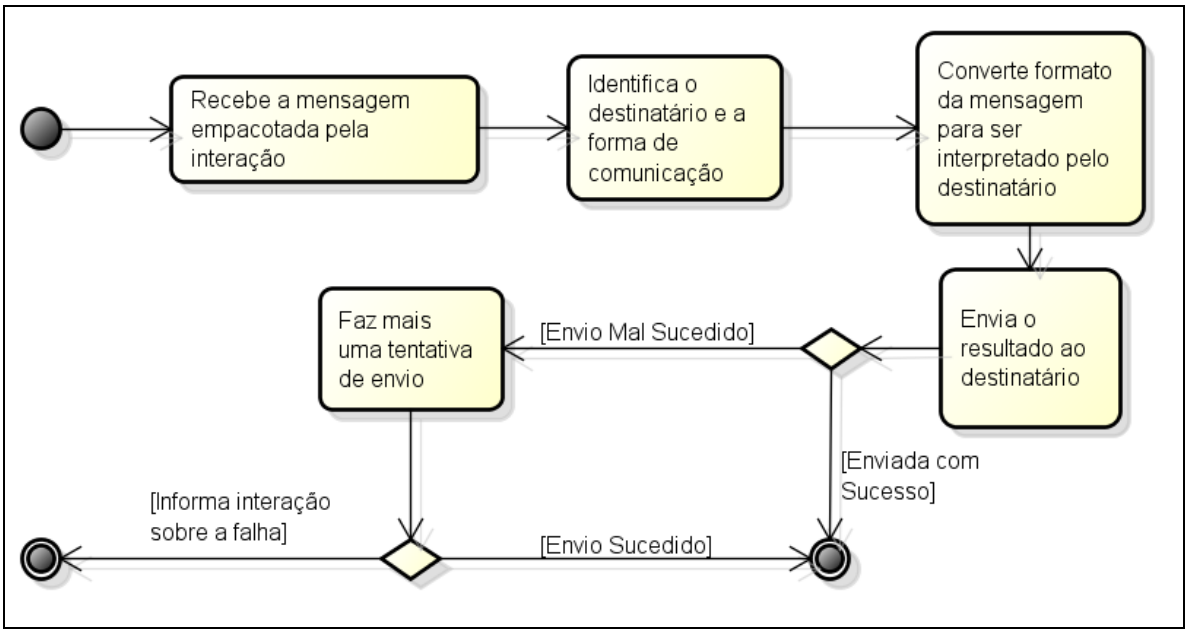
Módulo responsável por realizar a comunicação com a camada de nível baixo. Ao realizar a comunicação, esse módulo escolhe o método ou forma para se comunicar com o dispositivo físico, ou seja, formas de comunicação, como por exemplo, RS232, USB (*Universal Serial Bus*), Ethernet, Zigbee, entre outras; ao ser realizada a comunicação, envia a informação recebida para o módulo de interação e vice-versa, como representam as Figuras 16 e 17;

Figura 16 - Comunicação Local do Servidor, processo ao receber dados da Camada Física.



Fonte: do Autor.

Figura 17 - Comunicação Local do Servidor, processo ao enviar dados da Camada Física.



Fonte: do Autor.

3.5.2 Interação

O módulo de interação é responsável por traduzir a mensagem em XML que o módulo de comunicação local servidor recebeu, ou codificar a mensagem que será enviada aos controladores. Ou seja, este é um módulo transitório, no qual é identificado o protocolo de interação que a mensagem está utilizando, o dispositivo, o tipo do valor e o próprio valor

transportado. Essa rotina é valida tanto para codificação como decodificação das mensagens. Por exemplo, é preciso enviar uma ordem de ligar a luz. Ao saber disso, primeiramente o módulo de interação a versão do modelo XML e identifica o protocolo de interação que a mensagem recebida utiliza, em seguida busca o identificador do dispositivo que irá executar a ordem (necessário, pois em um nó de controle é possível ter diversos dispositivos atuadores e sensores), o tipo de dado ordem, e o valor correspondente a ação de ligar a luz, ao encontrar essas informações coloca-as em um envelope e envia para a comunicação local do servidor. A mesma rotina é valida para o sentido inverso (do nível baixo para o nível alto). Por exemplo, a temperatura registrada em um sensor chegou a 25ºC, o nó de controle envia uma mensagem contendo o protocolo que utiliza a identificação do sensor, do tipo de valor (que é de temperatura em graus Celsius) e da temperatura registrada que é 25; coloca essas informações em um envelope de resposta dentro da comunicação local do Servidor que o encaminhará para a Interação realizar sua rotina. Uma vez realizada essa rotina o módulo de interação envia o resultado para o próximo módulo. As Figuras 18 e 19 representam os processos do módulo de Interação.

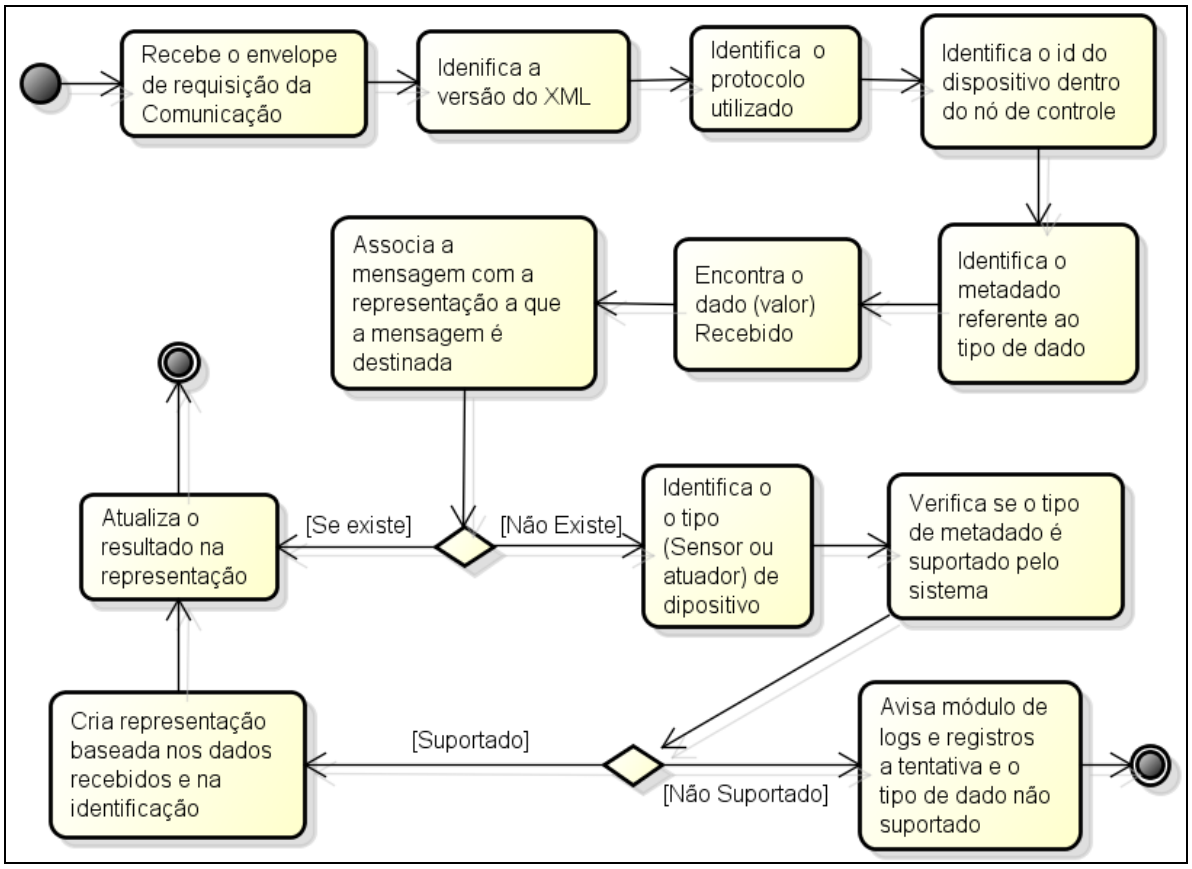
Para realizar tais tarefas é implementado neste módulo o protocolo de interação, que contêm as referências dos valores previsíveis ao qual são utilizados para codificar e decodificar as mensagens. Sendo assim, é possível transportar mensagens com tamanho menor. Este protocolo contempla, como demonstra a Tabela 5, as referências de: atividade, tipo de valor, atributo visado, dado e dispositivo suportado.

Tabela 5 - Referências do Protocolo de Interação contém

Referências	Referências
Atividade	Das atividades atuador e sensor
Tipo de valor	Dos tipos de valores suportados, exemplo: Boolean, Double e Integer
Atributo Visado	Dos atributos que podem ser visados, exemplo: Dado e Status
Dado	Dos dados previsíveis, como exemplo, para Booleano os valores true e false
Dispositivo	Dos tipos de dispositivos referentes ao componente. Exemplo: LED e Motor

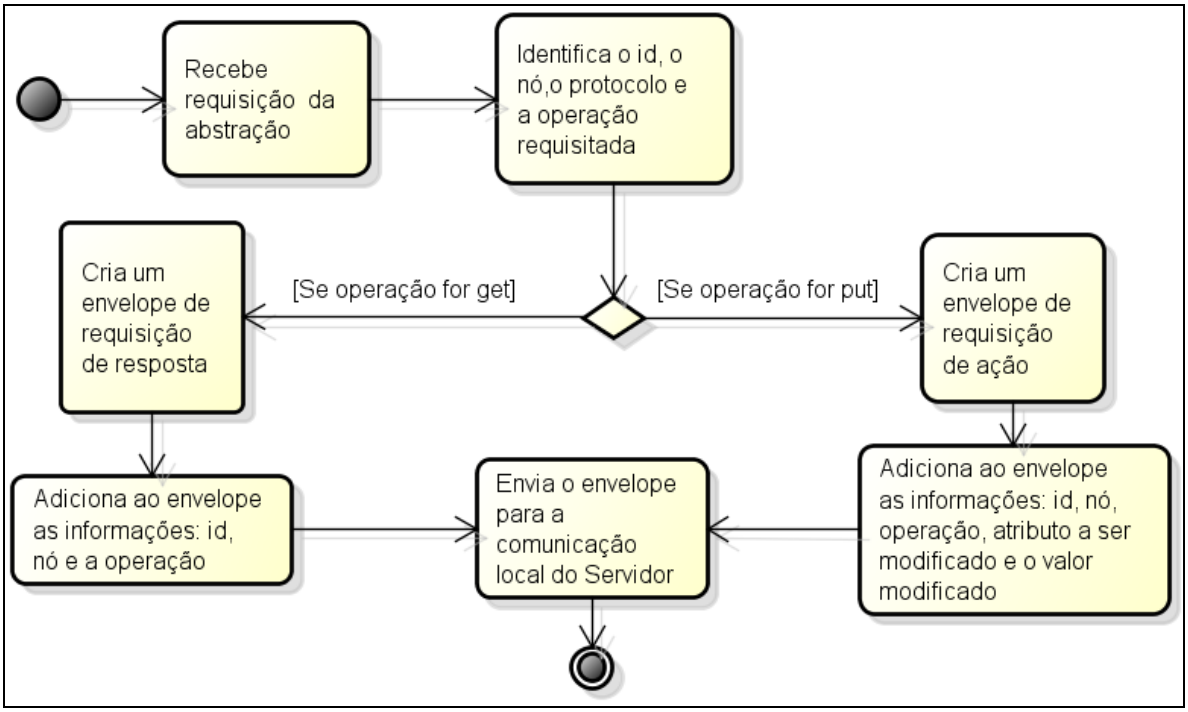
Fonte: do Autor.

