UNIVERSIDADE DE BRASÍLIA
FACULDADE DE TECNOLOGIA
DEPARTAMENTO DE ENGENHARIA MECÂNICA

DESENVOLVIMENTO DE UM *GATEWAY* BACNET/HTTP E DE UM AMBIENTE DE PROJETO PARA SISTEMAS DOMÓTICOS INTEGRADOS COM TV DIGITAL

MAGNO BATISTA CORRÊA

ORIENTADOR: CARLOS HUMBERTO LLANOS QUINTERO

DISSERTAÇÃO DE MESTRADO EM SISTEMAS MECATRÔNICOS

PUBLICAÇÃO: ENM.DM- 043A/11

BRASÍLIA/DF: SETEMBRO - 2011.

UNIVERSIDADE DE BRASÍLIA
FACULDADE DE TECNOLOGIA
DEPARTAMENTO DE ENGENHARIA MECÂNICA

DESENVOLVIMENTO DE UM *GATEWAY* BACNET/HTTP E DE UM AMBIENTE DE PROJETO PARA SISTEMAS DOMÓTICOS INTEGRADOS COM TV DIGITAL

MAGNO BATISTA CORRÊA

DISSERTAÇÃO DE MESTRADO SUBMETIDA AO DEPARTAMENTO DE ENGENHARIA MECÂNICA DA FACULDADE DE TECNOLOGIA DA UNIVERSIDADE DE BRASÍLIA, COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM SISTEMAS MECATRÔNICOS.

APROVADA POR:

Prof. Carlos Humberto Llanos Quintero, Dr. (Departamento de Engenharia Mecânica - UnB) - (Orientador)

Prof. Ricardo Pezzuol Jacobi, Dr. (Departamento de Ciências da Computação - UnB) - (Examinador Interno)

Prof. Paulo Roberto de Lira Gondim, Dr. (Departamento de Engenharia Elétrica - UnB) - (Examinador Externo)

BRASÍLIA/DF, 30 DE SETEMBRO DE 2011.

FICHA CATALOGRÁFICA

CORREA, MAGNO BATISTA

Desenvolvimento de um *Gateway* BACnet/HTTP e de um Ambiente de Projeto para Sistemas Domóticos Integrados com TV-Digital

[Distrito Federal] 2011.

xvii, 135p., 297 mm (ENM/FT/UnB, Mestre, Sistemas Mecatrônicos , 2011). Dissertação de Mestrado - Universidade de Brasília. Faculdade de Tecnologia.

Departamento de Engenharia Mecânica .

1. Automação Domótica
2. BACnet
3. Gateway
4. Set-top box

I. ENM/FT/UnB
II. Título Mestre

REFERÊNCIA BIBLIOGRÁFICA

CORREA, MAGNO BATISTA (2011). Desenvolvimento de um *Gateway* BACNet/HTTP e de um Ambiente de Projeto para Sistemas Domóticos Integrados com TV Digital. Dissertação de Mestradoem Sistemas Mecatrônicos, Publicação ENM.DM- 043A/11, Departamento de Engenharia Mecânica , Universidade de Brasília, Brasília, DF, 135p.

CESSÃO DE DIREITOS

NOME DO AUTOR: Magno Batista Corrêa.

TÍTULO DA DISSERTAÇÃO DE MESTRADO: Desenvolvimento de um *Gateway* BACNet/HTTP e de um Ambiente de Projeto para Sistemas Domóticos Integrados com TV Digital.

GRAU / ANO: Mestre / 2011

Magno Batista Corrêa
Quadra 154 Lote 28
72.871-074 Céu Azul - Valparaiso - GO - Brasil.

DEDICATÓRIA

Dedico esta dissertação primeiramente para aqueles que buscam o conhecimento pelos caminhos tortuosos da vida, buscando soluções simples para problemas que não necessariamente existem. A você, que está em busca do bem mais valioso do mundo, esta obra é inteiramente dedicada.

A aquelas que sempre me amam
e a aqueles que nunca me amaram.

AGRADECIMENTOS

Agradeço, não para seguir uma forma, nem por falta de originalidade, mas sim com verdade em palavras, a Deus que me concedeu grandes oportunidades, me guiando pelos caminhos dos estudos e do trabalho duro.

Agradeço também a meus pais, que segundo as suas próprias palavras, me deram o único bem que uma família pobre pode dar a seus filhos, a educação. Agradeço a minha mãe, Maria Salvina Batista Glória, que me ensinou as lições mesmo sem saber, me ensinando a buscar compreender além do que é explicado, através da observação e do amor. Agradeço a meu finado pai, Mailton Silva Corrêa, por todo o trabalho, domingo a domingo, que me deram o suporte para estudar sem ter que começar a trabalhar ainda criança, para o sustento da família.

Agradeço a minha esposa, Maria de Jesus Sousa Corrêa e minha filha Maria Eduarda Sousa Corrêa, pelo conforto e compreensão nas muitas noites de trabalho.

Agradeço ao meu orientador professor Doutor Carlos Humberto Llanos Quintero que, acima de tudo, acreditou em mim, mesmo quando eu não acreditava.

Agradeço a Universidade de Brasília e a República Federativa do Brasil, pelo investimento na minha formação.

Agradeço aos amigos que me deram suporte, em especial o professor Mestre Jones Yudi Mori Alves da Silva, ao mestrando Bernardo Vergne Dias, professor Mestre Evandro Leonardo e ao Doutorando Daniel Muñoz, pela revisão do texto e pelos comentários valiosos e pelo apoio nas horas de desespero. A todos os outros colegas que deram o suporte com a sua presença, meu muito obrigado.

Agradeço por fim, a tudo que não é meu, que me propicia a consciência que não poderei encher um pote com a água de um oceano.

RESUMO

DESENVOLVIMENTO DE UM *GATEWAY* BACNET/HTTP E DE UM AMBIENTE DE PROJETO PARA SISTEMAS DOMÓTICOS INTEGRADOS COM TV DIGITAL

Autor: Magno Batista Corrêa
Orientador: Carlos Humberto Llanos Quintero
Programa de Pós-graduação em Sistemas Mecatrônicos
Brasília, setembro de 2011

O desenvolvimento de aplicativos de automação domótica e o uso dos receptores de TV digital são dois tópicos com grande interesse social. Este interesse é motivado pelo aumento do poder aquisitivo da população e pelos incentivos fiscais dos governos, a fim de que sejam adotados os padrões de TV digital. Neste trabalho estes conceitos são abordados em profundidade, onde foi realizada uma revisão bibliográfica voltada para os conceitos de percepção da automação e aspectos de TV digital, além de um desenvolvimento voltado para as tecnologias atuais de mercado. Neste trabalho são apresentados o projeto, desenvolvimento e testes de um *gateway* BACnet/HTTP, voltado para a execução em dispositivos embarcados. O desenvolvimento e os testes foram realizados em diversas plataformas, sendo elas: (a) a arquitetura PC, (b) um SBC (*Single Board Computer*) com o processador central um ARM e (c) um STB (*Set-Top Box*) da arquitetura SH4. Também é apresentado o projeto, desenvolvimento e testes de um navegador orientado a imagens, que pode ser executado sobre qualquer dispositivo com *framebuffer*. Este navegador serve de interface gráfica para o gateway desenvolvido. Também foi desenvolvido um dispositivo USB BACnet baseado em um microcontrolador AVR, servindo de protótipo para o desenvolvimento de dispositivos BACnet dos mais diversos tipos, sendo apresentado um dispositivo BACnet voltado para o acionamento de lâmpadas e cargas gerais com o controle do tipo liga/desliga.

ABSTRACT

DEVELOPMENT OF A BACNET/HTTP GATEWAY AND A DESIGN ENVIRONMENT FOR HOME AUTOMATION SYSTEMS INTEGRATED WITH DIGITAL TV

Author: Magno Batista Corrêa
Supervisor: Carlos Humberto Llanos Quintero
Programa de Pós-graduação em Sistemas Mecatrônicos
Brasilia, September of 2011

The development of home automation's applications and the use of digital TV receivers are two threads with a great social interest. This interest is motivated by the increasing population's earning power and tax incentives by the government, so that the standards for digital TV should be adopted. In this paper, these concepts are deeply covered, where a review of papers and main concepts has been done, and a development geared to the current market technologies. This paper presents the design, development and testing of a BACnet gateway/HTTP, facing execution in embedded devices. The development and testing were performed on different platforms, namely: (a) the PC architecture, (b) an SBC (Single Board Computer) with an ARM CPU and (c) an STB (Set-Top Box) in SH4 architecture. It also presents the design, development and testing of a images-oriented browser, that can be run on any device with framebuffer. This browser acts as a graphical interface to the developed gateway. Also a BACnet's USB device was developed based on AVR microcontroller, providing a prototype for the development of BACnet devices of all kinds, being presented a BACnet device for the automatic control of lighting and general load, controlled by an on/off switch.

SUMÁRIO

LISTA DE FIGURAS

Lista de Algoritmos e Códigos

LISTA DE SÍMBOLOS, NOMENCLATURA E ABREVIAÇÕES

ADC: *Analog-to-Digital Converter*. Conversor Digital/Analógico.

ALU: *Arithmetic Logic Unit*. Unidade Lógica Aritmética.

Arithmetic Logic Unit (ALU): Parte da CPU que executa operações aritméticas e lógicas

Analog-to-Digital converter (ADC): Dispositivo, geralmente um CI que converte uma grandeza analógica expressa em tensão para o seu valor binário correspondente.

bit: A menor unidade de uma informação digital, que pode tomar o valor 0 ou 1.

byte: Uma palavra digital de 8-bits.

central processing unit (CPU): A parte central de um computador. A CPU realiza todos os cálculos e e realiza os controles de funções do computador.

crossover cable: Vide null modem.

CPU: Vide central processing unit.

DAC: Vide digital-to-analog converter.

digital-to-analog converter (DAC): Circuito que traduz sinais digitais para valores analógicos, geralmente de tensão.

download: Processo de transferência de um arquivo ou programa de um computador para outro, onde o destino é o computador atualmente utilizado.

embedded controller: Um controlador baseado em microcontrolador que está permanentemente instalado na máquina a que se destina a controlar.

gateway: Sistema responsável pela conversão de um protocolo em outro.

GCC: vide GNU Compiler Collection.

GNU Compiler Collection: Conjunto de ferramentas voltadas para a compilação de programas em diversas linguagens, principalmente linguagem C.

input/output (I/O): Fluxo de informação para dentro ou para fora da unidade compu-

tadorizada

I/O Vide input/output.

interface serial: Tipo de interface onde a informação é transmitida 1 bit a cada vez, num barramento único.

instruction set: Grupo de comandos de programas que um microprocessador em particular é desenhado ara reconhecer e executar.

LAN: Vide local area network.

local area network (LAN): Um sistema que torna possível que múltiplas unidades computacionais se comuniquem umas com as outras, através do mesmo meio de comunicação.

memória: Parte do computador que guarda uma informação digital. A informação da memória é armazenada em bytes, onde cada byte possui um endereço.

memory-mapped input/output: Sistema onde as portas de I/O são tratadas tal como endereços de memória.

microcontrolador: Circuito integrado que incluí microprocessador, memória e I/O.

microprocessador: Circuito integrado digital que executa operações básicas de um computador, mas que requer algum suporte de outros CIs para ser funcional.

null modem: Cabo que torna possível duas unidades se comunicar umas com as outras, geralmente por um protocolo serial. PC Vide personal computer.

personal computer (PC): Um computador de uso geral, baseado em microprocessador, auto-contido. Geralmente se refere aos computadores compatíveis com o padrão IBM.

PLC: Vide programmable logic controller.

porta: Parte do computador onde as linhas de dados de I/O são conectadas. Cada porta tem seu próprio endereço.

programmable logic controller (PLC): Controlador robusto, auto-contido e baseado em microcontrolador desenhado especificamente para ser utilizado em um ambiente industrial.

RS-232: Padrão de transmissão serial de dados, em que serão especificados os níveis de tensão e o modelo de empacotamento de dados.

single-board computer (SBC): Um computador baseado em microporcessador montado sobre um único cartão PCI.

Set-Top Box: Receptor de TV digital.

STB: vide Set-Top Box

UART: Vide universal asynchronous receiver transmitter.

universal asynchronous receiver transmitter (UART): Componente de hardware para empacotamento de dados. Geralmente é realizado por um CI específico que converte uma informação paralela em serial, e vice-versa.

word: Unidade digital de informação que um dado microprocessador utiliza. Geralmente as palavras (words) são de 4,8, 16, 32 e 64 bits.

1 INTRODUÇÃO

A televisão é um eletrodoméstico presente em quase todas as casas. No Brasil está ocorrendo uma grande transformação no modo de transmissão do sinal de TV aberta, passando do sinal analógico para o digital, tendo como o ano de 2016 o marco final das transmissões analógicas. Devido esta mudança no modo de transmissão unido aos diversos incentivos do governo federal está ocorrendo a popularização da TV digital no mercado brasileiro.

A utilização da TV digital faz necessário o uso de *receptores de TV digital*, conhecidos em inglês como *set-top box* ou simplesmente STB. Estes STBs possuem um grande poder de processamento (HEATH, 2007; STMICROELECTRONICS, 2009; STLINUX, 2009; GINGA, 2010), geralmente com *hardwares* com mais de 400MHz (HEATH, 2007). Tal *hardware* fica em boa parte do tempo ocioso, visto que as funções de descompactação de vídeo estão em decodificadores dedicados. Um uso interessante para este *hardware* no seu tempo "ocioso" seria a automação da residência do proprietário.

Contudo, o desenvolvimento de sistemas de automação domótica apresenta diversos requerimentos, que vão desde a definição da qualidade técnica do produto gerado, até uma crescente necessidade de qualidade estética e de acabamento. Também são fatores importantes para o bom atendimento ao usuário do sistema, a observância dos quesitos de usabilidade e praticidade do produto, além de um preço acessível à maior parte da população. Para que seja disponibilizada uma solução de automação domótica, vários aspectos técnicos devem ser definidos como por exemplo, a solução de interface gráfica, o protocolo de comunicação adotado e a arquitetura de sistema.

A fase de definição do protocolo de comunicação geralmente requer um tempo considerável, além de ser um dos fatores determinantes na definição da solução, devido às limitações que a escolha do protocolo pode infligir na definição da solução, tanto a nível dos elementos de automação que poderão ser adquiridos ou confeccionados, quanto da arquitetura que poderá ser adotada.

Para ilustrar o fator limitante da escolha de um protocolo, caso se opte pela escolha de

um protocolo comercial como (BTICINO, 2010) e (METASYS, 2010), tanto os elementos de automação quanto a arquitetura de solução estarão limitados às possibilidades disponibilizadas pelo fornecedor, geralmente com pouca possibilidade de customização. Por outro lado, caso se opte pelo desenvolvimento de um protocolo, ter-se-á um consumo de tempo e recursos para a elaboração do dado protocolo, além da necessidade do desenvolvimento de todos os elementos de automação. Num outro cenário, ao se escolher um padrão aberto como o BACnet (ASHRAE135, 1995), se tem uma economia no tempo de definição do protocolo, além da possibilidade de comprar alguns dos elementos de automação. Ao se adotar um padrão aberto, pode-se aperfeiçoar a relação tempo/custo, além de poder contar com o suporte de literatura acadêmica e técnica a respeito do protocolo, bem como uma comunidade que trabalha e desenvolve produtos com este dado protocolo, diminuindo o *time to market* dos produtos. Além dos benefícios econômicos ainda pode-se agregar conhecimento para a sociedade com uma escolha de um padrão aberto.

Ao longo da história, a automação residencial e a televisão vêm seguindo caminhos praticamente paralelos, com alguns encontros esporádicos. Estas intersecções históricas geralmente ocorrem quando se tem automações para a TV, como é o caso do controle remoto, que inicialmente era um controle remoto a "fio" (vide figura 1.1) e que atualmente possue transmissão a infravermelho.

Figura 1.1: Controle remoto Fisher a fio. Acervo próprio do autor.

Outro caso de convergência entre a automação residencial e os aparelhos de TV são os circuitos fechados de TV (CFTV), onde ocorreu o desenvolvimento de aparelhos de filmagem de vídeo, transmissão a cabo e recepção em múltiplos canais no mesmo

aparelho de TV a fim de prestar maior segurança à residência ou prédio.

Já a integração da TV com os computadores começou a com a confecção dos primeiros computadores pessoais, como o popular TK2000 (vide figura 1.2), que se utilizava do aparelho de TV como monitor. Os aparelhos de videogame, desde o *Odissey* (vide figura 1.2) da empresa *Magnavox* criado na década de 1970, se utilizavam da televisão para a projeção de suas imagens. Atualmente os aparelhos de TV voltaram a ter uma resolução compatível com os monitores, tornando quase que indistinguível um monitor de um televisor.

Figura 1.2: TK2000 e Odissey. Exemplos de integração entre TV e computadores. Acervo próprio do autor.

Outra integração da TV com os computadores é a inserção do computador dentro da TV, buscando a diminuição do número de componentes discretos e o provimento de serviços de melhor qualidade, como por exemplo o *close caption* e a nomeação de canais com o OSD (*On Screen Display*). O avanço do poder computacional tornou viável a inserção, a um custo aceitável, de processadores com mais de 200MHz dentro de um aparelho de TV (HEATH, 2007). Este poder computacional é semelhante aos computadores da década de 1990 (que eram capazes de acessar a Internet e possuíam sistemas multimídia e jogos diversos).

Assim, a domótica apresenta uma série de funcionalidades como, por exemplo, o provimento de conforto, segurança e saúde para os habitantes da residência automatizada (vide figura 1.3). Já a TV digital busca atender a uma série de requisitos sendo estes, por exemplo, a entrega de serviços de qualidade (QoS, em inglês *Quality of Service*), garantindo a qualidade da informação e o provimento de iteratividade com o usuário. Infelizmente os *mundos* da domótica e da TV Digital estão disjuntos e quando se tenta utilizar estes dois serviços, encontra-se atualmente um cenário quase caótico, com a necessidade de utilização de vários controles remotos, custos elevados para a manutenção

dos dois sistemas e até mesmo um certo grau de desconforto, devido a necessidade de convivência com este cenário.