Figura 18 - Processo da Interação ao receber o envelope da Comunicação Local Servidor.



Fonte: do Autor.

Figura 19 - Processo da Interação ao enviar o envelope para a Comunicação Local Servidor.



Fonte: do Autor.

3.5.3 Abstração

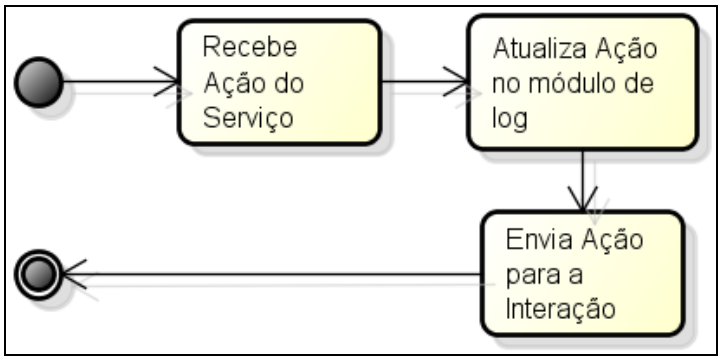
O módulo de abstração gerencia a representação lógica dos dispositivos sensores e atuadores individuais. Essa representação contém o dado atual monitorado pelos sensores e no caso dos atuadores além do dado atual referente a sua operação, fornece as opções de ação que o dispositivo pode realizar. Para tanto recebe do módulo de Interação ou de serviço às mensagens com os valores para agir sobre este. Esta representação é interpretada pela camada superior, diferente da mensagem pura, por isso a necessidade de relacionar esta mensagem com a representação lógica.

Uma representação lógica pode ou não estar associada a um serviço. Se não estiver, ela fica em uma lista a parte, a lista de representações não associadas, no qual apenas mostra que existe permitindo que seja associada futuramente em algum serviço.

Registro e logs: Este módulo trabalha diretamente com a representação lógica. Em suma, sempre que a representação lógica for modificada, antes da modificação será salvo o estado atual do objeto, essa informação será gravada em um banco de dados, relacionando sempre a representação visada, que camada originou a ordem (usuário ou hardware), a ação realizada, e o valor modificado pelo evento ocorrido.

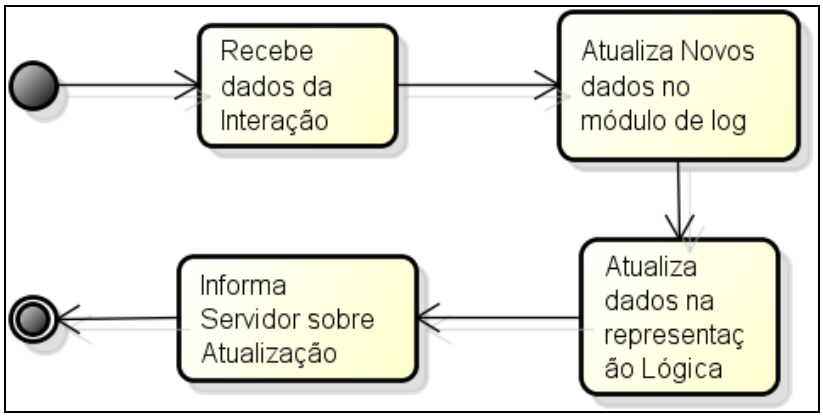
As atividades realizadas pelo módulo de abstração podem ser encontradas nas Figuras 20 e 21.

Figura 20 - Atividade das Representações Lógicas do módulo de Serviço para o de Abstração



Fonte: do Autor.

Figura 21 - Atividade das Representações Lógicas do módulo de Interação para o de Abstração



Fonte: do Autor.

3.5.4 Serviços

Este módulo agrupa as representações lógicas em serviços, relaciona-os por área física ou funcionalidades em comum. Sempre que houver alguma alteração na leitura do dispositivo, o seu serviço testará esta alteração no módulo de condições e com o resultado das condições toma uma decisão através do módulo de decisão, como demonstram os diagramas de atividade da Figura 22, onde a Figura 22a apresenta essa atividade quando for de origem do usuário e na Figura 22b a atividade quando a representação lógica for alterada. Cada serviço executa de forma independente;

Condições: O módulo de condições possui os gatilhos que executarão as decisões, por exemplo: se o ambiente ficar frio demais ou quente demais ele irá executar as decisões. Uma condição que deve ser considerada é a de tempo para atualizar dados da representação lógica, no qual sempre que esse intervalo de tempo expirar é enviado uma solicitação para atualizar as representações lógicas do serviço;

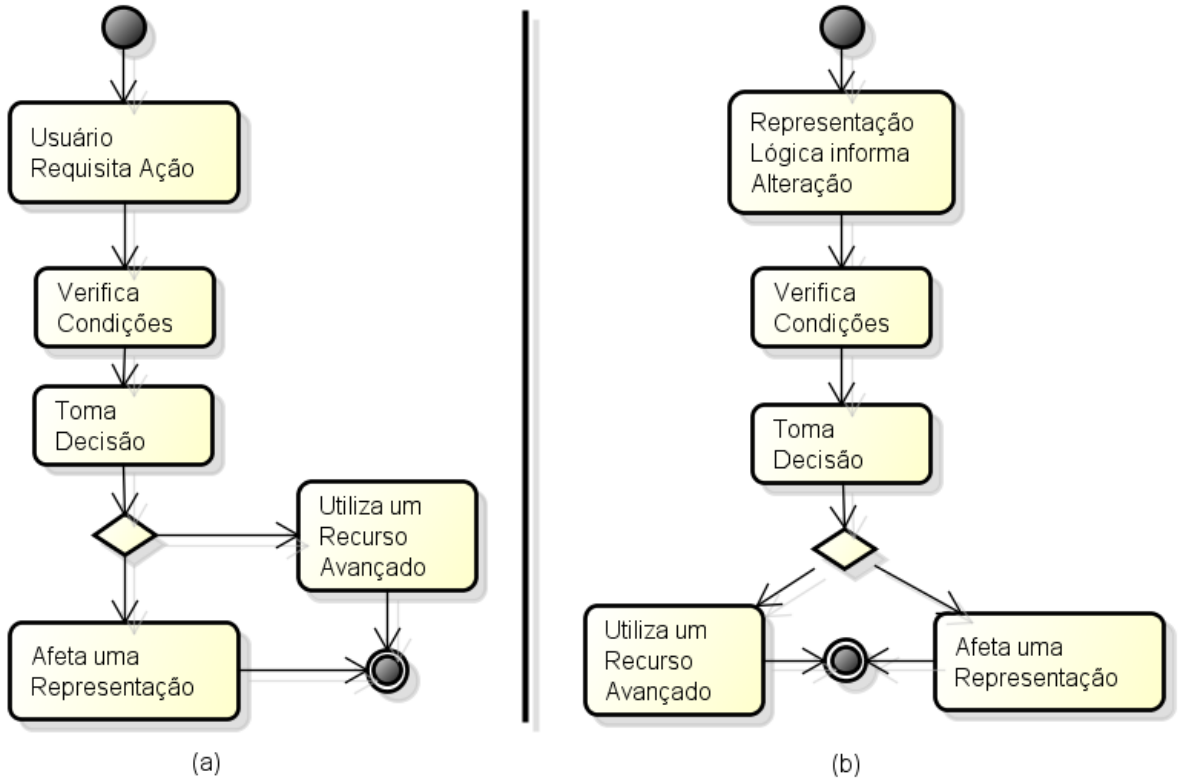
Decisões: Uma vez atendida uma condição, o módulo de decisões será acionado, podendo executar ações que podem afetar tanto outras representações lógicas como executar recursos avançados.

As ações que são possíveis para as representações lógicas: alterar o estado, alterar a operação, requerer retorno de dados, requerer atualização.

Ações de recursos avançados disponíveis: envio de e-mail ou SMS para o Morador com a mensagem sobre algum evento ocorrido e calcular um resultado baseado em dados oriundos do hardware.

Por exemplo, quando a temperatura do aquário na sala de estar for muito baixa, os peixes poderiam morrer por causa da baixa temperatura, a condição de temperatura baixa é acionada e toma as decisões de ligar o aquecedor, que é representado logicamente no módulo de abstração, neste caso envia mensagens SMS e e-mails avisando o morador (que está no trabalho) sobre o que ocorreu e que os peixes ficarão bem.

Figura 22 - Funcionamento do módulo de Serviços



Fonte: do Autor.

3.5.5 Usuários

Este é o módulo de software responsável pelo controle de acesso e por gerenciar todas as preferências, configurações e acessos dos usuários do sistema doméstico. O ideal é que sejam considerados no mínimo dois tipos de usuários: Anônimo ou Visitante que é utilizado caso não seja conhecido o usuário e o Morador que possui as configurações e preferências pessoais dele próprio. Além desses, há um usuário à parte, o Administrador, que deve ser tratado como um usuário restrito, pois possui acesso das alterações críticas do sistema, cabendo a ele a responsabilidade de gerir os usuários.

Cada usuário pode visualizar seus *logs* de acesso anteriores e executar ações no sistema baseado em suas permissões descritas no seu perfil, as quais ditam o que ele pode ou não acessar, visualizar ou editar. Podem-se dividir as permissões de acesso em: serviço e representação lógica.

Permissões sobre o Serviço: contempla a possibilidade de visualizar, editar, excluir ou incluir as informações individuais de cada serviço, bem como dos componentes nele contidos, aos quais são referentes às condições, decisões e a lista de representações lógicas a qual atua, como demonstra a Tabela 6.

Tabela 6 - Permissões sobre o Serviço

Componente	Ações possíveis
Gerência de Condições	Implica na adição, visualização, alteração e exclusão das condições contidas nos serviços, também relacionadas às representações lógicas.
Gerência de Decisões	Implica na adição, visualização, alteração e exclusão das decisões contidas nos serviços, também diretamente associadas às condições e as representações.
Lista de Representações Lógicas	Implica na adição, visualização, alteração e exclusão das representações lógicas no serviço, também implica se é possível ver que condições e decisões são relacionadas a esta.

Fonte: do Autor.

Permissões sobre a representação lógica: contempla a possibilidade de visualizar, editar, excluir ou incluir as informações individuais de cada representação, incluindo o acesso as ações que podem executar. Ações demonstradas e detalhadas pela Tabela 7.

Tabela 7 - Permissões sobre a Representação Lógica

Ação	Descrição
Inclusão	Implica na inclusão da representação lógica em um serviço.
Alteração	Implica na alteração de informações, estado de monitoramento e, no caso dos atuadores, também a troca de operação.
Visualização	Implica na possibilidade de ver os dados monitorados pela representação, bem como seus *logs* anteriores.
Exclusão	Implica na exclusão lógica no serviço, assim o hardware não será monitorado pelo serviço, mas continuará a funcionar normalmente.

Fonte: do Autor.

3.5.6 Comunicação Aplicação Servidor

Módulo responsável por gerenciar a comunicação do Servidor. Nele é definida a tecnologia que será empregada na comunicação entre aplicação cliente e servidor, podendo ser, por exemplo: Socket, WebService, Ethernet e HttpRequest. Por questões de segurança, em algumas formas de comunicação é necessário considerar o uso de criptografia.