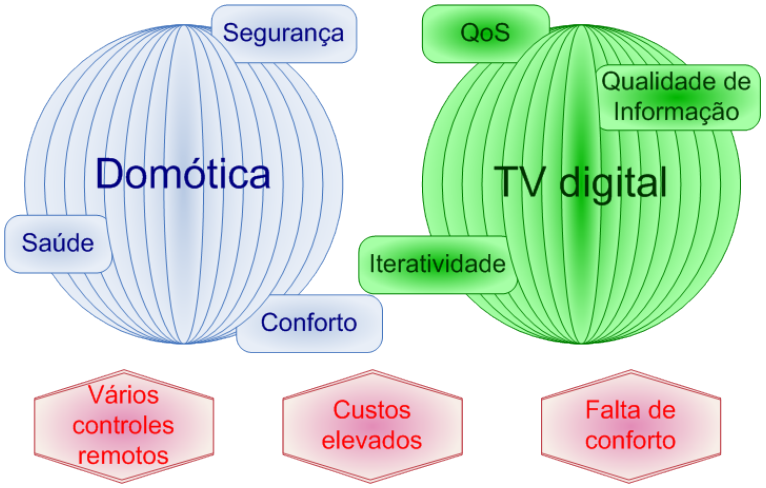

Figura 1.3: Mundos da Domotica e da TV Digital

Esta dissertação busca prover meios para a *fusão* destes dois *mundos*, através do desenvolvimento de um ambiente de projetos para sistemas domóticos integrados com TV digital, além de facilitar o desenvolvimento de soluções de automação com o desenvolvimento de um *tradutor* do protocolo que a casa utiliza para um simples e acessível a todos os interessados em realizar automação residencial.

1.1 ASPECTOS GERAIS

Uma proposta de integração de sistemas de automação residencial com o uso do *gateway* BACnet/HTTP embarcado em um receptor de TV digital é apresentado na figura 1.4. Nesta proposta, o usuário irá enviar comandos através do controle remoto, tendo um retorno visual através do aparelho de televisão. O receptor de TV digital além de receber a programação difundida através do sinal de TV digital da torre de TV, também possui um aplicativo de automação residencial.

Os dispositivos BACnet podem interagir nesta proposta com o receptor de TV digital tanto através de redes IP quanto de redes MS/TP (*Master Slave/ Token Passing*), respondendo aos comandos digitados pelo usuário pelo controle remoto, retornando os estados para o usuário através do aparelho de TV. O usuário pode ter controle de diversos elementos de sua residência como, por exemplo, o controle sobre o acionamento de iluminação, tanto a artificial (lâmpadas) quanto a natural (persianas), climatização

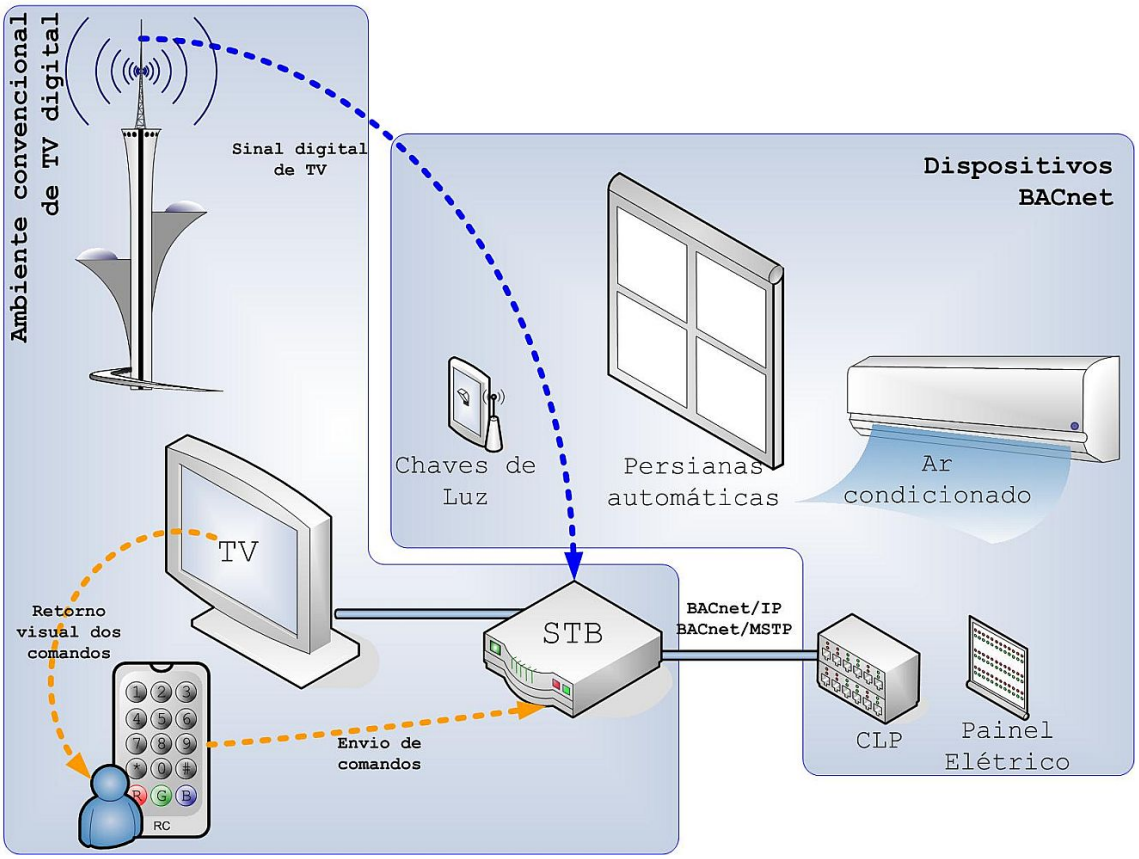

Figura 1.4: Exemplo de uma possível implementação de um sistema integrando automação residencial com o uso de STB e do *gateway* BACnet/HTTP

(ar-condicionado, aquecedor e umidificador), dentre diversos outros. Este acionamento pode ser através de uma rede sem fio ou através de um barramento RS485 até o dispositivo, ou ainda através de um CLP (*Controlador Lógico Programável*) residencial instalado ao lado do painel elétrico. Numa abordagem mais ampla, o sistema pode ainda integrar soluções de segurança, com a instalação de sensores externos, sensores de presença ou mesmo prover a visualização de câmeras de segurança.

1.2 DESCRIÇÃO DO PROBLEMA

Alguns métodos de desenvolvimento de aplicações com alta qualidade de interface gráfica e funcional para automação doméstica vêm sendo apresentados na literatura nos últimos anos (UMBERGER, 2008; ÖSTERLIND et al., 2007; LITZ; GROSS, 2007). Contudo, estas soluções visam o desenvolvimento de um produto único e não uma solução que sirva de apoio para o desenvolvimento de outros produtos, tornando necessária a utilização dos subprodutos gerados nestes trabalhos como uma solução fechada, e não como um suporte para desenvolvimento de uma nova solução. Nesta linha de

raciocínio, esta dissertação não visa o desenvolvimento de um produto fechado, mas sim de uma plataforma que permita o desenvolvimento de produtos, através de meios facilitadores de expansão, além de um método de realizar tal desenvolvimento, apresentado de uma forma eficaz. O desenvolvimento de aplicações para receptores de TV digital baseadas em *software* livre apresenta grandes desafios, quase todos oriundos de falta de documentação de qualidade, além do trabalho inerente à configuração de um ambiente de desenvolvimento integrado. Esta dissertação vem de encontro a estas dificuldades, apresentando uma documentação e um ambiente pré-configurado para a elaboração de soluções de automação domótica, bem como a confecção de produtos de baixa complexidade voltados para os ambientes embarcados de TV digital.

1.3 OBJETIVOS

1.3.1 Objetivos Gerais

O objetivo geral desta dissertação é estudar e propor uma plataforma de projetos para automação residencial, bem como projetar e desenvolver um *gateway* BACnet/HTTP em linguagem C voltado para sua execução multi-plataforma. Também é objetivo realizar testes de portabilidade dos códigos desenvolvidos, executando os programas em STBs com as mesmas características dos comercializados no Brasil, além de outras plataformas embarcadas. Também se tem o objetivo de desenvolver um navegador específico em linguagem C para servir de interface gráfica no STB, provendo acesso às funcionalidades de uma rede de automação doméstica.

1.3.2 Objetivos Específicos

- Estudar, projetar, desenvolver e testar um *gateway* BACnet/HTTP, desenvolvido em linguagem C e multi-plataforma;
- Projetar e simular um dispositivo BACnet.
- Projetar e utilizar, nos casos de uso, ferramentas de criação de conteúdos HTTP para utilização do *gateway* BACnet.
- Estudar plataformas abertas de *hardware-software* sobre as quais STBs comerciais têm sido desenvolvidos.

- Desenvolver um navegador de uso específico para acesso a rede BACnet através do *gateway* BACnet/HTTP voltado para o uso em STBs.
- Desenvolver um dispositivo BACnet simples em microcontroladores a fim de ser subsídio aos testes.

1.4 JUSTIFICATIVA

Atualmente existe uma demanda crescente para a utilização do sistema de TV digital no Brasil e no mundo, através de incentivos de órgãos governamentais. No Brasil, as transmissões de TV analógicas deverão ser encerradas no ano de 2016 de acordo com a legislação em vigor a partir de projeções da ANATEL (Agência Nacional de Telecomunicações) em Audiência Pública na Câmara dos Deputados (MINASSIAN, 2009). Levando em conta a grande aceitação dos aparelhos de TV analógica no Brasil e supondo que todos estes aparelhos receberão um conversor de TV digital, ou serão trocados por aparelhos com STB integrado, haverá uma presença massiva de STBs em todas as classes sociais brasileiras até o fim das transmissões analógicas. Alinhado a isto, o Brasil vem passando por um período de grande prosperidade econômica, com um aumento significativo da classe média. Neste cenário promissor, onde se tem um aumento do poder aquisitivo dos brasileiros, deverá se observar o aumento do interesse por produtos de conforto, dentre os quais as soluções de automação residencial.

A presente dissertação tem por objetivo equacionar este cenário, buscando gerar ferramentas e processos que darão suporte aos trabalhos em solução de automação residencial, utilizando o poder computacional de um STB de uma forma otimizada, tornando assim o aparelho de TV um supervisório de baixo custo, ao agregar funcionalidades ao mesmo. Este acréscimo de funcionalidade é possível na abordagem desta dissertação através de uma simples atualização de *firmware* do STB. Contudo, após a consolidação e comercialização de STBs com *midlewares*, os conceitos e protótipos desta dissertação poderão ser portados para uma solução de aplicativo sobre o *midleware* adotado, sem grandes modificações.

Dentre os diversos padões abertos de automação predial, optou-se nesta dissertação pelo BACnet, devido à especificação livre de custos e por ser um padrão internacional, onde diversos fabricantes produzem atuadores e outros equipamentos, com os mais diferentes propósitos e valores. Além disto, esta dissertação fornece um ferramental teórico

básico para que, a partir do protótipo de um dispositivo BACnet piloto desenvolvido e apresentado, possa-se confeccionar atuadores e sensores BACnet a um custo atrativo e que possam ser comprados por pessoas em diversas classes sociais, promovendo uma "inclusão de automação residencial", algo semelhante a "inclusão digital" em domótica.

1.5 CONTRIBUIÇÕES DO TRABALHO

A presente dissertação visa à inserção do programa de Sistemas Mecatrônicos da Universidade de Brasília na elaboração de projetos e pesquisa do protocolo BACnet para automação predial em geral, trazendo um aprofundamento do conhecimento dentro do grupo de pesquisa do GRACO[1], provendo uma documentação acessível, com modelos e códigos de exemplo.

A dissertação visa também a configuração de um ambiente de desenvolvimento de *software* para receptores de TV digital, sendo este um trabalho de base, onde um dos produtos é a configuração de ambientes e de plataformas de desenvolvimento.

Apresenta-se também uma metodologia de integração de sistemas multi-plataforma, através da concepção de diversas camadas de integração e de abstração, com a aplicação de filosofias e padrões de programação consagrados no mercado, buscando uma forma mais ágil e intuitiva de programação para sistemas embarcados.

Por fim, se tem a definição de um *gateway* que pode ser utilizado por programadores medianos, sem conhecimentos avançados em automação predial/residencial. Esta dissertação buscou uma estruturação de código que visa à facilitação da produção de conteúdos ricos de automação predial/residencial, seja este embarcado dentro de um receptor de TV digital ou em outro dispositivo qualquer.

1.6 METODOLOGIA DE PESQUISA

Para a realização desta dissertação foi adotada a seguinte metodologia de pesquisa:

1. Estudo dos aspectos tecnológicos de TV digital, ferramentas e plataformas;

[1]Grupo de Automação e Controle - UnB

2. Estudo dos fundamentos tecnológicos de automação residencial: realização de um estudo aprofundado de terminologias, aspectos de conforto, dentre outros;

3. Estudo de técnicas para automação residencial: realização de um levantamento e de uma análise de outras soluções de automação residencial;

4. Escolha dos protocolos e padrões a serem adotados: onde foram analisados mais profundamente os diferentes protocolos de automação residencial e os diferentes padrões de meio físico;

5. Análise dos produtos do mercado: foram analisadas as famílias de produtos e, a partir deste estudo, foi elaborada a especificação do sistema objeto desta dissertação;

6. Desenvolvimento do ambiente de projeto para sistemas domóticos integrados com tv digital;

7. Desenvolvimento do *gateway* BACnet/HTTP;

8. Teste do ambiente e do *gateway* desenvolvido.

1.7 RESULTADOS ALCANÇADOS

Os resultados alcançados com a realização desta dissertação podem ser divididos, para uma melhor compreensão, nos seguintes subgrupos: (a) protótipo de sistema desenvolvido, (b) formação de conhecimento para o grupo de pesquisa e (c) formação de conhecimento para o pesquisador.

1.7.1 Protótipo de sistema desenvolvido

Desenvolvimento de um ferramental de *software* de automação residencial capaz de ser executado dentro de um receptor de TV digital (STB) ou em outras plataformas, sendo capaz de acionar um dispositivo do tipo digital, ou seja, que pode assumir o estado *ligado* e o *desligado*. Também se tem o desenvolvimento de um navegador de uso específico capaz de executar em ambientes limitados, como sistemas embarcados e STBs.

1.7.2 Formação de conhecimento para o grupo de pesquisa

Os resultados alcançados no âmbito de formação de conhecimento para o grupo de pesquisa podem ser enumerados da seguinte forma: (a)pesquisa, análise e customização de um ambiente de desenvolvimento para *softwares* embarcados dentro de receptores de TV digital da família ST40, (b) análise do protocolo BACnet, servindo com referência bibliográfica, (c) pesquisa e análise de redes sem-fio, em especial a IEEE 802.15.4 (ZIGBEE, 2010), (d) pesquisa e análise de uma pilha BACnet popular, que pode ser utilizada em outros projetos no âmbito de automação dentro do programa de mestrado em Sistemas Mecatrônicos e em toda a comunidade científica.

1.7.3 Formação de conhecimento para o pesquisador

Houve crescimento pessoal do mestrando as seguintes áreas: No âmbito desta dissertação obteve-se um aprofundamento do conhecimento em sistemas embarcados, bem como a aquisição de conhecimento de receptores de TV digital. No âmbito acadêmico, o pesquisador também desenvolveu suas habilidades de pesquisa, através da confecção desta obra e da leitura de artigos científicos. No âmbito técnico, ocorreu um aumento do conhecimento de diversas linguagens de *software* e de metodologias de desenvolvimento sistemático de produtos.

1.8 ORGANIZAÇÃO DO TRABALHO

O presente trabalho é organizado tal como apresentado na figura 1.5 e descrito da seguinte forma:

O capítulo 2 apresenta a fundamentação necessária para o desenvolvimento desta dissertação, abordando assuntos como aspectos de automação predial, receptores de TV digital e microcontroladores, além de recursos de *software* para o desenvolvimento dos programas.

O capítulo 3 descreve o desenvolvimento do trabalho, bem como a metodologia de desenvolvimento adotada para os estudos dos aspectos tecnológicos das TVs Digitais e da Automação Predial, e as escolhas baseadas nestes estudos. Engloba também as

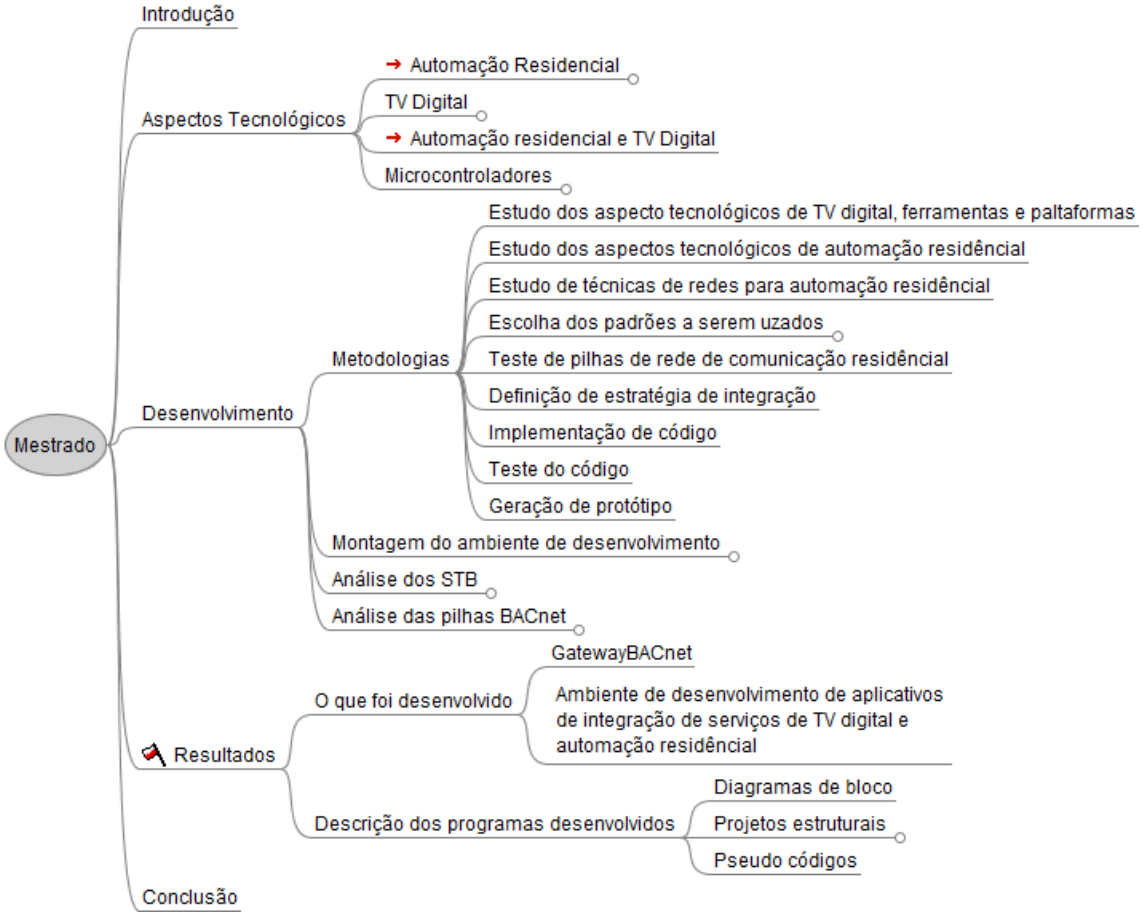

Figura 1.5: Diagrama de estrutura do texto

análises comparativas dos Receptores de TV digital comerciais e as pilhas BACnet de uso livre.

O capítulo 4 descreve os resultados obtidos e a descrição do programa desenvolvido.

O capítulo 5 apresenta as conclusões e os trabalhos futuros.

2 ASPECTOS TECNOLÓGICOS

2.1 INTRODUÇÃO

Durante a fase de revisão bibliográfica e análise das tecnologias que integram o estado-da-arte, foram estudadas várias tecnologias de *software*, arquiteturas de sistemas e *hardwares* específicos, permitindo a aquisição de novos conhecimentos, bem como a fixação de conceitos previamente obtidos. O estudo aprofundado dos tópicos de sistemas de *hardware* e *software* se fez mais necessário observando esta dissertação no contexto do programa de Mestrado em Sistemas Mecatrônicos, onde o protocolo BACnet e os receptores de TV digital (do inglês *set-top box* ou simplesmente STB) ainda não foram aproveitados como tema de pesquisa.

Para a realização da presente dissertação foi necessário, primeiramente, o aprofundamento dos conhecimentos em automação predial, seguido de um estudo das características e particularidades do protocolo BACnet e da TV digital, assim como a busca na literatura de integrações de TV digital com automação residencial.

Segue-se no estudo dos aspectos tecnológicos a compreensão mais aprofundada dos microcontroladores e dos sistemas que eles integram: (a) SoC *Sistem on Chip*-Sistema em um CI e (b) SBC (*Single Board Computer*-Computador de Única Placa), devido ao fato dos STBs estarem sendo desenvolvidos com estes componentes. Uma breve leitura dos controladores lógicos programáveis (CLPs) também se faz necessária pelo fato destes serem atualmente os principais agentes de automação residencial e industrial.

Por fim, foi realizada uma leitura referenciada das características do compilador de *software* GCC (STALLMAN, 2002), responsável pela compilação dos programas desenvolvidos, tanto o *gateway* BACnet HTTP e o navegador de acesso aos aplicativos de automação, nas versões que foram testadas no STB comercial adotado quanto no SBC (TS7100) de referência de portabilidade. Também foi estudado o programa Eclipse (ECLIPSE, 2010), que serviu de ambiente de desenvolvimento de *software*, devido às diversas qualidades que o tornaram a ferramenta de desenvolvimento para diversas arquiteturas. O conjunto de ferramentas GCC e Eclipse também foi utilizado para o

desenvolvimento do *firmware* do dispositivo BACnet de referência, que se utiliza de um microcontrolador AVR.

2.2 AUTOMAÇÃO RESIDENCIAL E PREDIAL

A automação residencial e predial possui vários nomes que variam de acordo com a intenção de enfatizar alguma dimensão específica pelos grupos de pesquisa que elaboram e redigem os textos técnicos sobre o assunto. Geralmente, quando se utiliza o termo automação residencial (*Home automation*), se tem o intuito de enfatizar as atividades que envolvem uma casa, procurando se isentar dos problemas de ruídos que geralmente se encontra em um ambiente fabril. Em outros trabalhos que tratam do assunto é costume encontrar o termo automação predial (*Building Automation*), que visa ser mais genérico, podendo atender também a prédios comerciais e a hotéis, tal como ocorre com a própria definição do BACnet (ASHRAE135, 1995), onde as letras BA significam *Building Automation*, cujo interesse abrange diversos tipos de edificações, desde os prédios comerciais até os residenciais.

2.2.1 Aspectos básicos de automação residencial

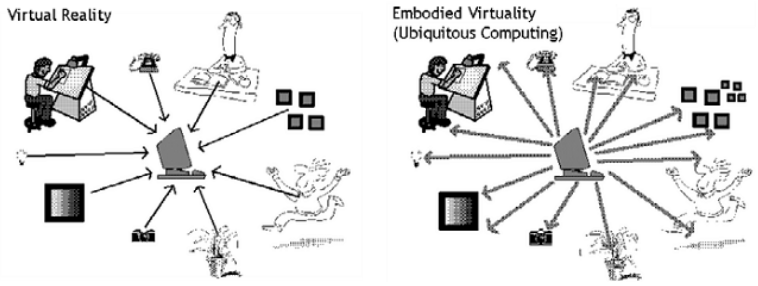

Figura 2.1: Quadrinho de (WEISER, 1991) mostrando a diferença entre computação ubíqua e realidade virtual

No trabalho de (WEISER, 1991) foi introduzido o termo "*computação ubíqua*", no seu artigo *The Computer for the 21st Century* para a revista *Scientific American*, que recebeu um razoável número de seguidores (MüHLHäUSER; GUREVYCH, 2009). Neste contexto, a computação ubíqua objetiva o uso de computadores em todos os espaços (lugares), diferentemente da *realidade virtual*(vide figura 2.1), que visa colocar todos os lugares dentro do computador. O mesmo conceito é diferente da *inteligência artificial* no seu modo mais puro, pois ao invés de se ter um grande conhecimento em um grupo reduzido de máquinas, se tem um conhecimento reduzido em um grupo grande de

máquinas. E por fim, a computação ubíqua se distingue da tecnologia de agentes na sua concepção inicial, quando não busca ser um intermediário entre o usuário e o mundo computadorizado, buscando não excluir as interfaces de usuário e a interação humana com os computadores. Outro termo também aplicado é *domótica*, que trata mais da gestão dos recursos habitacionais (LITZ; GROSS, 2007; ARAYA, 1995).

Apesar dos benefícios relevantes para a qualidade de vida das pessoas, a *computação ubíqua* (WEISER, 1991) vem sendo tratada de uma forma marginal, realizando atividades efêmeras como parar no andar correto ou chamar as pessoas pelo nome, o que não parece interessante devido a uma série de fatores, como (a) os altos investimentos necessários para a confecção dessas funcionalidades, (b) o alto risco associado e (c) muitas pessoas rejeitarem novas tecnologias devido a quesitos pessoais (ARAYA, 1995). Por outro lado (no contexto de automação residencial e predial) existe o termo Ambiente Inteligente (*Ambient Intelligence*)[1] (MüHLHäUSER; GUREVYCH, 2009; FRIEDEWALD et al., 2005) que é mais utilizado nos artigos europeus (MüHLHäUSER; GUREVYCH, 2009).

Nesta dissertação, os termos referenciados e discutidos serão utilizados sem grandes distinções, tendo como foco a tecnologia envolvida para a geração do conforto e segurança humana através da utilização de vários dispositivos com poder computacional e interface com usuário, quando for necessário.

2.2.2 Automação residencial, trabalhos correlatos

Durante a etapa de preparação desta dissertação foram analisados vários trabalhos versando sobre automação residencial em seus diversos aspectos, dentre os quais foram selecionados alguns mais relevantes, por conterem conceitos ou produtos que convergem com os interesses desta dissertação, para serem apresentados mais profundamente.

(LITZ; GROSS, 2007) desenvolveu um trabalho na Universidade de Kaiserslautern, Alemanha, que trata da definição de conceitos que envolveram a elaboração de um sistema de nome PAUL (*Personal Assistant Unit for Living*). Neste trabalho, ocorre a definição das necessidades básicas das pessoas idosas, porém que pode ser extrapolado para todas as idades de pessoas, através do diagrama esquemático da figura 2.2:

[1]Nota de tradução. Não é interessante traduzir para “Inteligência ambiental"devido ao forte conceito ecológico que permeia nossa sociedade

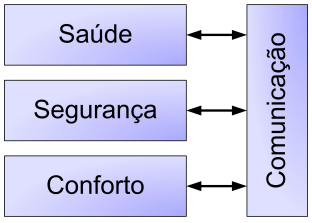

Figura 2.2: Necessidades básicas das pessoas (idosas) (Fonte: (LITZ; GROSS, 2007) modificado)

Esta estrutura mostra as necessidades de saúde, segurança e conforto, todas estas dependendo de um sistema de comunicação, que se torna uma necessidade básica para atender outras necessidades primárias. Também a comunicação desempenha um papel importante como necessidade por si só, visto que as pessoas são seres sociais que precisam de relacionamentos com familiares, amigos, vizinhos e prestadores de serviços.

Um fato interessante é a ordem hierárquica implícita colocada na figura 2.2, com a saúde no topo, seguida da segurança e por último do conforto. Esta priorização é coerente, se for levado em conta a manutenção da integridade física e psicológica do indivíduo, onde se coloca no topo o que pode causar a deteriorização mais imediata do indivíduo e que pode ser causa da morte do mesmo.

A busca de práticas que levam ao conforto em aplicações domóticas é vista através de dois pontos principais nas análises conceituais de (LITZ; GROSS, 2007), para a elaboração do sistema PAUL, sendo elas: (a) busca pela facilitação da vida cotidiana e (b) desenvolvimento de sistemas que venha a atuar antes do início de problemas de saúde. Para a segurança é seguida a mesma métrica, onde se tem conceitualmente dois pontos de atenção: (a) os problemas vindos de dentro do próprio ambiente ou prédio, tais como vazamentos de água, controle de incêndios, dentre outros. e (b) problemas vindo de fora do ambiente ou prédio, tal como os assaltantes.

No trabalho de referência (LITZ; GROSS, 2007) ocorre uma análise dos aspectos importantes a serem tratados sobre a saúde, definindo uma premissa tão interessante quanto importante: “deve-se tentar com aplicações domóticas coibir as situações perigosas para a saúde, e se não for possível, deverá ocorrer a sua previsão”. A subdivisão dos riscos a saúde a despeito dos outros pontos de atenção foi dividido conceitualmente em 3 classes:

a) *Situações com risco de vida, com ação repentina:* como derrames cerebrais e ataques cardíacos.

b) *Situações sem risco de vida, com ação repentina:* como quedas onde a pessoa não levanta.

c) *Situações sem risco de vida, sem ação repentina.*

Na análise teórica define-se que a comunicação se dá por dois motivos: (a) entrar em contato com outras pessoas, como familiares e vizinhos e (b) na obtenção de serviços. Com esta análise se tem uma nova dimensão hierárquica, que é a variação de tempo da ocorrência danosa a integridade do indivíduo.

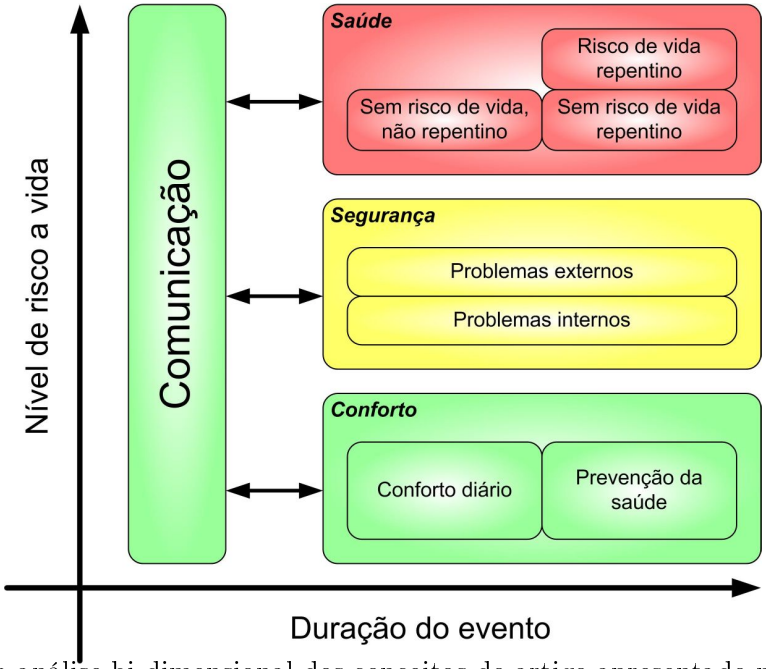

Figura 2.3: Nova análise bi-dimensional dos conceitos do artigo apresentado nesta dissertação

Ao ser analizado o texto de (LITZ; GROSS, 2007), observou-se que faria sentido analisar este trabalho em um gráfico bi-dimensional, onde se tem em um dos eixos uma ordenção da *duração do evento* a ser analizado, sendo que por exemplo, a comunicação teria uma duração bem reduzida, representando a ocorrência da transissão de uma dada informação em um intervalo de tempo pequeno. A outra dimensão seria o *nível de risco à vida*, onde uma situação de conforto repesentaria um nível de risco mais baixo, bem como uma situação de segurança teria um nível intermediário e uma situação de saúde seria de risco elevado. Nesta análise a comunicação representaria um papel fundamental em todos os *níveis de risco à vida.*

Com a análise bi-dimensional dos pontos a serem abordados em uma aplicação domótica tal como apresentada na figura 2.3, verifica-se que podem vir a existir algumas outras especificidades que poderiam ser abordadas, como por exemplo situações com risco de

vida e de ação lenta, tal como ocorre em um envenenamento por gás ou alergia do indivíduo a algum produto da mobília, além da subdivisão da segurança em problemas imprevisíveis e previsíveis, dentre outras análises.

A concepção do sistema PAUL foi embarcada em um TablePC com rede sem fio, onde as informações da casa são transmitidas através do protocolo KNX (EN50090, 1996),(ISOIEC14543, 2007), sendo voltado para um apartamento de $65m^2$. O sistema PAUL foi desenvolvido para ser utilizado pela população mais idosa, sendo a sua interface voltada para menus intuitivos e botões grandes, onde o usuário com pouca capacidade motora pode selecionar a funcionalidade desejada. Tal preocupação também foi observada no desenvolvimento do *gateway* BACnet, principalmente na confecção de uma interface intuitiva. O sistema PAUL foi projetado para possuir um número finito de sensores, onde estes foram reaproveitados para os módulos desenvolvidos, através de um política de reaproveitamento das informações, apresentado nos diagramas figura 2.4 e figura 2.5.

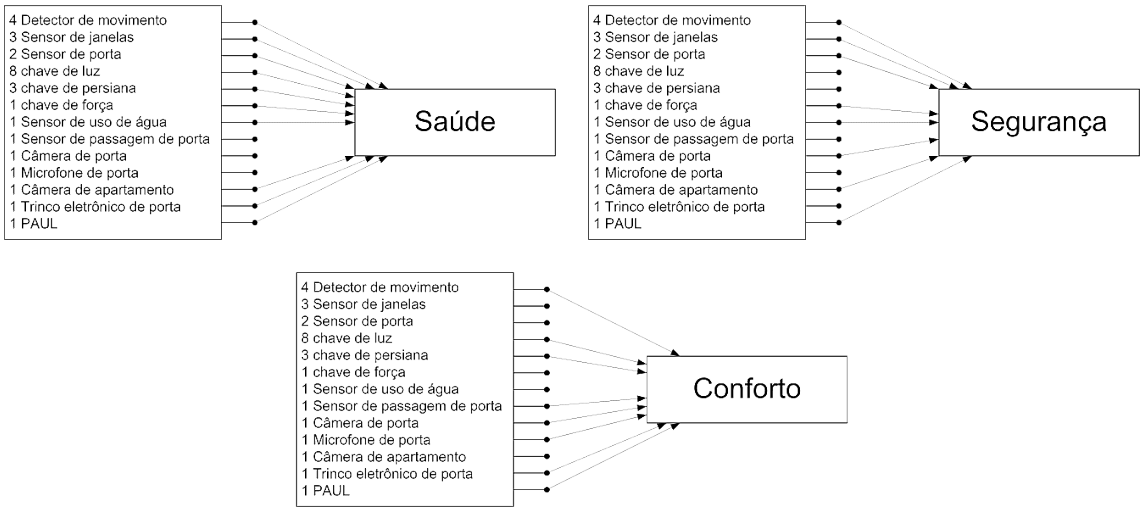

Figura 2.4: Política de reaproveitamento de informações de sensores. (Fonte: (LITZ; GROSS, 2007) modificado)

As informações foram hierarquizadas em três níveis dentro do PAUL, de acordo com o nível e a profundidade de processamento até a obtenção de cada informação, tal como apresentado na figura 2.5.

As informações primárias são aquelas que passaram por menos processamento, sendo compreendidas pelas contidas nos sensores. Já as secundárias são aquelas que se baseiam nos estados do apartamento definido como modelo, através de uma sequência de eventos provenientes dos sensores. Por outro lado, as informações finais são as de mais alto nível de abstração, após um processamento mais profundo, que no caso do PAUL

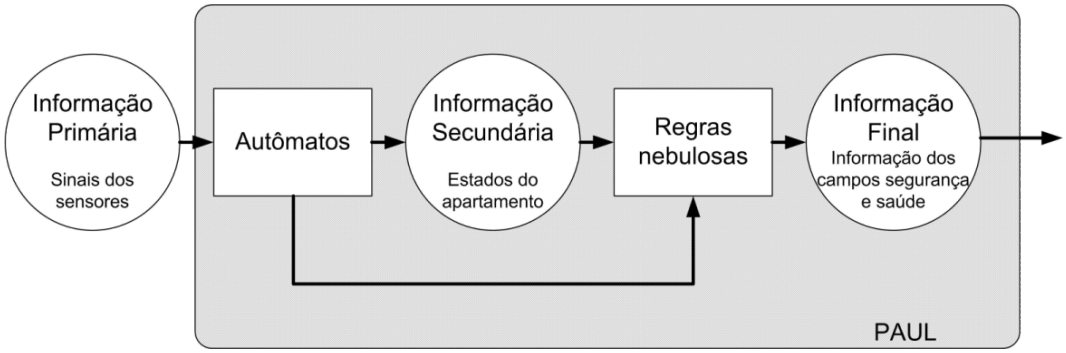

Figura 2.5: Hierarquia de informação e microestrutura do PAUL (Fonte: (LITZ; GROSS, 2007)-modificado)

foi concebido com lógica nebulosa.

O trabalho apresentado por (FRIEDEWALD et al., 2005) tráz à tona um conceito interessante que é o *AmI (Ambient Inteligence)*. A inteligência do ambiente não é uma predição realista, mas sim uma visão desejada do futuro, com grandes incertezas, e muitas vezes ocorrendo que vários ambientes com inteligência trazem um aumento da complexidade do cotidiano ao invés de diminuí-lo.

As características fundamentais dos *AmI* são a descentralização, transparência e inteligência, onde dispositivos embarcados nos mais diferentes aparelhos venham a prestar funcionalidades com interfaces inteligentes e intuitivas, muitas vezes tomando decisões de uma forma transparente para as pessoas.

Os pontos fortes da visão *AmI* (vide figura 2.6) são as definições de sistemas: (a) orientados a pessoas, (b) familiares ao usuário, (c) não intrusivos, (d) controláveis e (e) voltados para serem utilizados por um grande número de pessoas.

Como estes *pontos fortes* da visão *AmI* estão altamente inter-relacionados e não podem ser hierarquizados, fez sentido durante a análise do trabalho de (FRIEDEWALD et al., 2005) para a elaboração desta dissertação, a representação destes *pontos fortes* de forma circular, pois todos indicam a qualidade do sistema domótico e seu nível de aceitação popular.