3.6 COMUNICAÇÃO DE APLICAÇÃO ENTRE CLIENTE E SERVIDOR

A interação do nível Alto com o nível Intermediário ocorre em quatro casos: Ao efetuar autenticação de usuário para acessar o sistema; Ao sair do sistema; Quando o nível alto observa os dados dos dispositivos lógicos presentes no nível intermediário e; Quando envia ordens ao nível intermediário, baseadas no que os serviços e dispositivos dispõem.

3.7 CAMADA CLIENTE

A camada cliente tem a finalidade de possibilitar que diversos tipos de clientes consigam acessar o servidor por plataformas distintas. Para tanto são descritas nas próximas seções os módulos que possibilitam tal finalidade.

3.7.1 Comunicação Aplicação Cliente

Módulo responsável por gerenciar a comunicação do Servidor. Nele é definida a tecnologia que será empregada na comunicação com o servidor, devendo ser compatível a que o servidor utilize, por exemplo: Socket, WebService, Ethernet e HttpRequest.

3.7.2 Aplicação

A aplicação é o módulo de software que contém a interface ao qual o usuário poderá acessar suas preferências, configurações e interagir com os dados, ações dos serviços e componentes que estão em funcionamento. Para tanto, são implementados todos os recursos visuais para facilitar a interação com o usuário final. Por exemplo: aplicações feitas em HTML. Android, Java, Iphone, entre outras.

4 DESENVOLVIMENTO DO PRÓTOTIPO

Neste capítulo é detalhado o desenvolvimento do protótipo a partir do modelo conceitual, bem como considerações sobre o desenvolvimento e as dificuldades encontradas.

4.1 FERRAMENTAS UTILIZADOS

Ferramentas de Modelagem: Foram utilizadas as ferramentas de modelagem UML Astah Community para construir os diagramas de atividade e o diagrama de classes; para construir algumas das imagens foi utilizada a ferramenta Fireworks, versão Trial, e para construir as imagens contendo o resultado sobre as camadas fora utilizada a ferramenta Pencil.

Ferramentas para Desenvolvimento: Todo o desenvolvimento foi baseado nas linguagens Java, Android, JSP, HTML, XML e C++, para tanto se utilizou as IDEs Arduino para C++ e NetBeans para as demais linguagens. A comunicação ente as aplicações cliente e servidora foi feita por uma Web Service SOAP hospedada pelo servidor de aplicação GlassFish Server 3.1.2.1.

4.2 MATERIAIS UTILIZADOS

Nesta seção serão expostos os materiais utilizados para o desenvolvimento do protótipo, bem como algumas características dos mesmos, baseado nas informações disponibilizadas pelo WEBTRONICO (2012), sendo elas:

- *Placa arduino Duemilanove 2009:* Placa com todos os componentes necessários para que o microprocessador Atmega232p execute o sistema embarcado instalado. Este modelo disponibiliza em soquetes 14 terminais de entrada e saída digital, 6 entradas analógicas, um cristal oscilador de 16 MHz, uma conexão USB (utilizada para gravar o sistema embarcado no microcontrolador), uma entrada para alimentação, um cabeçalho ICSP, um botão de reset, saída com 5 volts de alimentação (proveniente do regulador de tensão acoplado na placa), saída de 3,3 volts fornecida pelo chip FTDI e saídas GND. Este arduino pode ser alimentado por uma fonte externa com plug de 2,1mm (centro positivo) com saída recomendada de 7 a 12 volts
- *Arduino Ethernet Shield V1.1*: Trabalha diretamente com a placa arduino utilizando também a mesma fonte de alimentação. Usa o chip ENC28J60 da microchip e já vem pronto com conector RJ45 para cabo de rede.

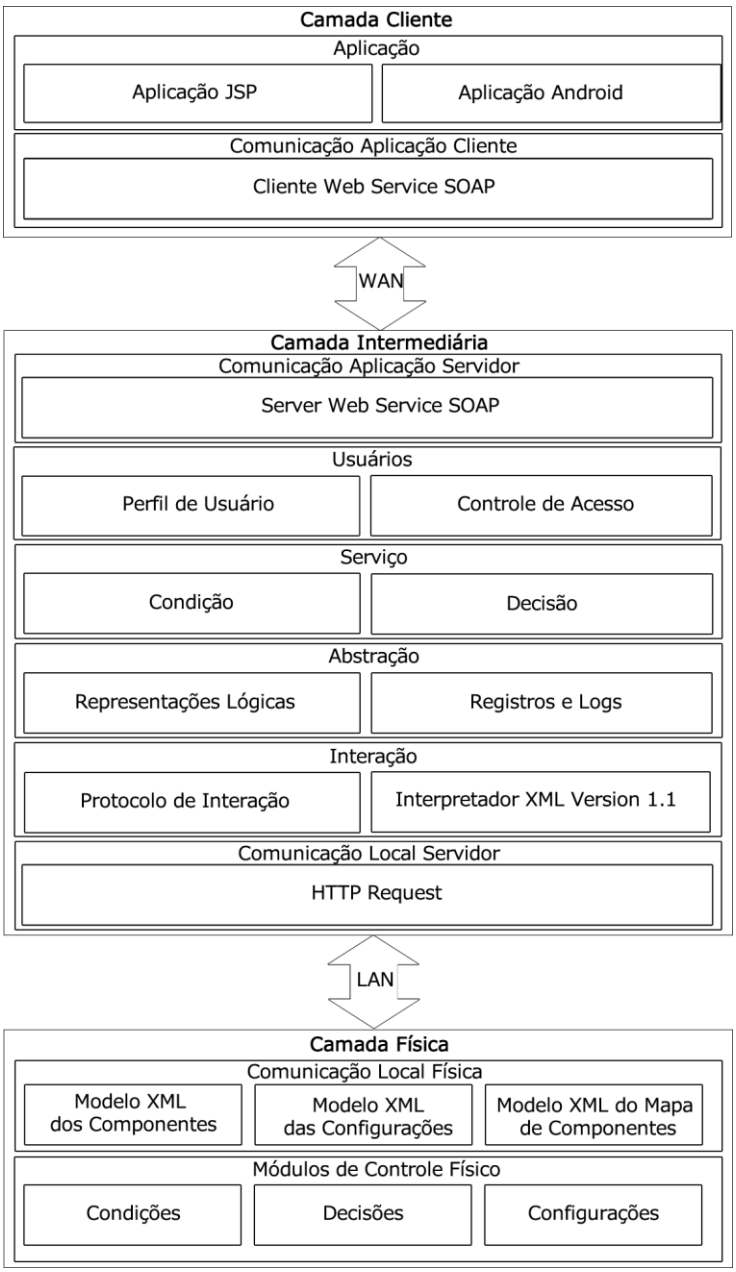
- Vinte unidades de jumper: Os quais são fios com terminais metálicos que permitem fazer a ligação do soquetes presentes na placa arduino com os componentes.
- Dois *LEDs:* Um LED é um diodo emissor de luz, que ao ser alimentado no terminal positivo e o terminal negativo ligado no GND emite luz. Pelo motivo do LED consumir pouca energia, a própria placa do arduino o alimenta.
- *Interruptor magnético:* É um interruptor magnético normalmente utilizado em alarmes. Utiliza a alimentação da saída 5 volts da placa arduino.
- *Fonte de alimentação*: Uma fonte com entrada 110v/220v w saída de corrente contínua de 12 volts com um plug de 2,1mm (centro positivo). Utilizado para alimentar o arduino.
- *Protoboard*: Para a montagem do protótipo utilizou-se uma *Protoboard* com 840 furos. Na *Protoboard* foram colocados os componentes LED e o interruptor magnético e utilizando os jumpers esses componentes foram ligados nos terminais da placa arduino, para assim serem controlados e alimentados.

4.3 ARQUITETURA PROPOSTA

Baseado na arquitetura conceitual abordada no capítulo anterior é proposto para o presente trabalho o desenvolvimento de um protótipo com ênfase na interação entre as camadas intermediária e física. A Figura 23 demonstra de forma geral a abordagem do desenvolvimento que nas próximas seções será detalhada.

Em resumo, o protótipo gerencia 3 dispositivos: um sensor de interrupção magnético ao qual interage com outro dispositivo, e dois atuadores LED, as quais um tem a função de piscar quando o servidor emitir a ordem a ele e o outro liga ou desliga dependendo se o interruptor estiver ativado ou não. Todos esses componentes interagirão com o servidor, atualizando suas respectivas representações lógicas ao qual o usuário poderá acessar e visualizar quais os estados dos componentes, podendo encaminhar ordens para os atuadores.

Figura 23 - Arquitetura Proposta



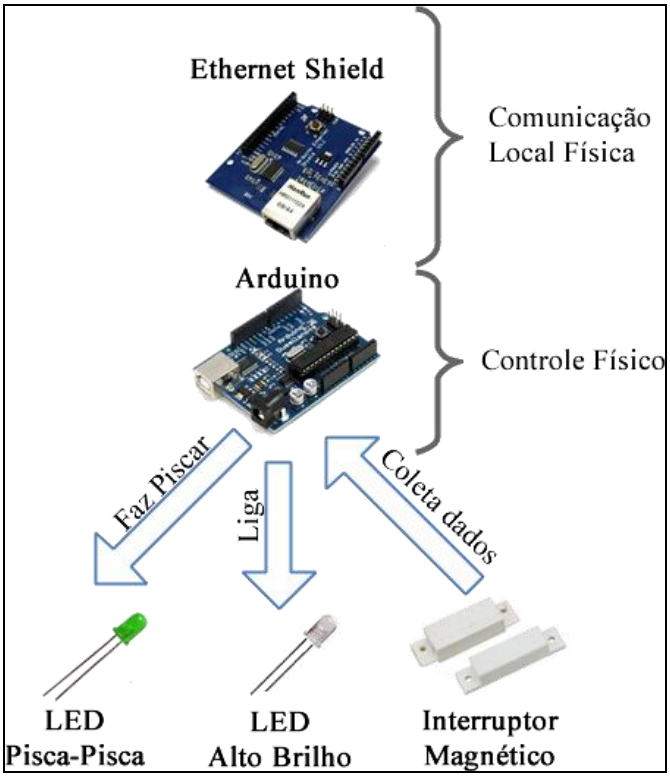
Fonte: do Autor.

4.3.1 Camada Física

Nesta seção serão apresentados todos os métodos e resultados obtidos no decorrer do desenvolvimento desta camada. Serão comentados separadamente os módulos de comunicação local físico e o controle físico. A Figura 24 demonstra graficamente a estrutura que será abordada nessa seção, onde na CLF (Comunicação Local Física) usando Shields

Ethernet são expostos os comandos que executam as ações que modificam ou retornam os XMLs dos dispositivos, configurações e mapa. Sobre o controle físico será mostrada a estrutura que o protótipo usou para armazenar os dispositivos, a estrutura do protocolo de interação utilizado, bem como as condições e decisões que podem ser executadas.

Figura 24 - Estrutura proposta para a Camada Física



Fonte: do Autor.

Comunicação Local Física (CLF): Este módulo utiliza o padrão Ethernet provido pela Shield Ethernet acoplada no arduino, ao qual aceita conexões pela porta 80, possibilitando assim, a interação com o módulo de comunicação local servidor por Http Request, para tanto foram instaladas na memória do micro controlador do arduino as bibliotecas *"etherShield.h"* e *"ETHER_28J60.h"*, para que possa executar a Shield Ethernet. Ao ser acessado é possível executar as ações de: retorno dos dados completos e simplificados; dos componentes; retorno das configurações; retorno do mapa do nó de controle; e alterar os estados de funcionamento dos componentes. Para executar essas ações são passados por parâmetro, junto ao endereço IP, os valores que os executarão. Sua estrutura é sempre baseada nos envelopes da arquitetura conceitual e são influenciados diretamente pelo protocolo de interação, pois para funcionar corretamente, tanto o módulo de interação presente na camada intermediária, como o módulo

de configuração da camada física deve utilizar o mesmo protocolo de interação. A relação das ações e parâmetros que o protótipo executa, bem como suas descrições pode ser encontrada na Tabela 8.