Para ilustrar estes *pontos fortes*, um sistema domótico deve ser *familiar ao usuário* de tal forma que os habitantes não encontrem dificuldades para a operação do sistema. Deve ser também *não intrusivo*, evitando inflingir a seus usuários uma alteração negativa do seu modo de vida e dando opção a este de, mesmo em cenários positivos, evitar a alteração do seu modo de viver. Ao se falar que um sistema domótico tem que ser

Figura 2.6: Pontos fortes da visão *AmI* - Visão pessoal deste trabalho

controlável, deve-se garantir que o usuário possa enviar comandos ao sistema, mesmo que este possua sensores que percebam a intensão do mesmo.

Um outro ponto importante é a *universalização* dos sistemas domóticos, abrangendo todas as pessoas, em qualquer classe social, idade ou grau de instrução. A *orientação a pessoas* indica um formato de desenvolvimento de sistemas domóticos, levando o desenvolvedor a pensar primeiramente em pessoas. Um paralelo com este *ponto forte* é a *programação orientada a objetos* (POO), onde se codifica pensando em objetos e, no caso da domótica, deve-se pensar primeiramente em pessoas.

Uma definição interessante levantada é a de casa inteligente (*Smart House*) (Scott, N., 1998), onde as principais características se referem a sistemas que cumprem as seguintes condições: (a) ajudar seus habitantes a viver com saúde, felicidade e de forma segura, (b) realizar muitas tarefas automaticamente para minimizar o desgaste físico e psicológico em gerenciar a casa, (c) integrar a casa, o trabalho, o estudo e as atividades recreativas e (d) não preocupar as pessoas com detalhes técnicos de como ela, a casa, está atualmente funcionando.

Ao analisar os conceitos de casa inteligente é importante também analisar os conceitos de *casa*, *lar* e *domicílio*, a fim de esclarescer cada um destes termos, através da compreensão das características intríscicas dos mesmos. Esta compreensão é importante tanto para a correta utilização dos termos quanto para a compreensão do nível de relacionamento das pessoas com os lugares onde vivem.

Segundo o dicionário Aurélio, *casa* é um "Edifício de um ou poucos andares, destinado, geralmente, a habitação; morada, vivenda, moradia, residência"(HOLANDA, 2010). *Domicílio* é definido como "casa de residência; habitação fixa.". e, lar (s.) é uma forma especial de se definir a casa ou os assuntos relacionados a ela, como a convivência com a família e os vizinhos. *Lar* pode ter uma conotação sentimental ou carinhosa.

Em uma qualificação mais informal, quando falamos de *casa*, estamos nos referenciando ao prédio, edifício, *domicílio* ao lugar onde o habitante passa o maior tempo ou tem como residência oficial. Já no caso de *lar* podemos atribuir um caráter mais sentimental e filosófico. Essa definição destes termos esbarra em alguns problemas de tradução do inglês.

A automação residencial conforme proposto por (FRIEDEWALD et al., 2005) é realizada através de um *aplicativo doméstico*, que é um *software* que é executado nos diversos elementos de automação presente na residência. Ao definir as funções de um aplicativo doméstico (FRIEDEWALD et al., 2005) enumeram os seguintes aspectos: (a) automação residencial (suporte as funções básicas da casa), (b) comunicação e socialização, (c) descanso, refresco, entretenimento e esporte e por fim (d) trabalho e estudos.

Com base no trabalho de (FRIEDEWALD et al., 2005) pode-se traçar uma nova enumeração das funções de um aplicativo doméstico, buscando uma visão menos vinculada à cultura do autor, onde, por exemplo, não se precisa colocar no mesmo grupo de atividades os trabalhos e os estudos. Essa nova enumeração poderia se definida da seguinte forma: (a) suporte as atividades de manutenção da casa e (b) suporte à existência de habitantes.

Nesta subdivisão, a função do aplicativo doméstico em dar suporte a existência dos seus habitantes pode ser subdividida em: (a) manutenção da integridade física dos habitantes, (b) manutenção da integridade psicológica dos habitantes e (c) manutenção da integridade social dos habitantes. Outra enumeração poderia simplesmente subdividir o suporte à existência dos habitantes em: (a) atividades de lazer e (b) atividades obrigatórias; porém esta subdivisão apresenta-se muito negativa à realização do trabalho e vinculada à percepção de cada habitante do que é obrigatório ou lazer.

Segundo (FRIEDEWALD et al., 2005), muitas funções de automação doméstica existem hoje sem algum tipo de inteligência. Um contraste com estes sistemas é o exemplo dado pelo mesmo autor, onde se tem uma funcionalidade de ligar a TV, onde o canal

desejado é selecionado automaticamente, a intensidade luminosa natural (persianas) e artificial (lâmpadas) e a temperatura são ajustadas, entre outras funcionalidades. Este comportamento é plenamente viável através do desenvolvimento de uma aplicação que se utilize do *gateway* BACnet. Nas funcionalidades de automação residencial devem ser levado em conta os seguintes tópicos: (a) suporte as funções básicas da casa, (b) segurança e (c) suporte a vida independente. O autor também divide o contexto de automação em três visões, baseadas nas funcionalidades da automação doméstica, sendo estas:

- Considerando as funcionalidades de *comunicação e socialização* deve se observar como ocorrem os relacionamentos entre os indivíduos e o relacionamento da pessoa com a sociedade ao qual este está inserido, podendo estes relacionamentos ser enumerados da seguinte forma: (a) pessoa à pessoa, (b) pessoa à comunidade e (c) novas formas de socialização.
- Tendo em conta as funcionalidades de *descanso, refresco, entretenimento e esporte* se tem a seguinte relação de itens: (a) repouso e descanso, (b) refresco e higiene, (c) entretenimento e *hobbies* e (d) esporte e boa forma.
- Com relação as funcionalidades de dar *suporte ao trabalho e estudos* se tem a seguinte relação de itens: (a)trabalhos domésticos, (b) trabalho de manutenção e (c) trabalho em casa e estudos.

Um dos grandes desafios na elaboração de projetos de casas inteligentes é propiciar o aumento da qualidade de vida dos seus habitantes através da elaboração de sistemas e métodos de identificação da identidade das pessoas. Tal identificação pode ocorrer através da análise de hábitos, gostos, padrões de conduta, dentre outros (FRIEDEWALD et al., 2005).

Uma preocupação que é inerente ao processo de automação é que as funcionalidades fornecidas pelo processo não podem levar a uma deteriorização do corpo e da mente, principalmente das pessoas idosas, pela retirada das funções inerentes a condição humana. Um dos grandes desafios da domótica é prover soluções de alta qualidade, propiciando umaa melhor da qualidade de vida para seus habitantes. Esta melhora pode vir através estimular a solicalização das pessoas, além de propiciar um ambiente saudável. Estes tópicos e idéias apresentados no trabalho de (FRIEDEWALD et al., 2005) podem ser suportados pelas propostas desta dissertação, tendo em conta os seguintes aspectos:

1. O BACnet serve para automatizar a casa, integrando vários dispositivos automáticos, como luzes, persianas, dentre outros.

2. Sendo um protocolo de comunicação, o BACnet realiza a comunicação ao nível de dispositivos da casa, prove um meio de comunicação entre os usuários e a casa de uma forma mais amigável, através de um aparelho de TV ou aplicação embarcada. Neste contexto, a TV pode ser vista como um bom exemplo de dispositivo social.

O *gateway* BACnet proposto nesta dissertação vem a dar suporte a diversas implementações de aplicativos domésticos, sendo ilustradas algumas possibilidades na figura 2.7. Por exemplo, pode-se ter o acionamento de persianas e iluminação artificial, a fim de proporcionar maior conforto luminoso e economia de energia elétrica, além do acionamento do sistema de refrigeração; estes são exemplos de aplicativos domésticos, onde todas as operações podem ser realizada de forma automática ou pela ação do usuário através do uso do controle remoto, do aparelho de TV e do STB.

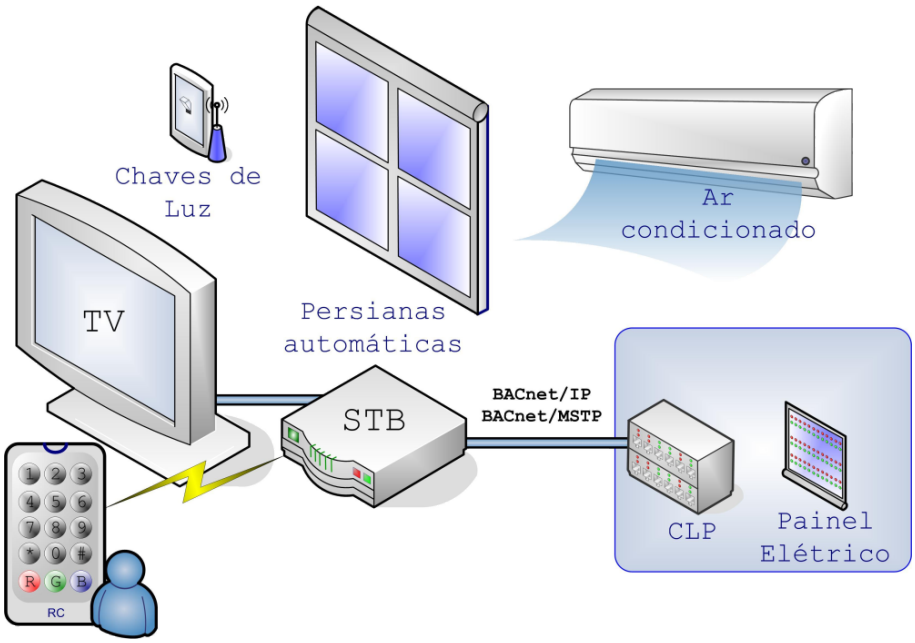

Figura 2.7: Exemplo de aplicação inteligente através do uso do *gateway* BACnet

A leitura destes trabalhos orientou o desenvolvimento desta dissertação, que vinha sendo muito voltado para as tecnologias e menos para a entidade fim, que é o ser humano. Um dos pontos de desenvolvimento em que se percebe esta intenção é na concepção da interface gráfica apresentada no navegador *X* (vide seção 3.2), agora voltada para pessoas idosas ou com alguma limitação.

2.3 PROTOCOLOS DE AUTOMAÇÃO RESIDÊNCIAL E PREDIAL

Para que ocorra uma automação residencial e predial faz-se necessário que se tenha mais de um componente de automação, e estes componentes têm que se comunicar entre si de uma forma efetiva e não ambígua, através de um protocolo. Mesmo que se tenha uma solução mínima de automação residencial como, por exemplo um portão automático, observa-se claramente dois componentes a se comunicarem, onde de um lado se tem um controle-remoto e de outro um atuador para o motor elétrico.

Ao longo do tempo, diversas empresas e instituições vêm desenvolvendo, aprimorando e definindo, diversos protocolos de automação residencial e predial, os quais podem ser divididos em dois grandes grupos: (a) protocolos abertos e (b) protocolos proprietários. Segue uma análise mais aprofundada de alguns exemplos de protocolos de automação a partir da divisão nestes dois grupos.

2.3.1 Protocolos de automação residencial e predial abertos

São considerados protocolos de automação residencial abertos aqueles que possuem sua definição acessível à qualquer pessoa ou companhia, seja pela aquisição gratuita de sua definição ou pela compra de uma norma de uma unidade certificadora (tal como a ANSI, a EIA ou a ABNT). A seguir serão descritos alguns exemplos de protocolos utilizados em automação predial, como o Konnex, LonWorks, CAN e BACnet.

2.3.1.1 Konnex

Konnex é um protocolo conhecido como EIB/KNX *European Installation Bus* (EN50090, 1996; ISOIEC14543, 2007), sendo mantido pela *EIB Association* e pela *Konnex Association*. As certificações do protocolo Konnex são de responsabilidade da *Konnex Association* (KASTNER, 2005). O EIB/KNX é um protocolo que opera em diversos meios físicos, dentre eles par trançado (a 29 VDC), rádio frequência (na faixa de 868 MHz) e IP, podendo conter um máximo de 60000 dispositivos dentro de uma rede. O protocolo EIB/KNX é um protocolo baseado em objetos, onde existem objetos de interface de sistema e objetos de interface de aplicativos. Maiores explicações sobre essa filosofia será fornecida no item 2.3.1.4 deste parágrafo sobre o BACnet.

2.3.1.2 LonWorks

LonWorks é um protocolo desenvolvido pela Echelon Corp e publicado como padrão oficial em 1999 sob o número ANSI/EIA-709. O sistema LonWorks (vide figura 2.8) é composto pelo protocolo de comunicação, por um controlador dedicado (*Neuron Chip*) e uma ferramenta de gerenciamento de rede.

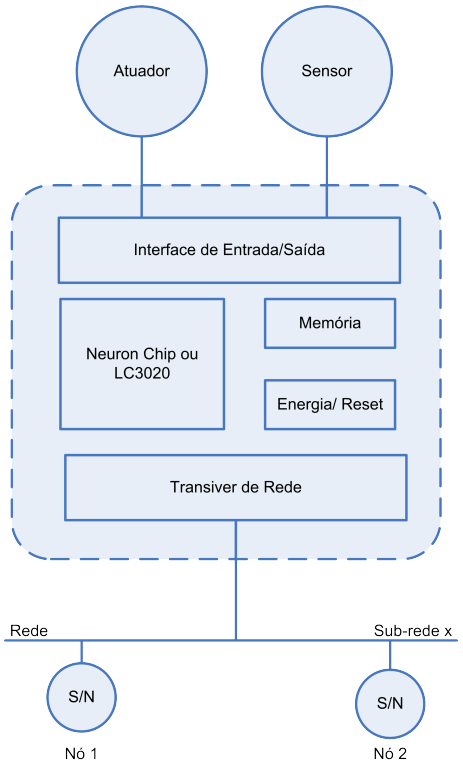

Figura 2.8: EIA-709 Arquitetura de nodos.Fonte: (KASTNER, 2005) (Modificado)

O *Neuron Chip* é um SoC (vide seção 2.7.1 (ECHELON, 2009)) desenvolvido pela própria Echelon Corp, que já possui em sua memória ROM um sistema operacional, o protocolo LonWorks e bibliotecas para as funcionalidades de IO. Sua versão mais atual é o Neuron 5000 (vide figura 2.9), com frequência interno de 80 MHz e com memória de 64KB de RAM e 16KB de ROM.

Outro SoC que implementa o protocolo LonWorks é o LC3020 (vide figura 2.10), desenvolvido pela Loytec (LOYTEC, 2009), que contém uma CPU ARM7-TDMI de 60 Mhz, uma porta Ethernet 10/100Base-T, memória de 8K de SRAM e 3K de ROM, além de ambiente de desenvolvimento Eclipse. O LC3020 pode ser utilizado com outros protocolos de automação residencial, dentre eles o BACnet.

Uma variação do protocolo ANSI/EIA-709 é o LonWorks/IP, registrado sob o número

Figura 2.9: chip Neuron 5000. Fonte: http://www.echelon.com/products/neuron/

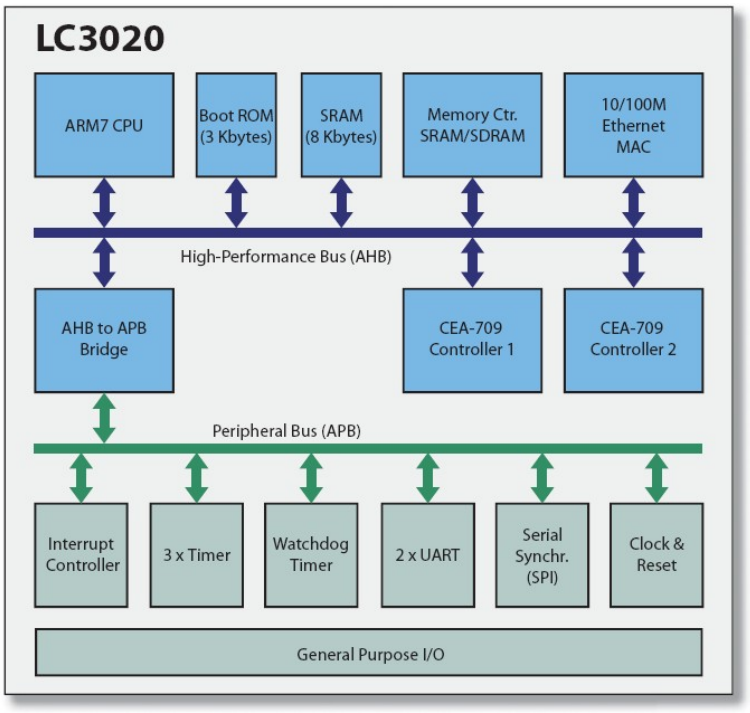

Figura 2.10: chip Neuron 5000. Fonte: (LOYTEC, 2009)

ANSI/EIA-852, que define tanto o uso do protocolo LonWorks através de um tunelamento por IP, quanto a definição de um protocolo sobre o protocolo IP (assim como ocorre com o TCP e o UDP). O Protocolo LonWorks possui também uma padronização para uma série de meios físicos (KASTNER, 2005), onde se faz necessário a instalação de um *transiver* de rede específico para cada tipo de rede na qual se deseja comunicar. Um meio físico interessante no qual existe padronização para o LonWorks é a rede elétrica (LP-10), além de fibra ótica.

A arquitetura de rede do LonWorks permite para cada domínio conter até 255 sub-redes com um máximo de 127 nós cada. Cada sub-rede geralmente corresponde a um canal físico, porém vários meios físicos podem compor uma sub-rede através de *gateways*

ou repetidores, além da possibilidade de coexistência de várias sub-redes no mesmo segmento físico.

2.3.1.3 CAN (*Controller Area Network*)

CAN é um protocolo desenvolvido pela empresa alemã Bosch ((GUIMARãES, 2003), (CORRIGAN, 2008), (CORRIGAN, 2008)) para ser utilizada em automóveis em um sistema de *múltiplos mestre* e com a capacidade de *broadcast* de mensagens. Este protocolo possui a capacidade de transmitir a uma taxa de sinalização de 1MBit por segundo (CORRIGAN, 2008).

O protocolo CAN se baseia no envio de pequenas mensagem a fim de garantir a integridade dos pacotes (CORRIGAN, 2008), bem como garante a existência de um grande número de nós interconectados, sendo que cada um pode ser considerado *mestre* em um dado momento. Este protocolo envia mensagens de forma serial síncrona, onde este sincronismo é dado a partir do início de cada mensagem lançada no barramento.

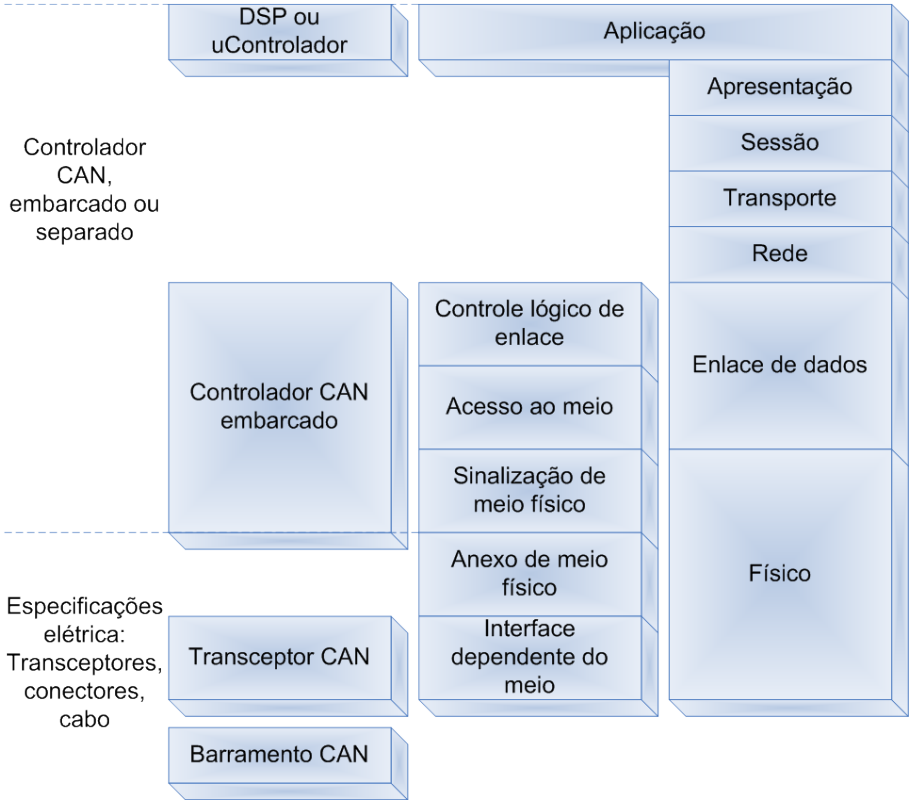

Figura 2.11: Comparativo do Protocolo CAN com o OSI Fonte: (CORRIGAN, 2008), modificado

O CAN é um protocolo com uma especificação que aborda as camadas de enlace de

dados e meio físico, especificando com profundidade o padrão elétrico a ser utilizado, por exemplo conectores, cabeamento, dentre outros. (vide figura 2.11). Por ser um protocolo robusto, serve de opção de automação em ambientes residenciais e prediais alta interferência a ruídos, tal como ocorre no acionamento de elevadores, onde os motores podem produzir ruídos eletromagnéticos suficientemente fortes para degradar a informação de controle. Um dos fatores determinantes na robustez do protocolo é a característica de que o bit zero não é representado por uma tensão ou corrente de valor nulo, tornando-se assim imune a ruídos eletromagnéticos (GUIMARãES, 2003).

2.3.1.4 BACnet (*Building Automation and Control networks*)

BACnet é um protocolo de comunicação (LIU; REN, 2007) voltado para a automação de edificações e seus sistemas de controle. Segundo a (PARSONS, 2009) a ASHRAE (*American Society of Heating, Refrigerating, and Air-Conditioning Engineers*) define o BACnet como "um grupo de regras governando a troca de informação através de uma rede de computadores". Estas regras estão descritas em um padrão especificado no padrão ANSI/ASHRAE Standard 135 (ASHRAE135, 1995) e no padrão ISO 16484-5.

O BACnet teve seu início oficial em junho 1987, quando comitê de projetos da ASHRAE buscou um padrão de que atendesse a necessidade de interoperabilidade entre muitos vendedores e muitas classes de automação predial(KASTNER, 2005). Durante a concepção do protocolo, se procurava um padrão aberto que pudesse ser implementado por qualquer vendedor e que de uma forma segura viria a se comunicar efetivamente com os equipamentos de outros vendedores ligados na mesma rede. Também foi alvo de interesse no desenvolvimento deste padrão o atendimento de vários tipos de automação, desde pequenas automações residências quanto a automação de grandes prédios, passando por exemplo pelo controle de HVAC, iluminação, segurança, combate a fogo, controle de aceso, manutenção dentre outros (PARSONS, 2009).

O BACnet tornou-se padrão ASHRAE/ANSI sob o número 135 em 1995 vindo a ser padrão ISO sob o número ISO 16484, mais especificamente na parte 5 (16484-5) em janeiro de 2003. O padrão ISO 16484 está dividido nas seguintes partes, sobre o título geral de *"Building automation and control systems"*, ou simplesmente BACS. Segue a título de informação, todas as partes do padrão ISO 16484:

1. *Parte 1*: Visão geral e definições

2. *Parte 2*: *Hardware*

3. *Parte 3*: Funções

4. *Parte 4*: Aplicações

5. *Parte 5*: Protocolo de comunicação de dados

6. *Parte 6*: Teste de conformidade de comunicação de dados.

7. *Parte 7*: Especificação de projetos e implementação.

a) Divisão administrativa do BACnet: a ASHRAE criou uma estrutura de "grupos de trabalho" (*Working Groups WG*) responsável pelo estudo de cada uma das vertentes do protocolo mais importantes. Atualmente, existem os seguintes WGs, sendo que este grupo vem sendo continuamente alterado. A seguir serão enumerados os principais WGs e suas respectivas atribuições.

1. *AP-WG (Applications)*: este grupo de está responsável pela criação de novos macro-objetos em uma perspectiva dos usuários. O mesmo está muito ligado ao OS-WG.

2. *EL-WG (Elevator)*: este grupo de trabalho está buscando a facilitação da monitoração de sistemas de elevador com o protocolo BACnet.

3. *IP-WG (Internet Protocol)*: este grupo de trabalho está responsável pela evolução do BACnet/IP, como por exemplo a sua adequação ao IPv6.

4. *IT-WG (Information Technology)*: é um grupo que verifica as futuras necessidades de comunicação dos BACSs.

5. *LA-WG (Lighting Applications)*: tal como o que ocorre com os elevadores, este grupo de trabalho está buscando melhores aperfeiçoamentos do BACnet para atender aos sistemas de iluminação. Desde a sua fundação, este grupo teve apoio de empresas do ramo, e já resultou na elaboração de um objeto novo, o *Lighting Output* (ASHRAE135, 2010b), próprio para iluminação.

6. *LSS-WG (Life Safety and Security)*: tal como o que ocorre com o EL-WG e o LA-WG, este grupo está buscando a especialização do BACnet para um uso específico. Atualmente, este grupo está trabalhando com propostas para o sistema de alarme de incêndios e controle de acesso.

7. *MS/TP-WG (Master-Slave/Token-Passing)*: devido a grande utilização do protocolo MS/TP foram encontrados muitos problemas na definição do BACnet para este tipo de rede e enlace de rede. Este grupo de trabalho busca a resolução dos problemas referentes ao protocolo MS/TP e o aperfeiçoamento da definição do meio físico mais utilizado pelo BACnet atualmente.

8. *NS-WG (Network Security)*: este grupo de trabalho está responsável pela melhora na segurança do BACnet, seja pela definição de formas de autenticação na rede e a transferência de controle de um dispositivo para outra. Em 1 de julho de 2010 foi publicado o Addendum 135-2008g (ASHRAE135, 2010a) que trata da segurança de rede.

9. *OS-WG (Objects and Services)*: este grupo altera a infra-estrutura do protocolo, adicionando ou retirando alguns objetos e serviços.

10. *SG-WG (Smart Grid)*: este grupo de trabalho está focado em fazer com que o prédio se comporte como uma parte de um sistema distribuído de processamento e atuação.

11. *TI-WG (Testing and Interoperability)*: este grupo de trabalho está responsável pela manutenção do padrão 135.1 de testes, além de identificar e sanar os problemas de interoperabilidade devido as falhas de especificação. Este grupo de trabalho afeta diretamente os BTL *BACnet Testing Labratories*(BTL, 2010).

12. *WN-WG (Wireless Networking)*: este grupo de trabalho está responsável pela definição do protocolo BACnet em redes sem fio. Atualmente está trabalhando na especificação do BACnet sobre Zigbee e sobre Ethernet sem fio.

13. *XML-WG (XML Applications)*: este grupo de trabalho está responsável pela investigação de aplicações do XML em sistemas BACnet.

Além da divisão citada anteriormente, a ASHRAE instituiu a figura dos BIGs (*BACnet Interest Groups*), os grupos de interesse do BACnet, formado por empresas e pesquisadores de uma dada região demográfica.

Atualmente os BIGs existentes são o BIG-NA (que abrange o Canadá, Estados Unidos e México), o BIG-EU (formado por vários países na Europa) e o BIG-RU (aplicado somente na Rússia), dentre outros.

b) Definição do protocolo BACnet: o BACnet está organizado em uma estrutura reduzida de 4 camadas (vide figura 2.12), com referência à pilha de referência OSI. Dentro do padrão BACnet estão definidos as camadas de *aplicativo* e de *rede*, sendo que a camada de rede é muito simplificada. Já nas camadas de *enlace de dados* e de *meio físico* existe uma série de meios definidos, sendo que este grupo está continuamente sendo expandido e alterado (ASHRAE135, 1995).

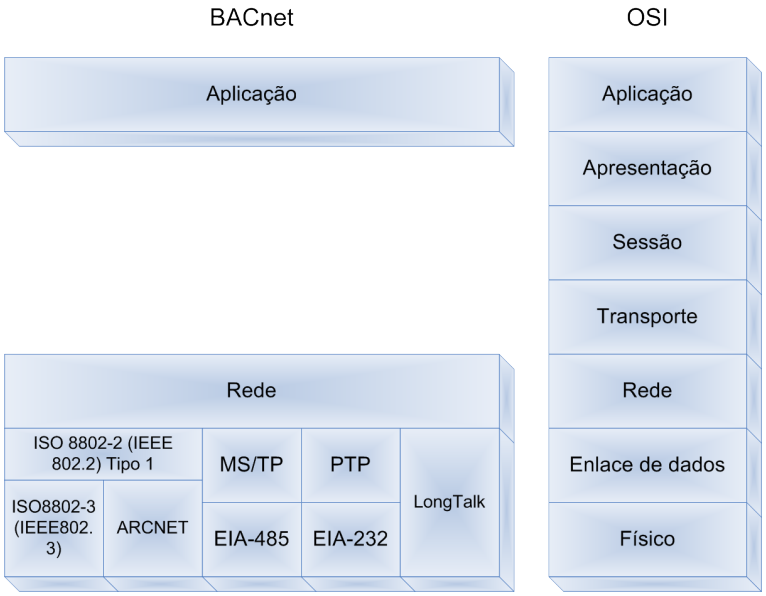

Figura 2.12: Comparativo das pilhas BACnet e OSI. Fonte: (ASHRAE135, 1995), modificado

O Ethernet pode ser implementado sobre cabo coaxial, par trançado ou fibra ótica. Atualmente, a definição BACnet Ethernet está caindo em desuso devido ao aumento da utilização do BACnet/IP. Já o MS/TP é utilizado sobre o EIA-485, que por sua vez se comunica através de par trançado, seja de 2, 3 ou 4 fios, sendo o meio mais popular, devido ao seu custo reduzido e ao grande número de projetos já realizados, bem como a sua capacidade de comunicação a uma distância de até 1200 metros sem repetidor (RS485, 1985).

O BACnet também pode ser executado sobre o ARCNET, podendo ser utilizado sobre cabo coaxial, par trançado ou fibra ótica. Este caso é raramente utilizado e não deve ser visto como uma boa escolha para projetos novos, assim como a versão PTP do BACnet que executa sobre EIA-232, diretamente ou sobre telefone (ASHRAE135, 1995).

Existe ainda o BACnet sobre o LonTalk como camada de enlace de dados, podendo vir a ser usado sobre qualquer mídia, assim como o BACnet sobre Zigbee que está sendo ainda definido. Alguns trabalhos propõem a utilização do BACnet sobre outras redes, como em (GRANZER; KASTNER, 2007), que propõem a definição do BACnet/KNX,

para aproveitar as redes Konnex existentes, bem como as vantagens das mesmas.

O BACnet define dois elementos principais, sendo eles os *objetos* e os *serviços*. O objeto é a representação de um determinado elemento físico ou um dado processo computadorizado. Por exemplo, se tem como exemplo de objetos as entradas analógicas, saídas digitais, dentre outros. Para exemplificar esta definição, se tem a figura 2.13, que ilustra um dispositivo BACnet fictício denominado Termo-001. Neste dispositivo, se tem um AO (*Ananlog Output*), um AV(*Analog Value*), um BV (*Binary Value*) e um objeto do tipo dispositivo (*Device*). O objeto AO, além de outras propriedades, possui uma propriedade do tipo *Prenset Value*, que no exemplo da figura 2.13 assume o valor de 21.0 e outra do tipo *Units*, com o valor de 62, que de acordo a definição do BACnet (ASHRAE135, 1995), significa graus Celsius (*degrees celsius*).

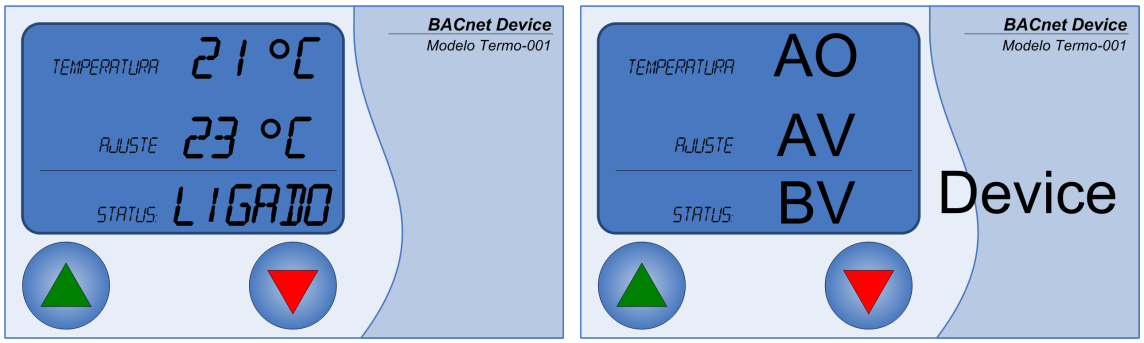

Figura 2.13: Exemplo de um dispositivo BACnet com os seus respectivos objetos e parâmetros

Por outro lado, os serviços no BACnet podem ser divididos nos seguintes grupos (ASHRAE135, 1995):

1. Alarme e eventos, com os serviços *GetAlarmSumary*, *GetEnrollmentSumary*, *SubscribeCOV*, dentre outros.
2. Acesso a arquivos, com os serviços *AtomicReadFile* e *AtomicWritedFile*.
3. Acesso a objetos, como por exemplo os serviços de *WriteProperty*, *ReadProperty*, dentre outros.
4. Manutenção remota de dispositivos, como por exemplo os serviços de *Who is*, *I am*, *Who has*, *I have*, *TimeSynchronization*, dentre outros.
5. Terminais virtuais, com os comandos *VT-Open*, *VT-Close*, *VT-Data*.

Por outro lado, o conjunto de serviços está sendo continuamente alterado devido os trabalhos dos *Working Groups* da ASHRAE.

c) Organização lógica de um dispositivo BACnet: um dispositivo para o BACnet nada mais é que um agrupamento de entidades chamadas *objetos*. Todo dispositivo possui necessariamente pelo menos um objeto do tipo *Device*, que por sua vez contém as informações básicas do dispositivo em seu conjunto de propriedades. Algumas propriedades de um objeto *Device* com os dados de um dispositivo fictício estão descritos na figura 2.14.

Propriedade	Valor	Domínio
Object_Identifier	(Device, Instance 3)	Listagem
Object_Name	"Controlador de Temperatura"	Texto
Object_Type	8(em 2^10 bits) [DEVICE]	Constante
System_Status	0 (em 1 octeto) [OPERATIONAL]	Constante
Vendor_Name	"UnB"	Texto
Vendor_Identifier	1001	Número
Model_Name	"Temp001"	Texto
Firmware_Revision	"1.0.1"	Texto
Application_Software_Version	"V1.0.1 - Março 07, 2010"	Texto
Location	"Quarto do Casal"	Texto
Description	"Controlador de Temperatura"	Texto
Protocol_Version	1	Número
Protocol_Conformance_Class	3	Número
Protocol_Services_Supported	B'11111111111111111111111111'	Mapa de Bits
Protocol_Object_Types_Suported	B'1111100000000000000'	Mapa de Bits
Object_List	((Analog Imput, Instance 1), (Analog Value, Instance 2), (Binary Value, Instace 1), (Device, Instance 3))	Listagem
Max_APDU_Length_Accepted	400	Número
Segmetation_Supported	3 (em 2 bits) [NO_SEGMETETION]	Constante

Figura 2.14: Exemplo de um objeto do tipo dispositivo (*device*)

Na figura 2.14, se tem na primeira coluna o tipo de propriedade que será informada, na segunda coluna o valor que deverá ser armazenado, na terceira coluna se tem o valor traduzido, quando for o caso de constantes do padrão (ASHRAE135, 1995), e na última, o domínio dos dados contidos na propriedade.

Na definição original do BACnet (ASHRAE135, 1995), o padrão de codificação de caracteres automático era o ASCII (ANSI X3.4); porém devido a grande utilização do BACnet como protocolo de automação em países que precisam de mais caracteres para expressar em sua língua os textos descritivos, foi adotado com o *Addendun k* (ASHRAE135, 2010c), na página 4, o padrão UTF-8 (ISO 10646), com a constante código hexadecimal *X'00*.

No BACnet, uma propriedade pode ser de algum dos seguintes tipos: (a) opcional, (b)

requerida, (c) somente leitura e (d) alterável. O que define se uma dada propriedade é opcional ou requerida é a própria norma do BACnet, que julga que sem aquela propriedade o objeto não teria as informações necessárias para que fosse completamente compreendido por outros dispositivos. Quanto a característica da propriedade de ser alterável ou de somente leitura, está ligada tanto à definição formal do objeto quanto pela interpretação do fabricante do dispositivo. Por exemplo, para um objeto do tipo *Binary Output* é natural que se possa alterar o parâmetro *valor atual* (*Present Value*), enquanto para um objeto do tipo *Binary Input* é natural que só se possa ler este valor. O mesmo não acontece com um objeto do tipo *Binary Value*, onde ele pode ser ou não alterável, dependendo da decisão do fabricante.

Os dispositivos geralmente são apenas servidores passivos que informam seus valores quando questionados pelo mecanismo de *poll* e *response* (vide figura 2.15 a). Contudo, se um dado evento durar menos que o período de varredura de rede, o sistema cliente não saberá que o evento ocorreu.