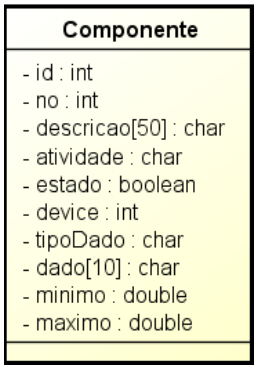
Tabela 8 - Ações que a Comunicação Local Física do Protótipo pode executar

Parâmetro	Ação
http://10.5.99.191/c	Retorna o XML com todos os dados de configuração, vide Figura 37 di Apêndice B
http://10.5.99.191/m	Retorna o XML com o mapa dos dados componentes, vide Figura 38 di Apêndice B
http://10.5.99.191/11ga	Retorna o XML com todos os dados do componente com id 1, vide Figura 39 di Apêndice B
http://10.5.99.191/11gs	Retorna o XML com os dados simplificados do componente com id 1, vide Figura 40 di Apêndice B
http://10.5.99.191/12ga	Retorna o XML com todos os dados do componente com id 2, vide Figura 41 di Apêndice B
http://10.5.99.191/12gs	Retorna o XML com os dados simplificados do componente com id 2, vide Figura 42 di Apêndice B
http://10.5.99.191/13ga	Retorna o XML com todos os dados do componente com id 3, vide Figura 43
http://10.5.99.191/13gs	Retorna o XML com os dados simplificados do componente com id 3, vide Figura 44 di Apêndice B
http://10.5.99.191/11pst	Atualiza o estado do componente, onde o “t” simboliza ativo e o “f” inativo.
http://10.5.99.191/12psf	Atualiza o estado do componente, onde o “t” simboliza ativo e o “f” inativo.
http://10.5.99.191/13pst	Atualiza o estado do componente, onde o “t” simboliza ativo e o “f” inativo.
http://10.5.99.191/13pdt	Atualiza a operação do atuador, onde o “t” simboliza ativo e o “f” inativo.
http://10.5.99.191/13pdf	Atualiza a operação do atuador, onde o “t” simboliza ativo e o “f” inativo.

Fonte: do Autor.

Controle Físico: O desenvolvimento desse módulo utilizou como ponto de partida a busca por uma forma de armazenar os dados dos diversos dispositivos na memória do sistema embarcado e que ainda permita uma fácil expansão da quantidade desses dispositivos presentes no nó. A partir dessas premissas fora avaliado o uso de estruturas e classes, sendo escolhida a classe para a implementação devido ao fato de se comportar melhor quando se tratava de dados char*, que necessitavam um tratamento diferenciado que fora feito pelos métodos dos objetos. A Classe Componente contendo os atributos utilizados na implementação pode ser encontrada na Figura 25.

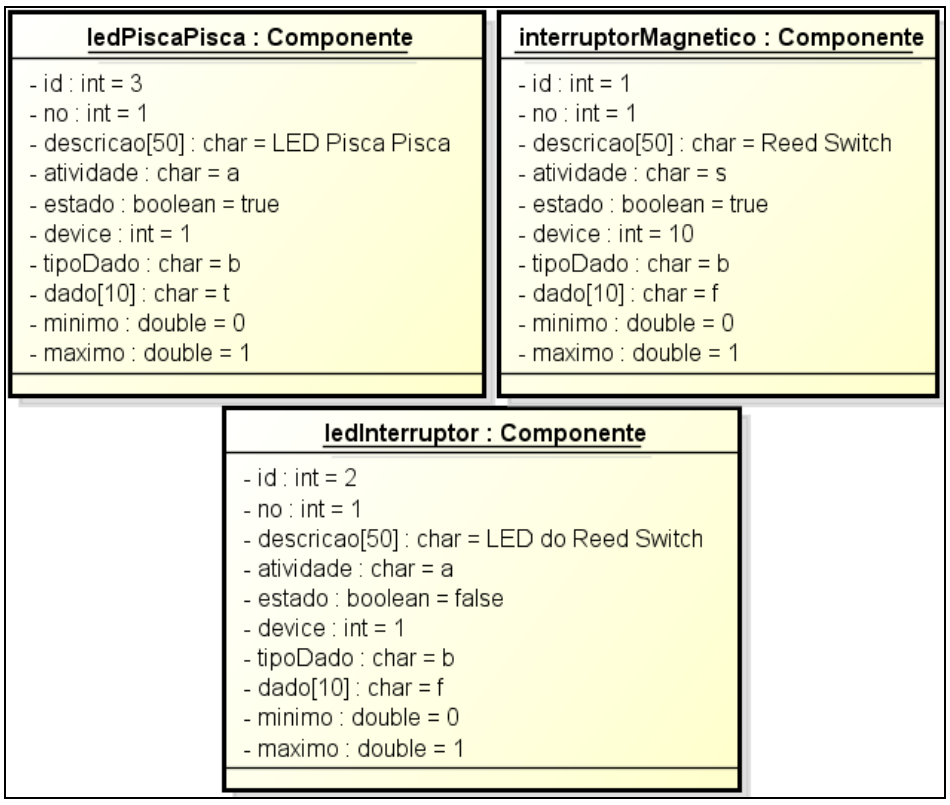
Figura 25 - Classe Componente utilizada na implementação da Camada Física



Fonte: do Autor.

Cada componente possui dados próprios, como pode ser observado no Diagrama de Objetos na Figura 26, verifica condições e executa decisões. Seque abaixo a lista dos componentes gerenciados pelo controle físico:

Figura 26 - Diagrama de Objetos contendo os dados iniciais dos componentes



Fonte: do Autor.

- O primeiro componente é o interruptor magnético, vulgo *reed switch*, que está conectado em um terminal que fica monitorando o seu comportamento, enquanto o

imã estiver adjacente a ele, o nível lógico monitorado será alto, quando ele for afastado, a chave interna do interruptor é desconectada fazendo com que o dado monitorado passe a ser um nível lógico baixo, ligando assim um Componente LED;

- O segundo componente é um LED destinado a ser ativado sempre que o Componente Interruptor for desativado;
- O terceiro componente é um LED destinado a Piscar enquanto o programa estiver em funcionamento. A única maneira de fazê-lo parar de piscar é pelo servidor.

Observando o funcionamento desses componentes pode-se concluir o controle físico do protótipo é parcialmente dependente do servidor, pois o interruptor magnético é independente do servidor, podendo interagir sem problemas com o LED ao qual controla, caso a comunicação com o Servidor estiver inativa. Por outro lado o LED pisca-pisca estará sempre ativado ou desativado, pois o único que pode alterar seu estado é o servidor.

Figura 27 - Valores de referência do Protocolo de Interação no nó de controle

```
////////////// Protocolo Default ////////////
////// Atividades
#define SENSOR   's'
#define ATUADOR  'a'
////// Estados
#define LIGADO   true
#define DESLIGADO false
#define VALORLIGADO 't'
#define VALORDESLIGADO 'f'
////// Tipos de componente - Chaves em INT
#define LED 1
#define TEMPERATURA 2
#define CAMERA 3
#define LAMPADA 4
#define INTERRUPTOR 5
#define INFRAVERMELHO 6
#define MOTORDC 7
#define MOTORAC 8
#define MOTORDEPASSO 9
#define INTERRUPTORMAGNETICO 10
////// Tipos de dados
#define INTEIRO   'i'
#define NUMERICO  'n'
#define BOOLEANO  'b'
#define STREAM    's'
////// Dado - para Booleano
#define ATIVO   "t"
#define INATIVO "f"
```

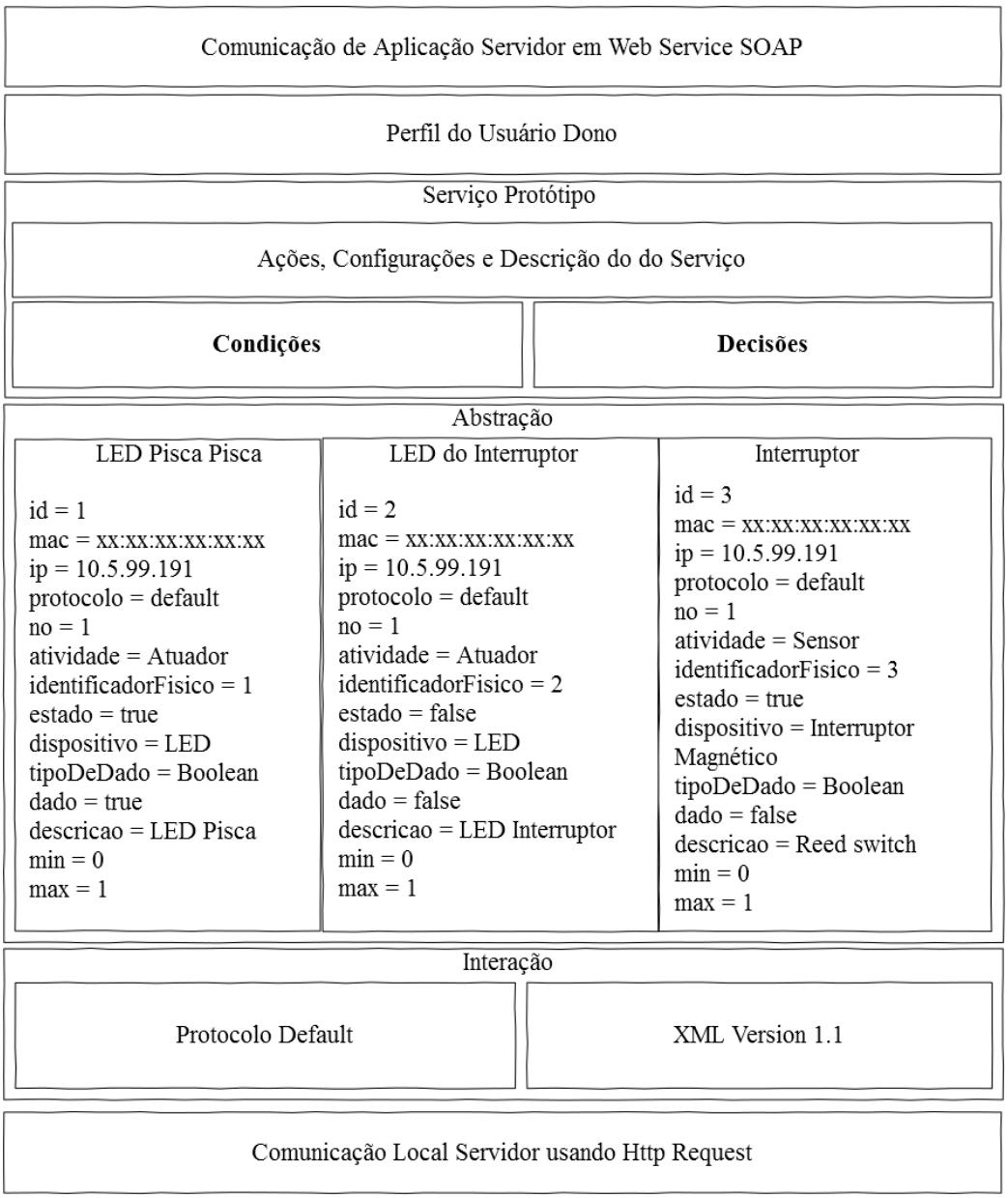
Fonte: do Autor.

Para que a camada intermediária consiga interagir com sucesso com a camada física, ambas tem de utilizar o mesmo protocolo de interação, para que assim possam traduzir com sucesso as mensagens transportadas. O código referente ao protocolo de interação utilizado pela camada física está representado pela Figura 27.

4.3.2 Camada Intermediária

Nesta seção serão comentados separadamente todos os resultados do protótipo proposto referente à camada intermediária. Para a proposta fora desenvolvido um Diagrama de Classes Geral, encontrado na Figura 36 di Apêndice A.

Figura 28 - Estrutura proposta da Camada Intermediária

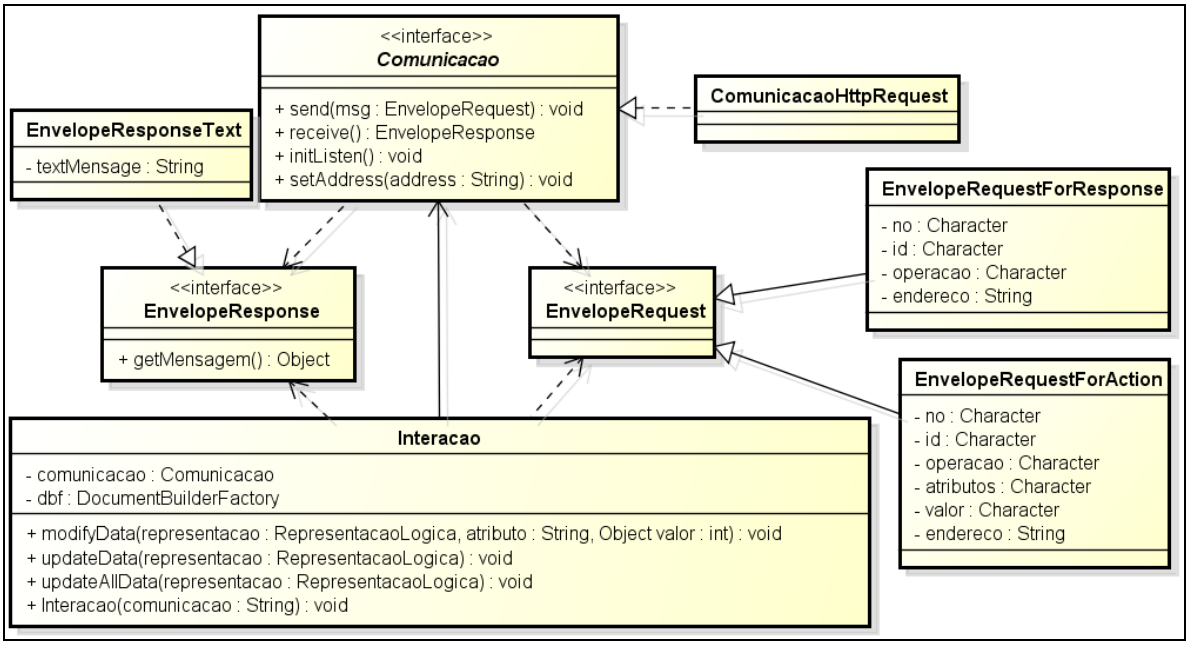


Fonte: do Autor.

A Figura 28 demonstra graficamente a estrutura que será abordada nessa seção, onde em síntese demonstra que a comunicação da aplicação servidora será feita por um Web Service SOAP; que o perfil de usuário utilizado será somente o Dono; que o Serviço que gerencia o protótipo terá ações pelo qual o usuário pode interagir com o sistema e também, estipula em "Condição e Decisão" que é feito o tratamento automático dos eventos ocorridos na execução do sistema; que a abstração contém as representações lógicas dos dispositivos conectados ao sistema, bem como alguns dados simulando os dados que os compõe; na interação tem-se o protocolo que é utilizado para integração dos dados e a versão do tratamento dos XMLs ao qual a camada intermediária receberá da camada física; e a comunicação local do servidor que utiliza requisições HttpRequest para enviar os envelopes de requisição e receber os dados.

Comunicação Local Servidor: Este módulo faz a comunicação com a camada física utilizando o método HttpRequest, para tanto recebe do módulo de interação envelopes de requisição para serem enviados a camada física e usa envelopes de resposta para receber os dados em formato XML. A Figura 29 mostra através de um diagrama de classes como foi implementado este módulo.

Figura 29 - Diagrama de classes proposto para a Comunicação Local Servidor

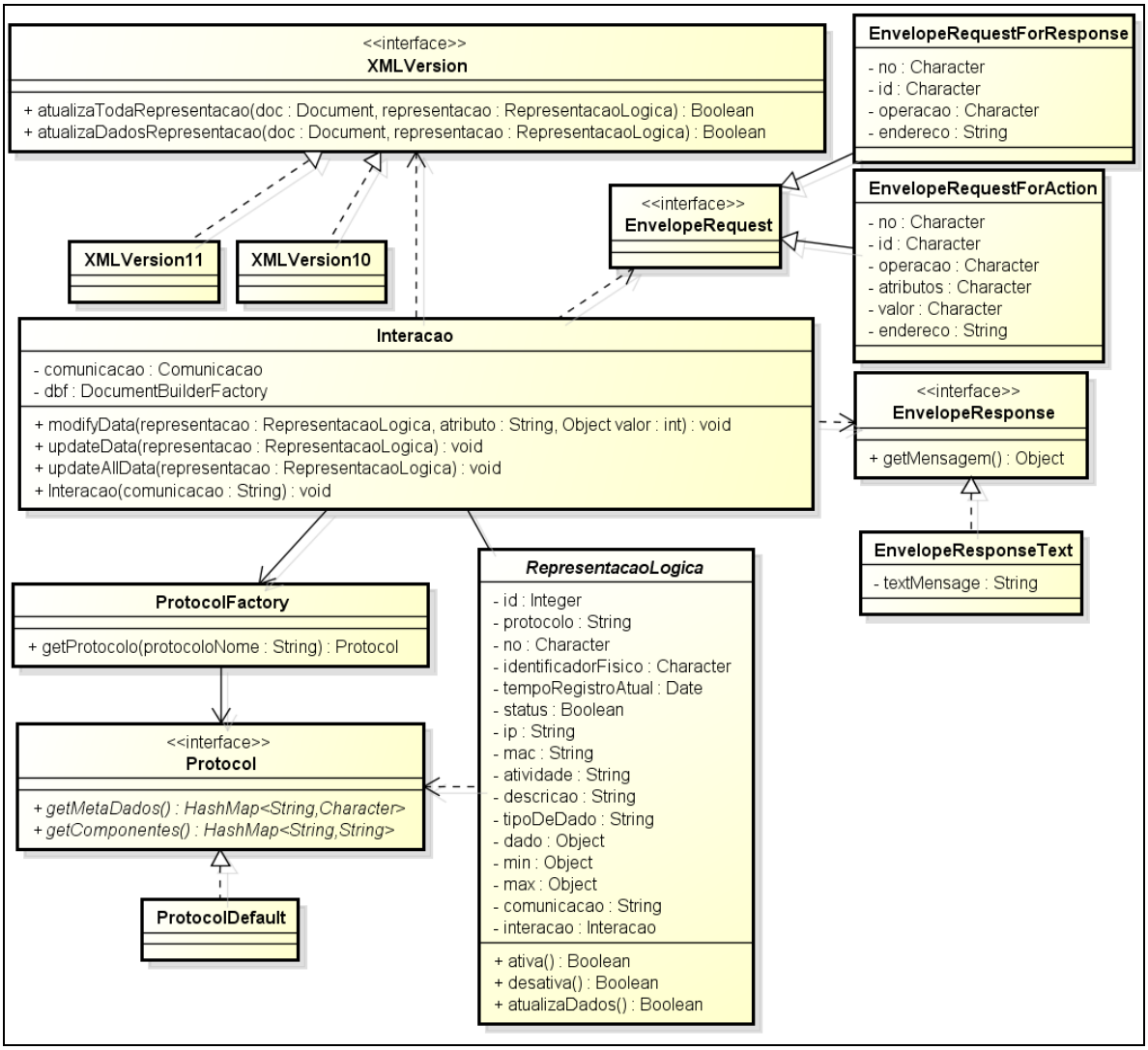


Fonte: do Autor.

Interação: Este módulo faz o tratamento das mensagens trafegadas entre as camadas intermediária e física com as devidas representações lógicas. Para tanto este módulo propõem

envelopes de requisição para requerer modificações e retorno de dados e um tratador das mensagens XML recebidas da comunicação local servidor através dos envelopes de resposta. E para fazer com que essas informações consigam ser interpretadas pela representação lógica é utilizado o protocolo de interação que traduz esses dados contidos nos envelopes. A Figura 30 mostra o diagrama de classes utilizado pela interação e a figura 31 exibe o protocolo de interação utilizado pelo protótipo.

Figura 30 - Diagrama de classes proposto para a Interação



Fonte: do Autor.

Abstração: Para o desenvolvimento da arquitetura proposta, a abstração faz o papel de representar logicamente os componentes presentes na camada física, em suma, utiliza valores de referência padrão do protocolo de interação para ser possível a interação do módulo de interação. Além disso, este módulo prevê que toda vez que uma atualização for realizada ou

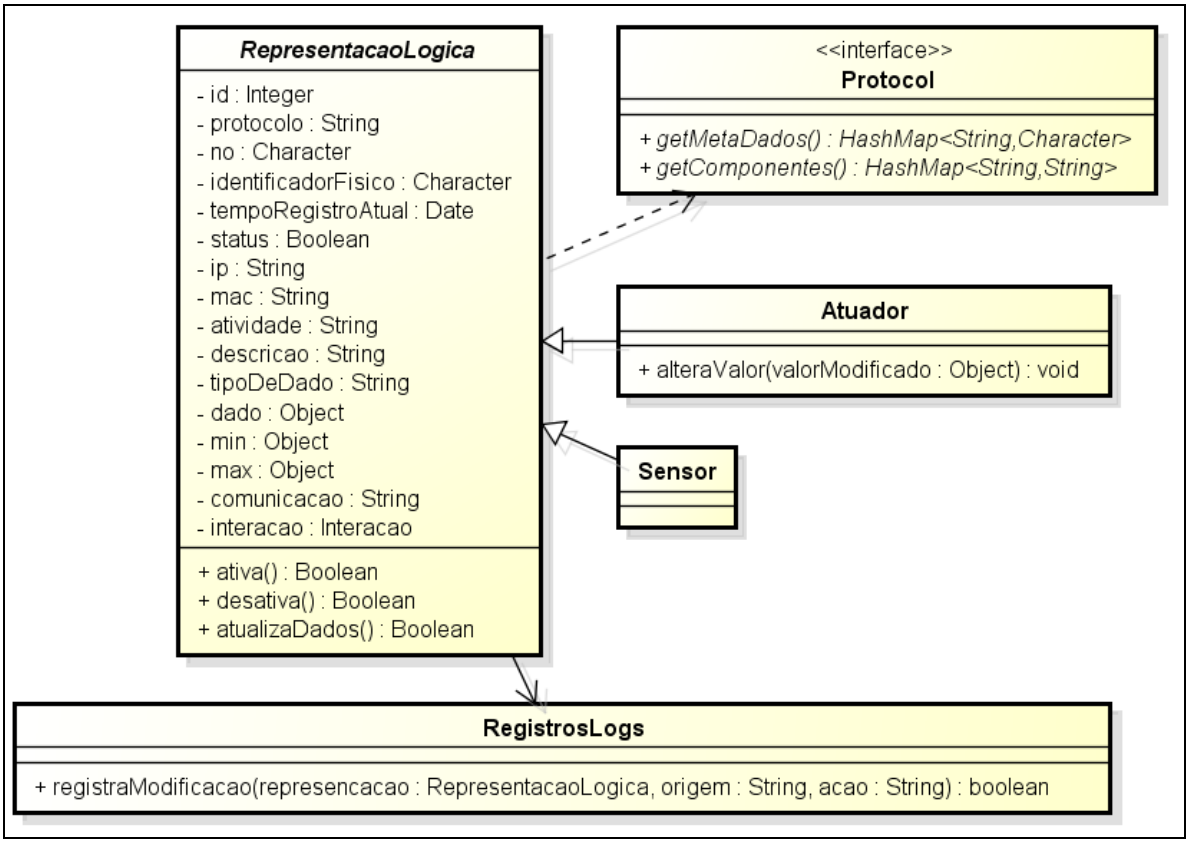
uma ação for executada, uma referência dessa ação deve ser gravada através da funcionalidade de registro de logs. Como demonstra o diagrama de classes presente na Figura 32, a representação lógica é uma classe abstrata, assim só pode ser instanciada como um de seus filhos, Sensor ou Atuador. A Figura 28 no módulo de abstração, exibe as representações lógicas que compõe este trabalho.