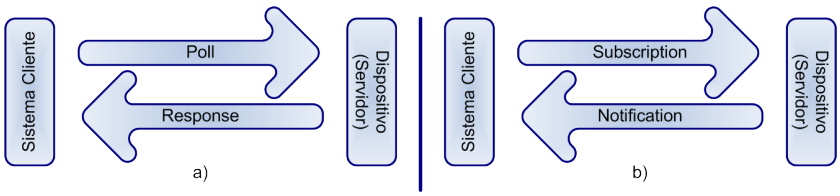

Figura 2.15: (a)Mecanismo de obtenção de valores e (b)por COV

Para resolver o problema dos eventos de curta duração, o BACnet define um sistema de notificação de alteração chamada COV (*change of value*). Neste mecanismo, um sistema cliente registra-se (*subscription*) no dispositivo servidor para receber a notificação da alteração de um dado parâmetro que deverá ser monitorado. O dispositivo servidor quando sofrer a alteração do valor notificará (*notification*) ao sistema cliente de que o valor alterou(vide figura 2.15 b).

O objeto do tipo dispositivo (*device*) possui um parâmetro informando se o dispositivo suporta ou não o COV. O BACnet COV é um subgrupo dos eventos de *alarm and event*. O registro é feito por serviços do grupo COV, como por exemplo *SubscribeCOV-Request* (para todo objeto), *SubscribeCOVProperty-Request* (para uma dada propriedade), sendo que a notificação é feita pelas mensagens *UnconfirmedCOVNotification-Request* e *ConfirmedCOVNotification-Request*.

Outro serviço de extrema utilidade e necessidade é o par de serviços *Who is* e *I am*, que é responsável pela montagem da rede BACnet. No caso do BACnet/IP, quando se têm

várias redes BACnet passando por diversos roteadores que não enviam mensagem de *broadcast*, foi desenvolvido a tecnologia BBMD (*BACnet/IP Broadcast Management Device*), onde um equipamento interligado as duas redes é capaz de retransmitir a mensagem de uma rede para outra.

Os serviços são mensagens trocadas entre os dispositivos para realização de alguma ação. Todo dispositivo tem que implementar pelo menos o serviço de *ReadProperty* para que tenha alguma funcionalidade (ASHRAE135, 1995). Com este serviço, consegue-se saber qualquer informação de um dispositivo previamente conhecido. Contudo, é interessante considerar como implementação mínima o serviço *I Am*, para que não seja necessário a inserção manual dos dispositivos da rede ou a pesquisa em todo a faixa de valores de dispositivos definidos pela rede (2 octetos).

Os serviços podem ser divididos em dois grupos:

1. *Com confirmação:* são mensagens *unicast* ou de reconhecimento. Essas mensagens requerem uma resposta do dispositivo questionado. A seguir seguem os principais comandos com confirmação definidos pelo protocolo BACnet com um descritivo resumido de seu significado.

   (a) *ReadProperty:* lê uma dada propriedade de um dado dispositivo. O retorno é o pacote contendo o valor da propriedade lida.

   (b) *WriteProperty:* escreve um dado valor em um dado parâmetro de um dado dispositivo. Pode retornar um erro caso o objeto não possua a propriedade ou se esta propriedade é apenas leitura.

   (c) *Operações de terminal como por exemplo:* a)*VTOpen:* abre uma conexão com um dispositivo com capacidade de comunicação por terminal, b)*VTClose:* fecha uma comunicação de terminal previamente aberta, c)*VTData:* envia os dados pela conexão estabelecida.

   (d) *ReinitializeDevice:* reinicia um dispositivo. Esta funcionalidade não é obrigatória de ser implementada pelo dispositivo dentro da definição do BACnet, portanto pode ser retornado um retorno de erro caso o tratamento da mensagem não seja suportada.

   (e) *CreateObject* e *DeleteObject:* cria ou remove um dado objeto em um dispositivo. Esta funcionalidade é um tanto contraditória, visto que representaria a criação de uma funcionalidade em um dispositivo, como por exemplo, criar

um *Analog Output*, sem nenhum *hardware* para expressar externamente esta variável a ser tratada.

2. *Sem confirmação:* geralmente são mensagens *broadcast*. Os principais comandos são os seguintes:

   (a) *I-Am:* envia uma mensagem falando que este dispositivo se encontra na rede, em um dados endereço. Geralmente é uma mensagem em resposta a mensagem *Who-Is*.

   (b) *I-Have:* envia uma mensagem informando que o dado dispositivo possui um dado objeto. Geralmente é uma mensagem em resposta a mensagem *Who-Have*.

   (c) *Who is:* envia uma mensagem pesquisando quais são os dispositivos que se encontram na rede. Geralmente é uma mensagem em que se espera como resposta a mensagem *I-Am*. Esta pergunta geralmente é feita no início da operação para a identificação da rede pelos controladores e periodicamente para a manutenção dos dispositivos interligados.

   (d) *Who Have:* envia uma mensagem pesquisando quais são os dispositivos que se encontram na rede. Geralmente é uma mensagem em que se espera como resposta a mensagem *I-Have*.

   (e) *Event Notification:* envia uma mensagem informando a ocorrência de um dado evento o dispositivo. Geralmente enviado pelos dispositivos para os controladores da rede.

   (f) *Time Synchronization:* envia uma mensagem informando a data e hora atual de um dado dispositivo. Geralmente enviado pelos controladores ou pelo um relógio central para os dispositivos da rede.

d) Estrutura de formação de um pacote BACnet: um pacote BACnet é formado por dois tipos de PDU (*Protocol Data Unit* Unidade de Informação do Protocolo): (a) um tipo concentrado na camada de aplicação chamado APDU (*Application PDU*) e (b) outro tipo concentrado na camada de rede NPDU (*Network PDU*) (vide figura 2.16). Um pacote BACnet também é composto por um cabeçalho (*Start Bits*) e um dígito verificador (*Check Sum*).

O pacote APDU contém as informações em mais alto nível a respeito dos dispositivos e serviços BACnet e este, por sua vez, está contido dentro do pacote NPDU que contém

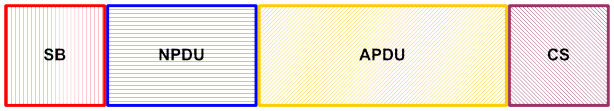

Figura 2.16: Estrutura de pacotes BACnet

algumas informações de rede. O NPDU está contido dentro do pacote da camada de enlace de dados e do pacote de camada física que serve de transporte para os pacotes BACnet, como por exemplo MSTP e o IP. Com esta estrutura, um dispositivo ao receber um pacote em cada camada fica responsável por "desempacotar" a porção do pacote enviando referente a sua camada, passando para a camada superior o restante do pacote.

e) Topologia de uma rede BACnet: uma rede BACnet é composta de vários dispositivos além de roteadores e *gateways*(vide figura 2.17). Os roteadores são responsáveis pela interligação de duas redes que conversam BACnet, seja com meios físicos diferentes ou não. Por exemplo, um roteador BACnet pode interligar uma rede BACnet MS/TP a uma rede BACnet/IP. Um *gateway* já é um dispositivo que realiza uma conversão de pacotes. Por exemplo, um *gateway* BACnet interliga uma rede BACnet/IP a uma rede Konnex.

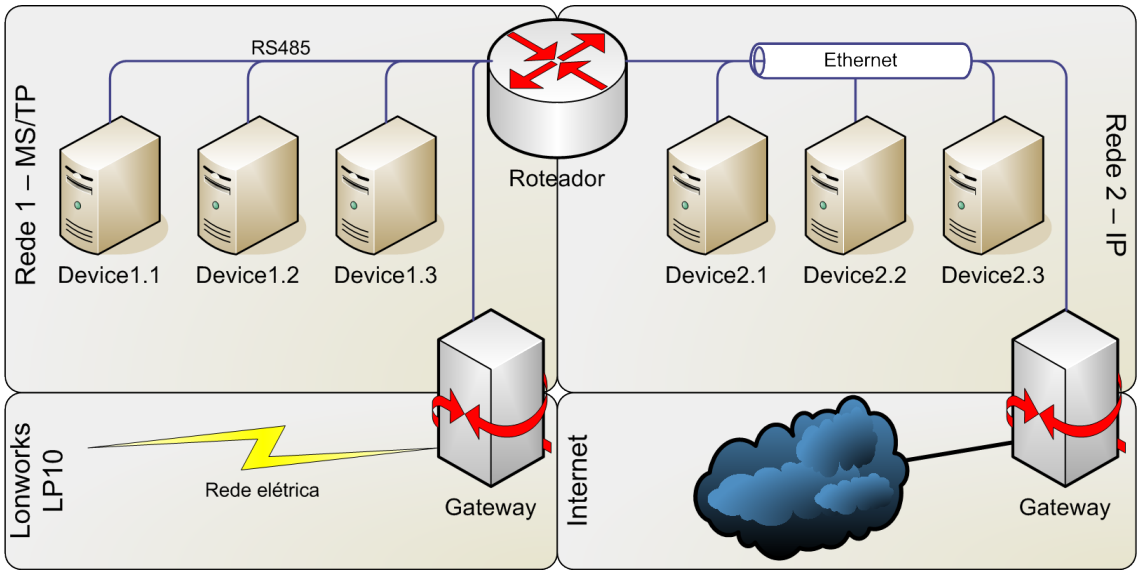

Figura 2.17: Elementos de rede BACnet

Na figura 2.17 é apresentado um exemplo de roteador BACnet interligando uma rede BACnet/MSTP (Rede 1 - tendo como meio físico um barramento RS485) a uma rede BACnet/IP (Rede 2 - tendo como meio físico um cabo Ethernet). Na mesma figura também é mostrado um *gateway* BACnet interligando a Rede 1 a uma rede LonWorks

LP10 (que utiliza a rede elétrica para mandar seus pacotes). Por fim, se tem a utilização de um *gateway* BACnet interligando a Rede 2 a Internet.

Um roteador BACnet pode vir a alterar apenas a informação específica do NPDU (vide figura 2.18 a), sem alterar a informação contida no APDU, que contém a informação em si. Cabe ressaltar que após esta alteração faz-se necessário fazer novamente o cálculo do dígito verificador.

Já um *gateway* BACnet altera completamente o pacote, alterando, removendo ou acrescentando informação de todo o pacote, perdendo qualquer referência com o pacote BACnet original (vide figura 2.18 b).

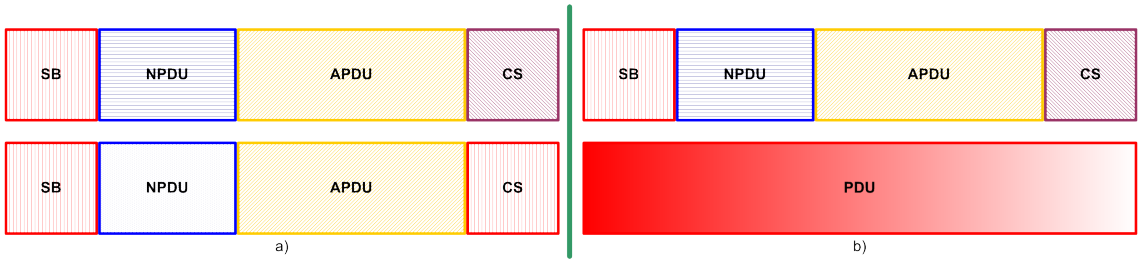

Figura 2.18: Estrutura de pacotes em (a) um roteador BACnet e (b) um *gateway* BACnet

2.3.2 Protocolos Proprietários de automação residencial e predial

São considerados protocolos de automação proprietários aqueles em que sua especificação não está aberta ao público, nem mesmo por meio de uma unidade certificadora, onde os desenvolvedores só podem utilizar os dispositivos comercializados por uma única empresa ou grupo de empresas, detentoras do conhecimento e da manutenção do protocolo.

Nesta dissertação, será feita uma breve referencia a alguns protocolos, considerados mais relevantes pelos seguintes motivos: (a) Metasys da Johnson Controls (METASYS, 2010) que é um protocolo líder de mercado na automação predial, (b) BUS da Btcino (BTICINO, 2010) que tem uma grande representatividade no mercado nacional e (c) outros protocolos proprietários representado as soluções mais pontuais.

A seguir serão descritos alguns exemplos de protocolos utilizados em automação predial, como o *Metasys* da *Johnson Controls* e o *BUS* da *Bticino* que são de grande

porte. Também será comentado a solução da *WP Inovações Tecnológicas*, no grupo das soluções de pequeno porte ou de âmbito local.

2.3.2.1 Metasys

Metasys é um protocolo da Johnson Controls (vide figura 2.19) (METASYS, 2010) voltado para a automação total de prédios, controlando a demanda de energia e luz, além de integrar com os sistemas de segurança e de incêndio. Atualmente, a empresa tornou público um subgrupo de funções do protocolo chamado N2Open, onde é capaz de se comunicar com o protocolo Metasys N2, mostrando um interesse comercial da empresa em atender o mercado interessado por soluções abertas. A Johnson Controls é uma das empresas que apóiam o protocolo BACnet.

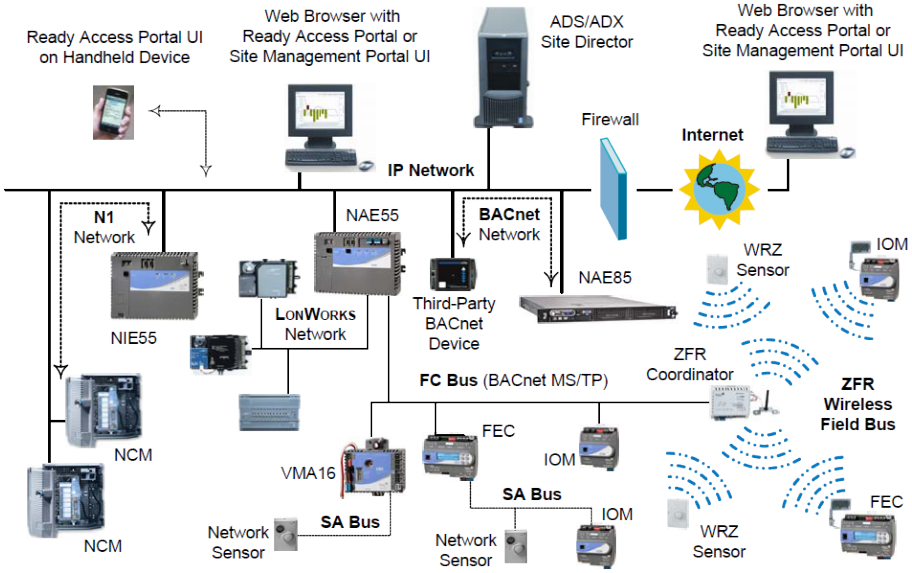

Figura 2.19: Metasys System Network Integration Fonte: (METASYS, 2010)

2.3.2.2 Btcino BUS

Btcino é uma empresa italiana (BTICINO, 2010) com sede em Milão (Itália) que desenvolve produtos para automação residencial com *design* avançado (vide figura 2.20) e que atua no mercado brasileiro. Seus produtos se comunicam em um protocolo proprietário e fechado chamado de BUS, que não se comunica apenas com outros produtos da própria Btcino. O mesmo possui uma vasta gama de produtos, como atuadores, painéis, quadros elétricos, dentre outros.

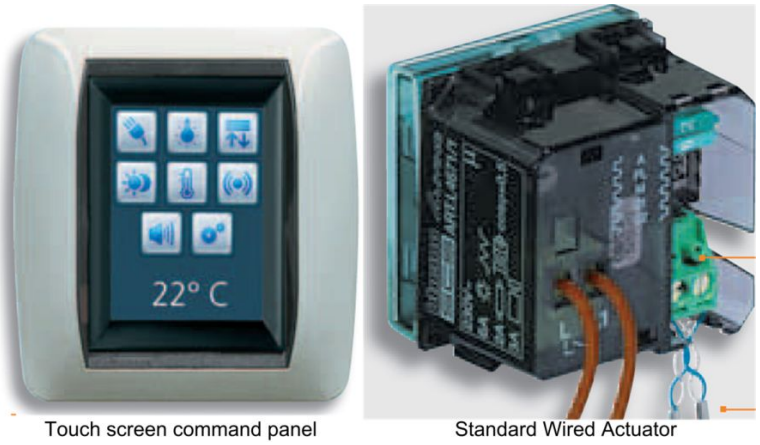

Figura 2.20: Produtos Bticino. Fonte: (BTICINO, 2010)- Catálogo 2010, modificado

2.3.2.3 As soluções de automação de pequeno porte ou de âmbito local

Existe ainda uma série de soluções de automação que geralmente utiliza soluções proprietárias de automação, onde os protocolos são desenvolvidos por um grupo reduzido de engenheiros ou até mesmo técnicos. Contudo, com a popularização e a facilidade de compreensão dos protocolos abertos, em especial o BACnet, essas soluções vem perdendo mercado devido a possibilidade de utilização de dispositivos de automação de diversos fabricantes nos protocolos abertos (como por exemplo o BACnet).

Um exemplo deste seguimento da automação residencial é o protocolo utilizado pela WP Inovações Tecnológicas (WPAUTOMACAO, 2010), uma empresa que atua no Distrito Federal do Brasil, cujo seu protocolo foi utilizado para automação de segurança do laboratório do GRACO através do projeto SIAP (Sistema Integrado de Automação Predial) (PIRES; GONÇALVES, 2006).

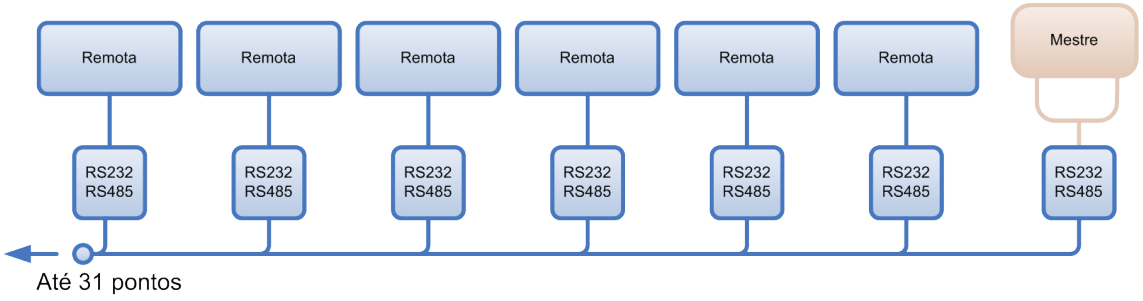

Figura 2.21: Estrutura de uma rede da WP/SIAP fonte: (PIRES; GONÇALVES, 2006), modificado

O protocolo SIAP é voltado para a comunicação de unidades chamadas *remotas* (vide figura 2.21) e uma estrutura chamada *mestre*. As remotas são unidades microprocessadas que integram atuadores e sensores na rede SIAP. Como o barramento padrão da

rede é o RS485 e a maioria dos microcontroladores possuem interface RS232, faz-se necessário a utilização de um conversor RS232 para RS485 (vide figura 2.21).