Figura 31 - Protocolo Default utilizado na proposta da Interação

```
// Metadados
metaDados.put(ATUADOR,'a');
metaDados.put(SENSOR, 's');
metaDados.put(TRUE.toString(), 't');
metaDados.put(FALSE.toString(),'f');
metaDados.put(BOOLEAN,'b');
metaDados.put(NUMERIC,'n');
metaDados.put(INTEGER,'i');
metaDados.put(STREAM,'s');
metaDados.put(DADO,'d');
metaDados.put(STATUS,'s');
// Dispositivos
dispositivos.put("1","LED");
dispositivos.put("2","Temperatura");
dispositivos.put("3","Câmera");
dispositivos.put("4","Lâmpada");
dispositivos.put("5","Interruptor");
dispositivos.put("6","Infravermelho");
dispositivos.put("7","Motor DC");
dispositivos.put("8","Motor AC");
dispositivos.put("9","Motor de Passo");
dispositivos.put("10","Interruptor Magnético");
```

Fonte: do Autor.

Figura 32 - Diagrama de classes proposto para a Abstração

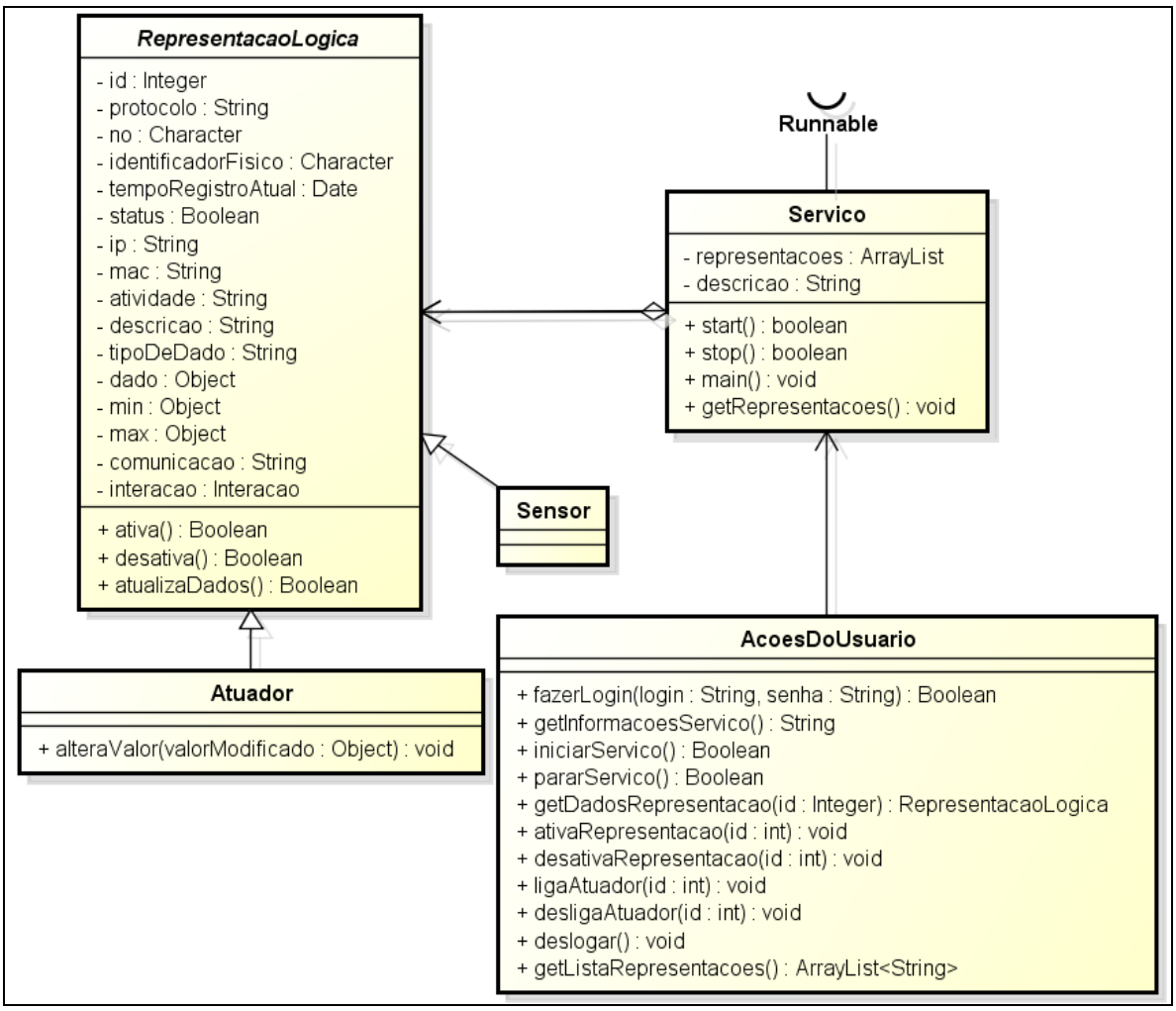


Fonte: do Autor.

Serviço: O módulo de serviço desempenha o papel de agregador e monitor das representações lógicas presentes no protótipo, como pode ser visualizado pela Figura 33. Desta forma, todas as representações agregadas a ele passam por um processo de avaliação das condições e tomada de decisões sendo elas:

- Condição: Estado do componente desativado.
    - Decisão: Não é efetuado o monitoramento do componente e Avisa o Usuário sobre alteração.
- Condição: Intervalo de tempo de espera para atualização expirou.
    - Decisão: Atualiza o dado atual monitorado dos componentes.
- Condição: Interruptor foi acionado.
    - Decisão: Alerta o Usuário.

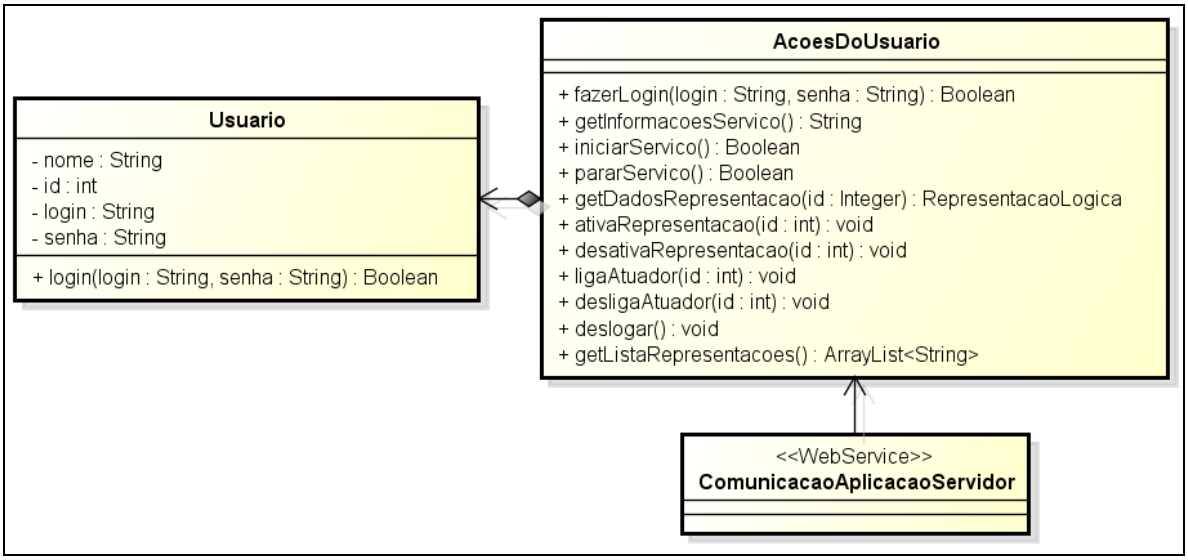
Figura 33 - Diagrama de classes proposto para o módulo de Serviço



Fonte: do Autor.

Usuário: O módulo de usuários na arquitetura proposta faz o papel de controlar o acesso das ações que podem ser executadas no sistema. Vide o diagrama de classes na Figura 34. O protótipo terá somente um usuário, o dono, ao qual interagirá com o sistema mediante autenticação.

Figura 34 - Diagrama de classes dos módulos de Usuários e de Comunicação Aplicação Servidor



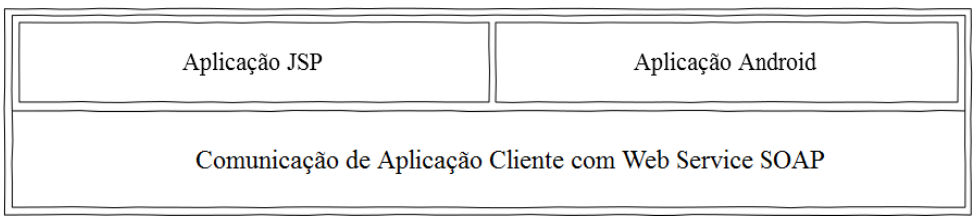
Fonte: do Autor.

Comunicação Aplicação Servidor: Este módulo tem o papel de permitir que as ações do usuário possam ser executadas na rede global. No diagrama de classes da Figura 35 é utilizada a forma de comunicação Web Service SOAP, implementada com Java, se fosse necessário poderia ser colocada outra forma de comunicação, como socket, por exemplo, ao qual executará as mesmas ações do usuário da mesma forma que as demais formas de comunicação.

4.3.3 Camada Cliente

A camada cliente tem a finalidade de possibilitar que diversos tipos de clientes consigam acessar o servidor por plataformas distintas. No projeto proposto fora implementado dois cliente: O primeiro cliente com interfaces em JSP e comunicação com Web Service SOAP utilizando a biblioteca jax-ws; O segundo na plataforma móvel Android e utilizando comunicação com Web Service SOAP pela biblioteca Ksoap2. A Figura 35 demonstra a estrutura proposta abordada por essa camada.

Figura 35 - Estrutura proposta para a Camada Cliente



Fonte: do Autor.

Comunicação Aplicação Cliente: Este módulo tem o papel de usar as ações do usuário presentes na comunicação aplicação servidor, possibilitando assim que a aplicação consiga interagir com o servidor.

Aplicação: Este módulo oferece ao usuário a possibilidade de interagir graficamente com o servidor. As ações que o usuário pode executar estão expressas na classe AcoesDoUsuario na Figura 34.

4.3.4 Considerações sobre o desenvolvimento

Até chegar ao resultado atual diversos aspectos da arquitetura e implementação foram aperfeiçoados. Dentre eles pode-se destacar para camada de nível baixo:

- O uso de pré-compiladores: O uso de pré-compiladores (*#define*) dentro do microcontrolador foi crucial, pois estes ocupam espaços menores que variáveis, assim, para otimizar o espaço, todas as variáveis que podem ser pré-compiladas, por não alterarem seu valor em tempo de execução, foram substituídas, é o caso da biblioteca que contém o protocolo de interação;
- Uso de tipos dados char* e char[] no lugar de String: A diferença de espaço ocupado entre esses tipos de dado é grande, o objeto String ocupa muito mais espaço de memória para ser utilizado, isso porque ele possui muitas funções que podem ser utilizadas sobre os valores, em outras palavras, comparando o espaço utilizado por um char[] com uma String notou-se uma diferença de 1490 byes entre eles. O resultado desse teste para o projeto é que não há necessidade para o caso em questão de utilizar um tipo de dado String, pois não há um aproveitamento de todos os recursos que dispõe;
- Uso de classes ao invés de estruturas: Uma dúvida que se tinha durante o início do projeto foi: como ter uma arquitetura flexível, que permita ser adaptada o mais facilmente possível para outros casos. A solução foi adotar uma estrutura ou uma classe, mas somente uma dessas possibilidades era necessária. Assim, foram executados testes entre eles, chegando ao resultado que as duas ocupam espaços equivalentes, isso levando em conta que o objeto possui apenas métodos de GET e SET (que somente acessam as variáveis privadas). Mas pelo fato da classe comportar o uso de métodos para tratar os dados que são inseridos, especialmente pelo caso da manipulação dos tipos de dado char* e char[], foi escolhida a classe;

- Caracteres tratados pela tabela ANSII (*American Standard Code for Information Interchange*): Todos os caracteres no microcontrolador Atmega328 obedecem à tabela ANSII, desta forma, um caractere pode ser representado pelo seu respectivo valor nos formatos: inteiro, hexadecimal, binário e Glifo. Fora muito importante descobrir essa informação, pois muitas vezes havia como saída um valor em Glifo e ao exibir ele através da comunicação o resultado era sempre um inteiro, em outras palavras, o valor inteiro desse Glifo na tabela;
- Criação do protocolo de interação: Só foi possível definir o protocolo de interação após descobrir que o Atmega328 utiliza a tabela ANSII, assim, todos os comandos do protocolo para integrar a camada física com a camada intermediária foram baseados em caracteres.

Dentre os resultados da camada intermediária pode-se destacar:

- Uso de padrões de projeto OO (Orientado a Objeto): O uso de padrões de projeto da OO possibilitou tratar os problemas de modelagem mais facilmente, tais como: possibilitar a separação de comportamentos em interfaces (Padrão *Strategy*), o que possibilita trocar em tempo de execução o comportamento, possibilita usar uma fachada para camuflar partes complexas que são executadas por funções simples (Padrão *Facade*), possibilitar a criação de fábrica de objetos que me retornará um objeto concreto que respeita a interface que a variável se refere (Factory), fornecendo escalabilidade ao código, e a possibilidade de tratar situações onde alguns objetos dependem de dados provenientes de outro objeto e quando algo muda no objeto observado é preciso que todos os seus dependentes sejam avisados (Padrão *Observer*).

- Uso de temporizador para atualização de dados pelo serviço: Nos testes da camada intermediária ocorreram erros ao atualizar os dados das representações lógicas pelo hardware, acontece que todos os dispositivos controlados estão conectados por uma comunicação apenas, e esta por sua vez consegue suportar um volume de dados de 10Mb por segundo, enquanto a placa do servidor consegue suportar 100Mb por segundo. Além disso, o processamento monocore da camada física é de 16 MHz enquanto o Servidor possui um processador dualcore de 2.27 GHz. O resultado é que a camada física não conseguia dar conta das requisições da camada

intermediária, a solução foi adotar um temporizador na atualização dos dados por parte do serviço da camada intermediária.

4.3.5 Dificuldades enfrentadas

O foco do desenvolvimento do protótipo consistiu na formulação de modelos para integrar os diversos hardwares que podem comunicar-se com o servidor através de uma LAN. Durante esse desenvolvimento as principais dificuldades enfrentadas foram relacionadas ao hardware, uma vez que programar com uma linguagem de baixo nível é bastante diferente da programação convencional em desktop. O processo foi lento, principalmente em relação ao Arduino. Não era esperado que fosse necessário dedicar tanto tempo, uma vez que, conforme o ARDUINO TEAM (2012), qualquer pessoa interessada, sem muito conhecimento consegue criar objetos ou ambientes interativos usando essa plataforma de protótipos flexível, com hardwares e softwares fáceis de usar. De fato é possível, contudo o resultado procurado utilizava recursos que ainda não são tão divulgados para serem utilizados nesses ambientes, tais como: structs, classes, map e transporte de grandes volumes de informação, além de boas práticas para economizar memória e processamento no ambiente limitadíssimo que dispõem os microcontroladores atmega328, que de acordo com o *datasheet* divulgado pela ATMEL (2009), dispõem somente de 32 KB (2KB usados para o bootloader) de memória flash e velocidade de clock de 16 MHz, muito menor se comparado à capacidade dos PCs atuais.

Outro grande desafio foi sua comunicação, quando fora comprada a Shield Ethernet, por lógica e inexperiência, adquiriu-se a placa mais barata encontrada, que infelizmente funcionava com uma biblioteca de baixo nível que não era a padrão do Arduino. Nesta biblioteca somente tinha sido implementada a parte de recebimento de informação e exibição em uma tela, não podendo conectar-se automaticamente caso precisasse atualizar o Servidor através do Cliente. Além desse aspecto, essa biblioteca não dispõe de um Buffer para a comunicação no nó de controle, o que limita para somente um único pacote, com tamanho limitado, a possibilidade de ser enviado ao servidor, o que ocasionou o erro de envio de diversos pacotes que levavam grandes informações em XML, obrigando a mudar a estratégia de envio para pacotes menores, o resultado disso é a adoção de versões de modelos XML, onde é possível utilizar várias táticas de envio e o servidor tratará devidamente cada uma.
Além disso, outra dificuldade importante enfrentada foi à incompatibilidade de utilizar o controle físico com a comunicação local cliente. Separadas as camadas funcionavam, contudo quando juntadas, somente o controle funcionava devidamente. Incialmente se pensava que o defeito causado pelo fato do microcontrolador ser monoprocessado, exigindo que tivesse que

usar duas placas Arduino, mas ao testar com as duas placas, o defeito persistiu. As pesquisas continuaram em busca de uma solução, e finalmente fora encontrada, de acordo com a equipe WEBTRONICO (2012) a alimentação ideal para o bom funcionamento da placa deve ser entre 7 a 12 volts, e atualmente estava usando uma alimentação por USB de 5 volts, após comprar uma fonte de alimentação 12 volts a placa funcionou corretamente.

O projeto da camada intermediária fora por diversas vezes postergado e refeito, pois algumas mudanças no hardware alteram diretamente o funcionamento do middleware, que até conseguir a estabilidade do projeto atual teve grandes mudanças, tais como adoção do protocolo de interação para poder reduzir o tamanho dos pacotes transportados.

CONSIDERAÇÕES FINAIS

Os sistemas distribuídos pervasivos e ubíquos residenciais ainda têm diversos desafios a enfrentar antes que se tornem realidade, mas já estamos rumo a realiza-la. Este trabalho buscou contribuir para o desenvolvimento de sistemas ubíquos para ambiente residencial, propondo uma arquitetura conceitual com modelos de interação para comunicação entre os diversos tipos de hardwares e com uma forma de comunicação que o torne ubíquo.

O objetivo principal fora alcançado, projetar a arquitetura, denominada UBGEAR, sendo muitas vezes revisada e alterada durante o processo. Os resultados das consultas a literatura e o desenvolvimento do protótipo permitiram chegar ao estado satisfatório atual. Em resumo a arquitetura procura propor um modelo de camadas divididas em física, intermediária e cliente. A camada física faz o controle dos componentes físicos e conecta-os a camada intermediária, que por sua vez faz o devido tratamento dessas informações para serem utilizadas por serviços que gerenciam um contexto maior ao qual inclui tanto as funcionalidades físicas como os recursos avançados disponíveis, além disso, essas informações podem ser acessadas pelo proprietário desses componentes através de interfaces ligadas na rede global, conseguindo a partir da camada cliente interagir com o sistema controlando-o e monitorando-o.

Por fim, a abordagem geral deste trabalho permite a abertura de diversos tópicos para serem abordados por trabalhos futuros, tais como:

- Fazer o reconhecimento de novos nós de controle adicionando-os em uma lista de espera ou um serviço padrão;
- Adicionar a funcionalidade de contexto na implementação;
- Avaliar as melhores formas de comunicação com o hardware;
- Comparar microcontroladores para ver desempenho no sistema;
- Desenvolver uma interface gráfica intuitiva para o gerenciamento das informações;
- Estudo o uso de streaming com o hardware, com testes e aperfeiçoamento da arquitetura;
- Verificar a possibilidade de escalonamento da comunicação colocando os envelopes por prioridade;
- Fazer um estudo sobre formas de tratamento de erro de comunicação entre elementos distribuídos.

REFERÊNCIAS

ARAUJO, Regina Borges. *Computação Ubíqua, Princípios, Tecnologias e Desafios*. XXI Simpósio Brasileiro de Redes de Computadores. 2003. Disponível em: <http://im.ufba.br/pub/MAT570FG/LivroseArtigos/045_AraujoRB.pdf>. Acesso em 06 nov. 2011.

ARAÚJO, Gustavo M., SIQUEIRA, Frank. *Integração de Dispositivos Móveis em Ambientes Ubíquos usando o Device Service Bus*. 2009.

ARDUINO TEAM. *Site oficial do Arduino*. Disponível em: <http://arduino.cc>. Acesso em: 16 jun. 2012.

ATMEL. *Site oficial do fabricante Atmel Corporation*. Disponível em: < http://www.atmel.com/ >. Acesso em: 11 jul. 2012.

AUGUSTIN, Iara. *Abstrações para uma Linguagem de Programação Visando Aplicações Móveis em um Ambiente de Pervasive Computing*. 2004. Disponível em: <http://www.lume.ufrgs.br/handle/10183/3866>. Acesso em: 03 nov. 2011.

BASSO, Carla de A. M. *Aplicação de XML para Estrutura de Ambientes de Controle Acadêmico baseado em Ontologias*. Porto Alegre: UFRGS, 2002.

BOLZANI, Caio A. Morais. *Análise de Arquiteturas e Desenvolvimento de uma Plataforma para Residências Inteligentes*. 2010. Disponível em: < http://www.teses.usp.br/teses/disponiveis/3/3142/tde-12082010-112005/en.php>. Acesso em 24 nov. 2011.