Segundo o protocolo WP/SIAP podem existir até 31 remotas, com a faixa de endereços indo de 1 a 31, sendo o endereço 0 o endereço de *broadcast* (PIRES; GONÇALVES, 2006). Esta característica deriva do fato de se utilizar 5 bits para o endereçamento das remotas. O protocolo WP/SIAP é um protocolo de um pacote de único formato e de tamanho fixo de mensagens, indicado na figura 2.22. Este protocolo garante a qualidade da mensagem com um *checksum* do tamanho de um octeto (vide figura 2.22), um *checksum* relativamente grande um pacote com 15 octetos totais (vide figura 2.22).

Dados	Tamanho	Descrição
Endereço	5 bits	Endereçamento das remotas
Código	3 bits	Código do comando de comunicação
Saída	8 bits	Apresenta a última saída da remota
Entrada	8 bits	Apresenta o estado das entradas de uma remota
Comando	8 bits	Envia um comando para uma remota
Acionamento	8 bits	Envia um acionamento para uma remota
Dado	8 bits	Envia um dado para uma remota
Leitura	64 bits	Apresenta uma leitura de uma remota
Checksum	8 bits	Verifica a integridade dos dados transmitidos na rede

Figura 2.22: Estrutura de um pacote no protocolo WP/SIAP fonte: (PIRES; GONÇALVES, 2006), modificado

2.4 MEIOS FÍSICOS E ENLACE DE DADOS

Meios físicos são os elementos de rede que se encontram representados na camada 1 da pilha de referência OSI (vide figura 2.23). Os meios físicos representam por onde e como as informações irão trafegar, através dos diversos dispositivos interligados na rede de automação. A análise dos diferentes meios físicos se faz necessária em um projeto de automação predial devido a dispersão geográfica dos elementos de automação, que geralmente se encontram na ordem de dezenas de metros. Outro fato que valida o estudo dos meios físicos é a existência de projetos de automação tanto em edificações novas quanto velhas, sendo que nestas últimas não se tem uma planta de automação instalada, pois o projeto inicial não previa esta operação. A unidade básica da camada 1 é o bit (TANENBAUM, 2003).

O enlace de dados refere-se a camada 2 na pilha de referência OSI (vide figura 2.23) que tem como principal tarefa a transformação do canal de transmissão bruta que é

dado pela camada física (TANENBAUM, 2003), em uma linha que pareça livre de erros de transmissão não detectados pela camada de rede. Outra finalidade importante desta camada é o controle do acesso ao meio, evitando que um transmissor rápido envie uma quantidade excessiva de dados a um receptor lento, aplicando algum mecanismo que regule o tráfego. A unidade básica da camada 2 é o *frame* (TANENBAUM, 2003).

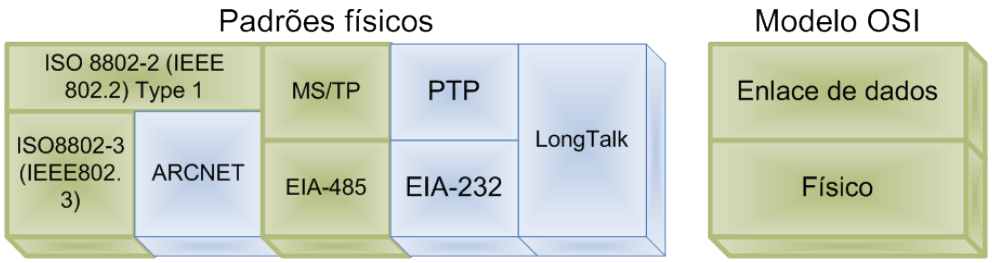

Figura 2.23: Comparativo dos meios físicos com a pilha de referência OSI

2.4.1 RS485

O RS485 é um protocolo de enlace físico baseado em par trançado e transmissão de dados de forma diferencial, ou seja, o bit é representado pela DDP (*Diferença De Potencial*) entre as duas linhas do par, normalmente chamada de linha A e linha B. Este protocolo é muito utilizado em automações prediais, devido ao seu custo relativamente baixo e a alta gama de produtos que atendem a esta especificação no mercado

Dentre as vantagens do RS485 cabe ressaltar: (a) grande robustez a ruídos; esta é uma característica inerente aos protocolos seriais de par diferencial e (b) boa taxa de transmissão de dados a uma distância de até 1.2Km (RS485, 1985). Dentre as desvantagens deste protocolo se tem: (a) queda da taxa de transmissão quando se tem um barramento muito grande e (b) necessidade da passagem de um barramento por dentro do prédio, muitas vezes levando a quebras de paredes e revestimentos, acrescentando ao custo de instalação o valor inerente a este procedimento.

O padrão RS485 é uma das formas de meio físico definidos na norma do BACnet devido principalmente a dois grandes motivos: (a) grande popularidade do padrão RS484 na automação predial ao longo dos anos e (b) suas características inerentes de distância e imunidade a ruídos. Este padrão geralmente é implementado em conjunto com o protocolo MS/TP. Como exemplo de protocolo que se utiliza do padrão RS485 se tem o WP (PIRES; GONÇALVES, 2006).

2.4.2 MS/TP

O MS/TP (*Master Slave/Token Passing*), é a utilização conjunta de dois padrões que é o Mestre Escravo *(Master Slave)* e o Passagem de *Token*(*Token Passing*). No BACnet, cada dispositivo é definido como um recurso mestre pelo momento que este é detentor do *Token*, passando a ser um escravo nas outras condições.

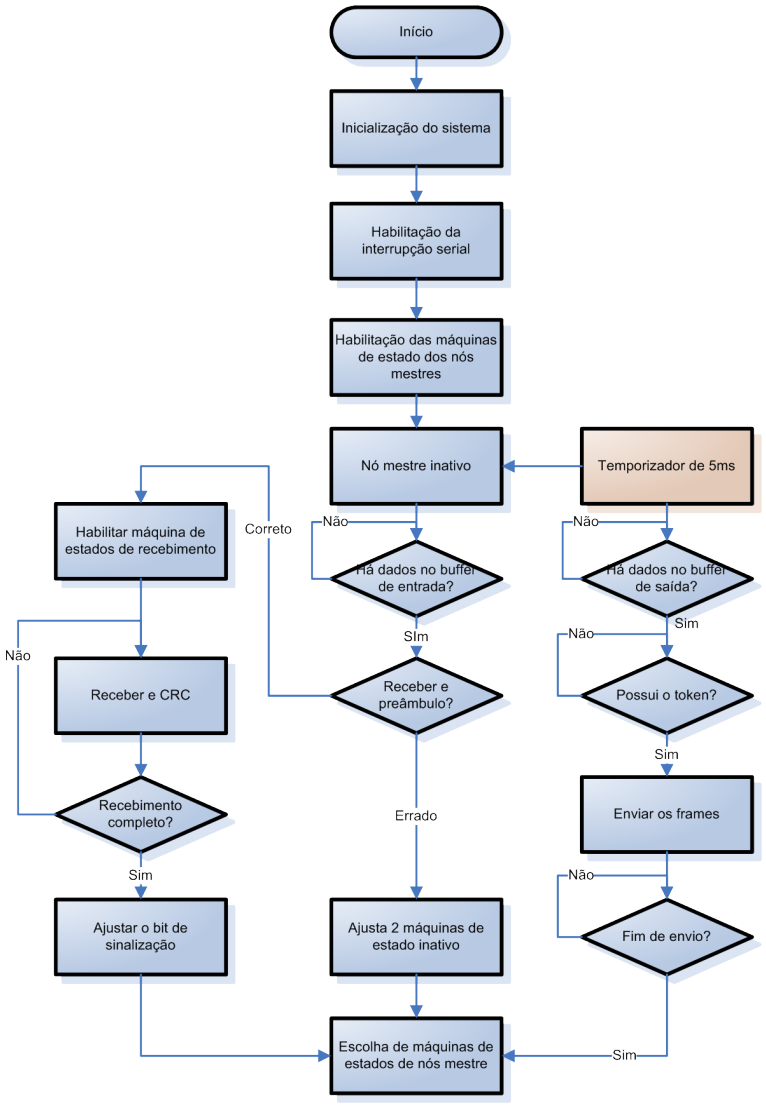

Figura 2.24: Diagrama de um aplicativo MS/TP para o protocolo MSTP. Fonte (LIU; REN, 2007) - modificado

Os trabalhos de Liu et al (LIU; REN, 2007) apresentam uma forma de implementar o protocolo MS/TP para ambientes embarcados, tendo como alvo final o padrão BACnet MS/TP. Na implementação proposta, o sistema se inicializa realizando seus procedimentos primários, como por exemplo habilitar a interrupção de porta serial (vide figura 2.24).

Um ponto importante na abordagem adotada é a alta necessidade de temporização do sistema, indicada pelo temporizador de 5ms. Os mesmos autores fazem uma análise matemática interessante das características de tempo real do sistema de acordo com o número máximo de *frames*, o tempo entre *frames* e o tempo total. Nesta análise é mostrando que estes parâmetros têm que ser bem dimensionados para evitar um aumento no atraso de serviço, que pode chegar a ser superior ao tempo do *token*.

$$G = \frac{1}{B} \sum_{i=1}^{N} \frac{L_i}{T_i} \tag{2.1}$$

O comportamento do sistema é modelado pela equação 2.1, onde G é a carga de rede (variando de 0 para bom e 1 para ruim), B é a taxa de transmissão, N é o número de nodos que geram mensagens na rede, T é o tempo médio de geração de mensagens e L é o tamanho médio da mensagem.

Pela Eq. 2.1, pode-se deduzir que cada vez que se introduzir dispositivos mestres (geradores de mensagens a cada recebimento de *token*) na rede, pior será o desempenho total da rede. Este cenário pode ser melhorado colocando cada vez mais dispositivos escravos, que são aqueles que não geram mensagens, mas sim respondem a elas.

Outro ponto importante é a manutenção em memória dos dados em memória até o recebimento completo da informação integra, garantida pela conferência do CRC.

2.4.3 O protocolo Ethernet

O Ethernet é um tipo de LAN (*Local Area Network* Rede de Área Local) padronizada sobre o número IEEE 802.3. Este tipo de rede teve seu início na Universidade do Havaí, servindo de inspiração aos pesquisadores Bob Metcalfe e David Boggs, que projetaram a Ethernet para interligarem (vide figura 2.25) os computadores pessoais (TANENBAUM, 2003). Sua maior característica é a existência de um barramento único com controle de fluxo por detecção de colisão e geração de um tempo de retardo aleatório e exponencial.

Tanenbaum (TANENBAUM, 2003) analisa no capítulo 4.3.5 o desempenho das redes Ethernet, focando principalmente no algoritmo de recuo binário exponencial, verificando que a Ethernet não é o melhor padrão para a situação de grande largura de

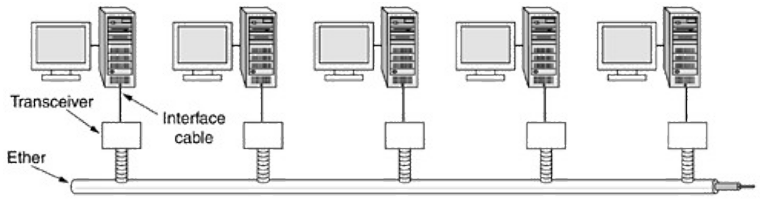

Figura 2.25: Arquitetura da Ethernet original. Fonte:(TANENBAUM, 2003)

banda e longas distâncias. Contudo, esta configuração de utilização não é a que se costuma encontrar, pois a faixa de largura de banda e as faixas de distância envolvidas na automação predial são muito menores que as esperadas na comunicação de computadores.

Outro ponto levantado é que a Ethernet é mais eficiente com *frames* com um maior número de bits. Para o caso da automação domótica as características mais impeditivas são: (a) o tamanho do cabo sem retransmissor e (b) o interesse por *frames* maiores, pois geralmente os comandos possuem poucas palavras. Contudo, o maior impeditivo para automação residencial é o custo associado à instalação de uma rede Ethernet, tanto contando o preço da infra-estrutura (cabos repetidores, roteadores, ...), quanto o preço elevado dos dispositivos de automação, quando comparados a implementações usando RS485.

Por outro lado, cabe citar entre as vantagens do Ethernet: (a) sua grande aceitação de mercado e (b) o grande número de dispositivos atualmente interligados neste tipo de rede, tornando-o um padrão de fato.

Na definição do padrão BACnet estão apresentados dois padrões que envolvem o Ethernet, o BACnet/Ethernet e o BACnet/IP. Estes padrões são de mais fácil implementação em computadores devido à qualidade das redes domésticas e ao alto grau de abstração fornecido pelos sistemas operacionais a estes padrões de comunicação. Sua implantação em dispositivos embarcados está progredindo devido ao surgimento de diversos SoCs com conexão Ethernet, mas ainda seu custo é elevado para o desenvolvimento em massa.

2.4.4 Zigbee

Zigbee é um padrão de transmissão de informação via radiodifusão voltado para redes de pacotes pequenos e com baixa taxa de transmissão. Seu nome vem do modelo

de ligação, muito semelhante o zig-zag que as abelhas (em inglês *bee*) realizam nas colméias.

Dentre as vantagens da utilização do Zigbee cabe citar: (a) o grande alcance e vencimento de barreiras quando se está na configuração *Mesh*, a um custo energético aceitável e a baixa potência de cada transmissor e (b) o baixo consumo energético para qual o padrão foi desenvolvido, particularmente útil para dispositivos portáteis que participam da automação doméstica, como *SmartPhones*, controladores portáteis, dentre outros. Dentre as desvantagens se tem: (a) alto custo associado ao produto, visto sua baixa escala de produção e ao seu relativamente novo desenvolvimento e (b) o problema com a questões de qualidade de implementação dos transceptores.

O padrão do BACnet para redes ZigBee está sendo definido no apêndice J da norma ASHRAE135 (ASHRAE135, 1995). Não existem no presente momento muitos trabalhos que se utilizem efetivamente desta definição, mas sim implementam o MSTP sobre o Zigbee.

2.5 INTRODUÇÃO À TV DIGITAL

A seguir será apresentado uma breve introdução sobre TV, em especial TV digital, a fim de contextualizar esta dissertação nas áreas de redes, automação residencial e TV digital.

2.5.1 Histórico

O histórico da evolução da televisão pode ser dividido em algumas dimensões, sendo que as adotadas nesta dissertação são as seguintes:

1. *Evolução por cor*: onde se tem iniciando com as primeiras televisões eletro-mecânicas, depois os televisores em preto e banco e por fim, a transmissão a cor.
2. *Evolução por componentes eletrônico*: televisores eletro-mecânicos, valvulados, mistas (válvula e transistor), transistorizadas e com circuitos integrados, sendo esta última dividida em CIs de baixa integração e CIs de alta integração.

3. *Evolução por tipos de telas*: CRT (*Cathode Ray Tube* tubo de raios catódicos), plasma, LCD (*Liquid Cristal Display* tela de cristal líquido) e LED (*Light Emiter Diode* diodo emissor de luz).

4. *Divisão por padrões de cor*: NTSC, PAL-M/N, Secan. A figura 2.26 mostra a utilização de cada um dos padrões no final do século XX.

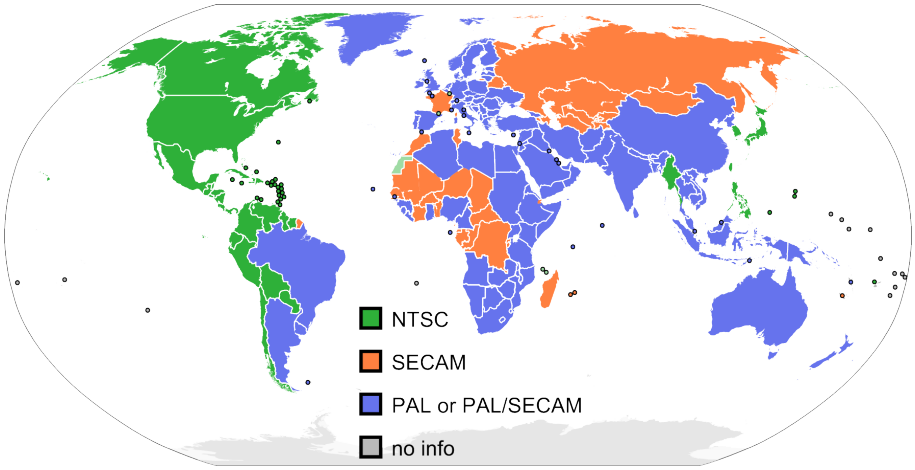

Figura 2.26: Utilização dos padrões de cor de TV no final do século 20. Autor: Akomor1, Henriok, GioMac. domínio público

2.5.2 Padrões de TV digital

Os padrões de TV digital definem os mecanismos de transmissão, recepção, configurações mínimas de *hardware*, faixas de frequência de operação, padrões de transporte de dados e compactação dentre outros pontos mínimos para garantir a interoperabilidade entre os equipamentos de TV digital comercializados (OLIVEIRA et al., 2008a). Na figura 2.27 se tem a representação das principais partes de um padrão de TV digital, com as soluções adotadas pelos principais padrões de TV digital da atualidade. As partes mais importantes da TV digital podem ser definidas como:

- *Transmissão:* É a parte do padrão de TV digital que define como os 0s e 1s serão transmitidos no meio físico. Outros exemplos de padrões de transmissão, não ligados a TV digital, são o AM, FM e o GSMK.

- *Transporte:* É a parte do padrão de TV digital que define um pacote de informação. As camadas de *transporte* e de *codificação* definem juntas o formato de vídeo adotado. Esta divisão ocorre também nos padrões de vídeo digital atualmente utilizadas como, por exemplo, os diversos formatos de vídeo de computador.

- *Codificação:* É como um quadro (em inglês *frame*) está representado, ou seja, codificado. Este quadro pode ter alguma relação com os adjacentes para promover uma maior compactação.

- *Middleware:* É uma camada de software que provê abstração do hardware disponível para a TV digital, padronizando o acesso a este *hardware*. Por estar entre o *hardware* e os *softwares* aplicativos, foi chamado de "meio"(*middle* em inglês).

- *Aplicações:* São aplicações de TV digital que executam sobre os *middlewares* e que provém algum recurso rico para o usuário final, como menu interativo, guia de programação, sistema de vendas pela TV e automação residencial.

Atualmente, todos os padrões de TV digital se utilizam do *MPEG2-Sistemas* como protocolo de transporte (vide figura 2.27 camada de transporte), o que facilita a criação de *hardwares* genéricos nesta camada. Outro ponto é que todos podem ser transmitidos por QAM (vide figura 2.27 camada de transmissão). A seguir serão apresentados os principais padrões de TV digital, com as soluções adotadas em suas diversas camadas, a fim de ilustração.