COULOURIS G., DOLLIMORE J., KINDBERG T. *Sistemas Distribuídos: Conceitos e Projetos*. 4. ed. Porto Alegre: Bookman, 2007. Disponível em: <http://books.google.com.br/books?id=KSZ1rIRWmUoC&printsec=frontcover&dq=sistemas+distribuidos&hl=pt-BR&ei=xqfWTtrqJ4HZ0QGG6ZjtAQ&sa=X&oi=book_result&ct=result&resnum=1&ved=0CDMQ6AEwAA#v=onepage&q&f=false>. Acesso em: 30 nov. 2011.

FIGUEIREDO, Carlos M. Seródio. *Introdução a computação Móvel*. Disponível em: <http://www.ufpi.br/subsiteFiles/ercemapi/arquivos/files/palestras/palestra3.pdf>. Acesso em: 01 nov. 2011.

KREGER, Heather. *Web Services Conceptual Architecture* (WSCA 1.0). 2001. Disponível em:<http://www.csd.uoc.gr/~hy565/newpage/docs/pdfs/papers/wsca.pdf>. Acesso em: 30 nov. 2011.

MARCIEIRA, Maria E. Bastos, *O Processo Nosso de Cada Dia*. Rio de Janeiro: Qualkitymark Editora, 2006. Disponível em: < http://www.mtprocessos.com.br/IndicadoresdeDesempenho.pdf>. Acesso em 29 nov. 2011.

MARGOLIS, Michael. *Arduino Cookbook*: Recipes to Begin, Expand and Enhance Your Projects. 2. Ed. Sebastopol: O'Reilly Media, 2011. Disponível em: <http://books.google.com.br/books?id=56H-

13NzsSQC&printsec=frontcover&dq=Arduino&hl=pt-BR&sa=X&ei=YIUPT8qULIzsggeJiLD7Aw&ved=0CFsQ6AEwBQ#v=onepage&q=Arduino&f=true>. Acesso em: 12 jan. 2012.

MCROBERTS, Michael. *Arduino Básico*. 1. Ed. São Paulo: Novatec Editora, 2011. Disponível em: <http://www.novatec.com.br/livros/arduino/capitulo9788575222744.pdf>. Acesso em: 14 jan. 2012.

MICROCHIP. *Site oficial do fabricante Microchip Technology Inc.*. Disponível em: < http://www.microchip.com/ >. Acesso em: 11 jul. 2012.

TANEMBAUM, Andrew S., STEEN, Maarten Van. *Sistemas distribuídos:* princípios e paradigmas. 2. Ed. São Paulo: Pearson Education Brasil, 2007.

TANEMBAUM, Andrew S. *Organização estruturada de computadores*. 5. Ed. São Paulo: Pearson Prentice Hall, 2007.

TEXAS. *Site oficial do fabricante Texas Instruments semiconductor innovations*. Disponível em: < http://www.ti.com/ >. Acesso em: 11 jul. 2012.

VENTURI, Eli. *Protótipo de um Sistema para Controle e Monitoração Residencial Através de Dispositivos Móveis Utilizando a Plataforma .NET*. 2005. Disponível em: <http://www.inf.furb.br/~pericas/orientacoes/CtrlResidencial2005.pdf >. Acesso em: 14/11/2011.

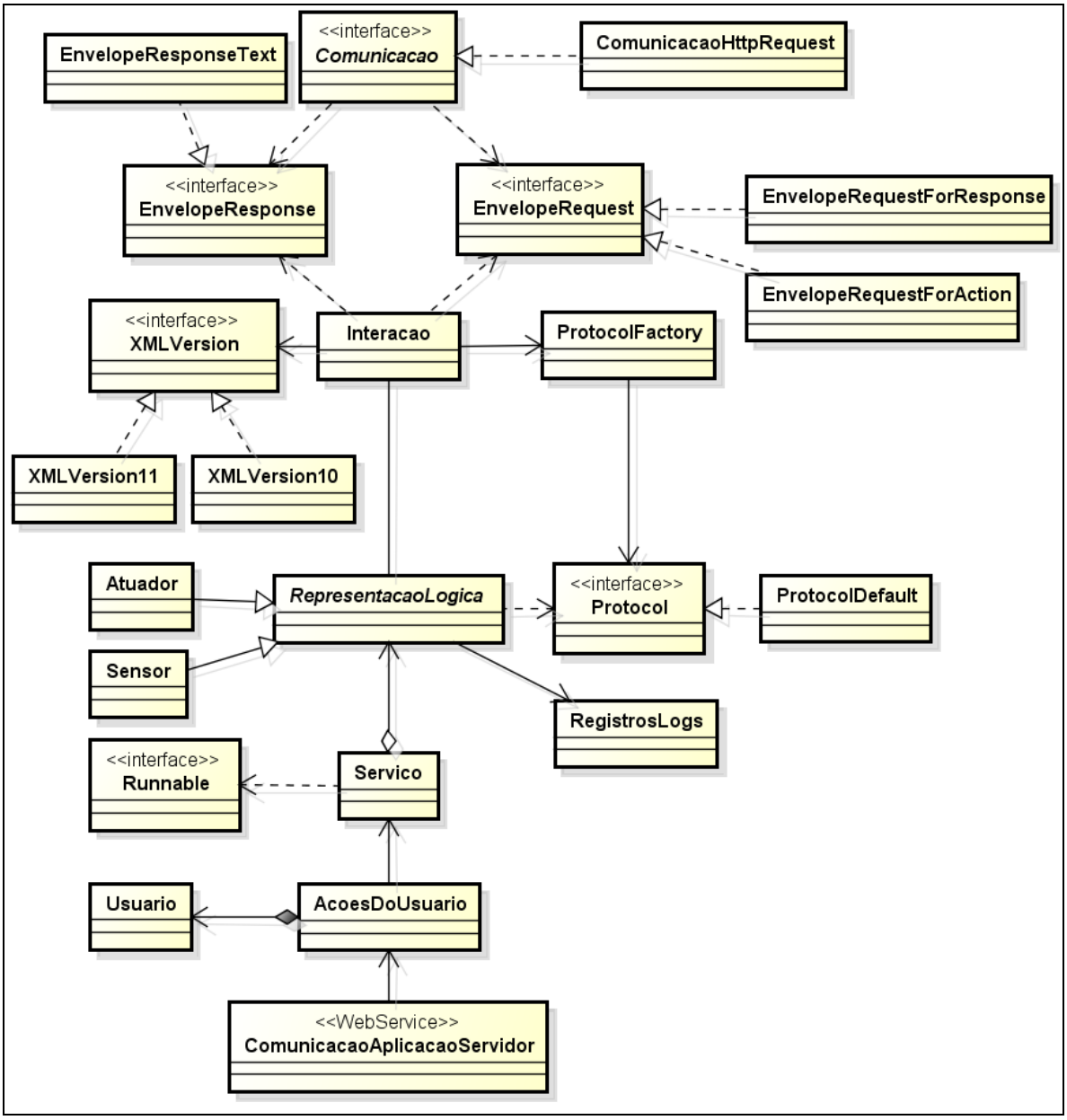
WEBTRONICO, equipe. . Disponível em: < http://www.webtronico.com/arduinos/arduinos/placas-arduino/arduino-duemilanove.html>Acesso em:

WEISER, M., BROWN, J. S. *"The coming age of calm technology".* 1996. Disponível em: <http://www.ubiq.com/hypertext/weiser/acmfuture2endnote.htm>. Acesso em: 27 nov. 2011.

ANEXOS E APÊNDICES

APÊNDICE A – Diagrama de Classes Completo da Camada Intermediária

Figura 36 - Diagrama de classes Completo da Camada Intermediária



Fonte: do Autor.

APÊNDICE B – XML Resultado das ações de retorno dos componentes da camada física

Figura 37 – XML contendo as configurações do nó de controle, pelo comando /c

```
<?xml version="1.0" encoding="UTF-8"?>
<root version="1.1">
    <config>
        <model>1.0</model>
        <serie>TCC</serie>
        <mac>84-85-88-16-0-36</mac>
        <ip>10.5.99.191</ip>
        <protocol>default</protocol>
        <nodes>1</nodes>
        <devices>3</devices>
    </config>
</root>
```

Fonte: do Autor.

Figura 38 – XML do Mapa de componentes presentes no nó de controle, pelo comando /m

```
<?xml version="1.0" encoding="UTF-8"?>
<root version="1.1">
    <map>
        <node>1</node>
        <id>1</id>
    </map>
    <map>
        <node>1</node>
        <id>2</id>
    </map>
    <map>
        <node>1</node>
        <id>3</id>
    </map>
</root>
```

Fonte: do Autor.

Figura 39 – XML resultado da coleta completa de dados do Componente 1, pelo comando /11ga

```
<?xml version="1.0" encoding="UTF-8"?>
<root version="1.0">
    <config>
        <ip>10.5.99.191</ip>
    </config>
    <metadata>
        <node>1</node>
        <activity>115</activity>
        <id>1</id>
        <status>116</status>
        <device>10</device>
        <datatype>98</datatype>
        <data>t</data>
        <about>Reed Switch</about>
        <min>0</min>
        <max>0</max>
    </metadata>
</root>
```

Fonte: do Autor.

Figura 40 - XML resultado da coleta simplificada de dados do Componente 1, pelo comando /11gs

```
<?xml version="1.0" encoding="UTF-8"?>
<root version="1.0">
    <config>
        <ip>10.5.99.191</ip>
    </config>
    <metadata>
        <node>1</node>
        <activity>115</activity>
        <id>1</id>
        <status>116</status>
        <datatype>98</datatype>
        <data>t</data>
    </metadata>
</root>
```

Fonte: do Autor.

Figura 41 - XML resultado da coleta completa de dados do Componente 2, pelo comando /12ga

```
<?xml version="1.0" encoding="UTF-8"?>
<root version="1.0">
    <config>
        <ip>10.5.99.191</ip>
    </config>
    <metadata>
        <node>1</node>
        <activity>97</activity>
        <id>2</id>
        <status>102</status>
        <device>1</device>
        <datatype>98</datatype>
        <data>t</data>
        <about>LED do Reed Switch</about>
        <min>0</min>
        <max>0</max>
    </metadata>
</root>
```

Fonte: do Autor.

Figura 42 - XML resultado da coleta simplificada de dados do Componente 2, pelo comando /12gs

```
<?xml version="1.0" encoding="UTF-8"?>
<root version="1.0">
    <config>
        <ip>10.5.99.191</ip>
    </config>
    <metadata>
        <node>1</node>
        <activity>97</activity>
        <id>2</id>
        <status>102</status>
        <datatype>98</datatype>
        <data>t</data>
    </metadata>
</root>
```

Fonte: do Autor.

Figura 43 - XML resultado da coleta completa de dados do Componente 3, pelo comando /13ga

```
<?xml version="1.0" encoding="UTF-8"?>
<root version="1.0">
    <config>
        <ip>10.5.99.191</ip>
    </config>
    <metadata>
        <node>1</node>
        <activity>97</activity>
        <id>3</id>
        <status>116</status>
        <device>1</device>
        <datatype>98</datatype>
        <data>t</data>
        <about>LED Pisca Pisca</about>
        <min>0</min>
        <max>0</max>
    </metadata>
</root>
```

Fonte: do Autor.

Figura 44 - XML resultado da coleta simplificada de dados do Componente 3, pelo comando /13gs

```
<?xml version="1.0" encoding="UTF-8"?>
<root version="1.0">
    <config>
        <ip>10.5.99.191</ip>
    </config>
    <metadata>
        <node>1</node>
        <activity>97</activity>
        <id>3</id>
        <status>102</status>
        <datatype>98</datatype>
        <data>t</data>
    </metadata>
</root>
```

Fonte: do Autor.