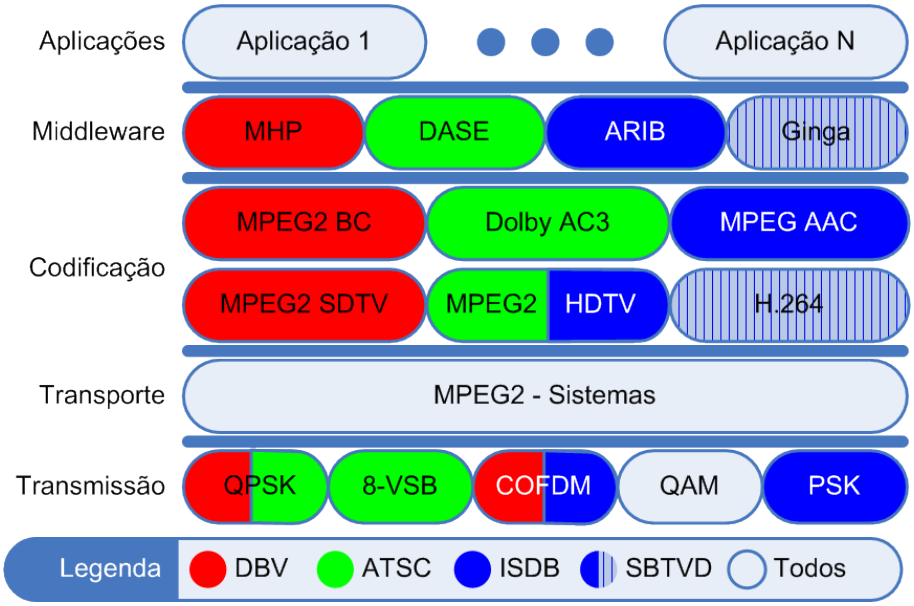

Figura 2.27: Arquitetura dos diferentes sistemas de TV digital

2.5.2.1 Europeu = DVB

DVB (*Digital Vídeo Broadcasting*) é um padrão desenvolvido pelo consórcio europeu *DVB Project*, para transmissão de televisão digital e é o padrão adotado pela maioria

dos países no mundo. O DVB se utiliza do *middleware* MHP, e a codificação de áudio, vídeo e dados se dá no padrão *MPEG2 BC* (THOM; PURNHAGEN; SUBGROUP, 1998) e *MPEG-2 SDTV* (CTPIM, 2004).

2.5.2.2 Americano = ATSC

O ATSC (*Advanced Television System Committee*) é um padrão desenvolvido a partir de 1987 e que entrou em operação em 1998 nos Estados Unidos (ATSC, 2010). O ATSC se utiliza do *middleware* DASE, e a codificação de vídeo e dados se dá no padrão *Dolby AC3* e MPEG-2 HDTV.

2.5.2.3 Japonês = ISDB

ISDB (*Integrated Services Digital Broadcasting*) é um padrão desenvolvido no Japão pelo consórcio DIBEG (*Digital Broadcasting Experts Group*), sendo a emissora japonesa NHK a participante mais ativa. O ISDB se utiliza do *middleware* ARIB, e a codificação de vídeo e dados se dá no padrão *MPEG2 AAC* e MPEG-2 HDTV. Sua transmissão pode ocorrer em COFDM, QAM e PSK

2.5.2.4 Brasileiro = SBTVD

O SBTVD (Sistema Brasileiro de TV Digital) é praticamente igual ao ARIB, só diferenciando o *middleware*, no qual se tem o Ginga para o SBTVD, além de não exigir suporte para o JavaTV, que tem cobrança de *royaltes*. Outra diferença importante é a adoção do MPEG 4 para a Codificação do audio e do vídeo (H.264). A grande vantagem nesta abordagem é a facilidade de obtenção de *hardwares* receptores.

2.5.3 Meios de transmissão

Os meios físicos de transmissão são uma característica importante das transmissões em *broadcast*, pois estes delimitam como e quem poderá acessar a informação. Para

a transmissão de TV digital existem os seguintes meios de físicos de transmissão: (a) satélite, (b) cabo, (c) terrestre e (d) IPTV.

O meio terrestre é aquele se utiliza de torres de transmissão por parte do emissor, sendo o meio de transmissão o ar, onde os dados são transmitidos na faixa de VHF ou UHF. No Brasil, a faixa adotada para a transmissão de TV digital foi a faixa de UHF. O receptor de TV digital é equipado com uma antena e é geralmente chamado de STB *set-top box*.

2.5.4 Padrão de codificação de vídeo

Para a transmissão de um conteúdo digital de áudio e vídeo se faz necessário a codificação da informação em um padrão, a fim de que se possa obter um melhor aproveitamento de banda, a confecção de aparelhos com interoperabilidade entre fabricantes dentre outros. Na transmissão de vídeo para TV digital dois padrões foram adotados pelo governo brasileiro (15606-1, 2008) o MPEG2 e o MPEG4.

2.5.5 *Middlewares* de TV digital

Middlewares são componentes de *software* embarcados em receptores de TV digital a fim de prover uma infra-estrutura básica necessária a interatividade, além de prover acesso a funções básicas de utilização do aparelho, como mudança de canais, ajuste de volume, EPG, dentre outros. A priori, um *middleware*(vide figura 2.27) deveria estar no nível de sistema operacional, porém os desenvolvimentos atuais estão tendendo a colocá-los no nível de API (*Application Programming Interface*) (MHP, 2010). Basicamente, cada padrão de TV digital criou seu próprio *middleware*, e esta divisão segue algo semelhante ao que ocorreu com os padrões de cor. Os *middlewares* dos respectivos padrões de TV digital são os seguintes: (a) DASE (*DTV Application Software Enviornment*) que vem a ser o *middleware* do padrão norte-americano, (b) ARIB (*Association of Radio Industries and Business*) que vem a ser o *middleware* do padrão japonês, (c) MHP (*Multimedia Home Platform*) sendo um *middleware* que permite execução de aplicações desenvolvidas em Java. Foi projetado para padrão europeu DVB (MHP, 2010). Sua especificação é evoluída e se encontra atualmente na versão 1.2. Basicamente, ele possui uma JVM com algumas restrições de segurança como, por exemplo, o

acesso a arquivos e programas do SO, acrescentando algumas funcionalidades específicas de TV digital, dentre elas o JavaTV, HAVi, e DAVIC e o (c) Ginga (sem significado específico) é um *middleware* aberto para o projeto do Sistema Brasileiro de TV Digital (SBTVD) (GINGA, 2010). Possui dois paradigmas de programação, um procedural, em que se utiliza Java, e outro declarativo que se utiliza NCL *Nested Context Language*. O Ginga ainda se encontra em desenvolvimento e não é adotado em nenhum *set-top box* devido as suas grandes necessidades de *hardware* e problemas técnicos.

2.6 A AUTOMAÇÃO RESIDENCIAL E A TV DIGITAL

Alguns trabalhos versam sobre a interligação da TV digital com a automação residencial. Estes trabalhos abordam o assunto sobre diferentes perspectivas, o que demonstra o puerismo do assunto no âmbito acadêmico, sendo este um forte indicativo de que esta é uma área onde podem ser gerados trabalhos inovadores.

Alguns trabalhos buscam desenvolver formas alternativas de integração, sem a utilização dos *middlewares* de TV digital, fundamentando a sua decisão principalmente no fato do alto custo computacional agregado aos *middlewares*, geralmente baseados em uma JVM (*Java Virtual Machine*) completa, como é o caso do MHP e do Ginga. Esse fator causa uma baixa aderência dos *middlewares* no mercado.

Tal abordagem é adotada por (UMBERGER, 2008), que desenvolve seu próprio *gateway Konnex/Webservice*. No artigo apresentado por (UMBERGER, 2008) é desenvolvido uma solução de integração de IPTV e automação residencial é baseada em uma rede Konnex. Esta solução foi desenvolvida em linguagem VisualBasic (CENTER, 2010) e sobre uma arquitetura PC (vide figura 2.28), rodando sobre um sistema operacional WindowsXP.

Na figura 2.28 é visível a estrutura adotada no projeto, separando claramente um *gateway* Konnex para um padrão de caracteres simples baseados no padrão RS232 como meio físico, e transportando caracteres no padrão ASCII (vide figura 2.29), onde cada caractere mapeia diretamente um comando para a rede, sem a definição de um protocolo mais sofisticado. Esta parte está visível na placa a direita da figura 2.28. No lado esquerdo se tem uma placa mãe de um PC pequena. Também se pode observar uma placa na parte inferior a direita de uma etapa de potência. Provavelmente, para

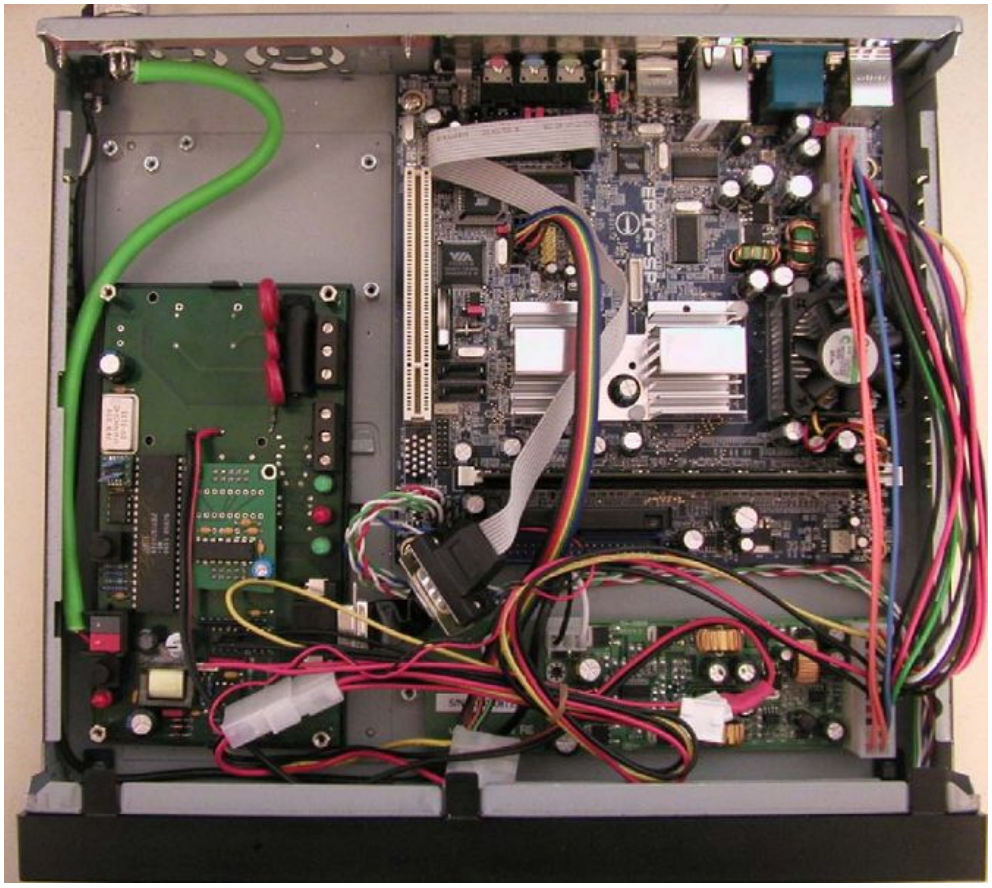

Figura 2.28: *hardware* desenvolvido para ser o *gateway* Konnex WebService. Fonte (UMBERGER, 2008)

esta foto foram retirados o HDD (*Hard Disk Driver* Disco rígido), *drivers* e outros para facilitar a tomada da foto.

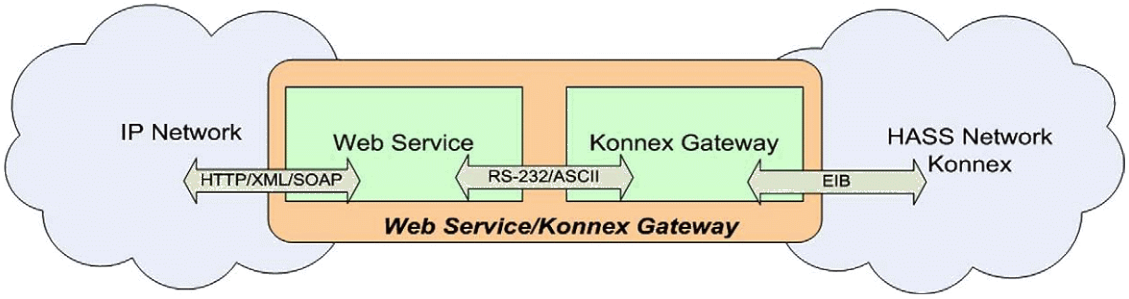

Figura 2.29: Arquitetura do *hardware* desenvolvido para ser o *gateway* Konnex WebService. Fonte (UMBERGER, 2008)

Dentro do PC encontra-se o desenvolvimento de um *web Service* que se liga ao *gateway Konnex* através de uma porta serial, sendo transmitidos os comandos pré-definidos simplistas.

Alguns trabalhos buscam a utilização dos *middlewares* de TV digital para agregar as funcionalidades de automação residencial, como ocorre como o trabalho de (OLIVEIRA

et al., 2008b), onde é proposto um sistema denominado de DIGA Ginga (*Digital Automation in Monitoring and Control using GINGA technology*).

O DIGA Ginga é um conjunto de propostas baseadas profundamente no desenvolvimento do *middleware* Ginga. Este projeto gera uma série de subsistemas sendo eles o DIGA SAÚDE (monitoramento pessoal) o DIGABEM (base global) o DIGA CASA, com o monitoramento da casa, dentre outros subsistemas denominados de DIGA. Seu protótipo é o projeto Pimenter (OLIVEIRA et al., 2008b), desenvolvido pela CEFET do Ceará. A idéia básica por trás do projeto é implementar uma série de serviços desprezando as características de *hardware*, esperando que os receptores de TV Digital evoluam respeitando a lei de Moore (PATTERSON; HENNESSY, 2008), tal como os PCs.

Outros trabalhos como o de Lin et al (LIN; CHEN, 2005) caminham em uma direção completamente oposta, abordando formas de se controlar o receptor de TV digital (vide figura 2.30), utilizando para isso diversos dispositivos, como por exemplo computadores, PDAs e celulares com acesso a internet. No mesmo trabalho é usado é utilizado o *middleware* MHP para a confecção de um servidor que, interligado pela rede IP, é capaz de acionar o EPG, uma torradeira (LIN; CHEN, 2005), dentre outros.

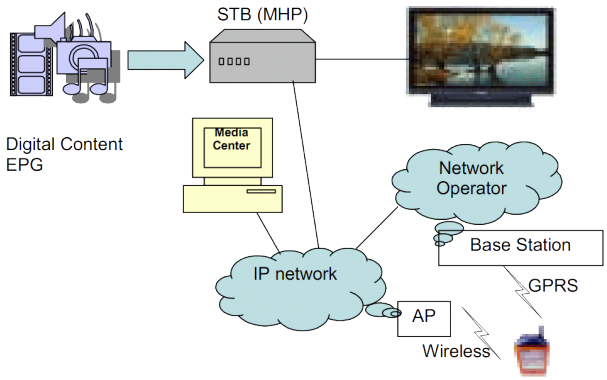

Figura 2.30: Modelo do sistema de controle remoto de TVD

No trabalho de Lin et al (LIN; CHEN, 2005), o servidor também se baseia em um conjunto limitado de comandos, expressos no caso por números de -1 até 9, onde todos os comandos, exceto o -1, são seguidos pelo caractere "retorno de carro". Quanto aos clientes, eles necessariamente precisam ter suporte ao HTTP, porém se possuírem capacidade de "tocar" um *stream* (fluxo) de vídeo do tipo MMS (*Microsoft Media Server*) ainda se tem a possibilidade de se ver o conteúdo que está sendo executado atualmente.

2.7 ESTUDO SOBRE *HARDWARES* UTILIZADOS NESTA DISSERTAÇÃO

Para a melhor realização desta dissertação, optou-se por um estudo sistemático dos *hardwares* e *softwares* que irão participar direta ou indiretamente da elaboração desta dissertação. No grupo dos *hardwares*, foi realizado uma divisão tal como apresentado na figura 2.31.

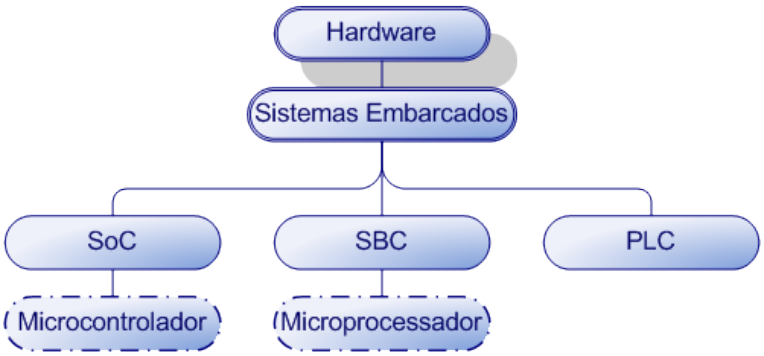

Figura 2.31: Estrutura do estudo de *Hardware*

Todos os *hardwares* estudados podem ser entendidos como sistemas embarcados, devido as suas características de tamanho, mobilidade e complexidade. Para uma melhor compreensão dos temas abordados, estes sistemas embarcados foram divididos em 3 grupos de estudo: (a) SoC (*Sistem on Chip*), (b) SBC (*Single Board Computer*) e (c) PLC (*Programeble Logic Controler*). Somente o grupo dos SoC tiveram um estudo mais aprofundado, analisando algumas características dos microcontroladores. Quanto ao PLC, este foi estudado a fim de servir de comparativo com as outras tecnologias e formas de automação 2.31.

2.7.1 SoC

SoC (*System on Chip* sistema em um circuito integrado) é um grupo de sistemas que se encontram quase que internamente dentro de um único dispositivo de circuito integrado. Os SoCs são a evolução natural dos dispositivos eletrônicos, que tiveram o início da sua integração com o desenvolvimento do circuito integrado, comportando vários transistores em uma única pastilha de silício.

Com a melhora nos processos de fabricação (vide figura 2.32) de circuitos integrados, ocorreu a necessidade do melhor gerenciamento do processo de fabricação, bem como a reutilização da propriedade intelectual desenvolvida para outro projeto. Assim, as empresas desenvolvedoras de circuitos integrados começaram a ter um repositório de partes de CIs, como por exemplo, unidades de memória ou processadores. Com essa reutilização de tecnologia se tem uma gama muito maior e configurável de CIs, destinados a diversos propósitos. Pode-se ter em um único CI um processador com memória RAM e ROM, comunicação serial, USB e até mesmo um *driver* de vídeo.

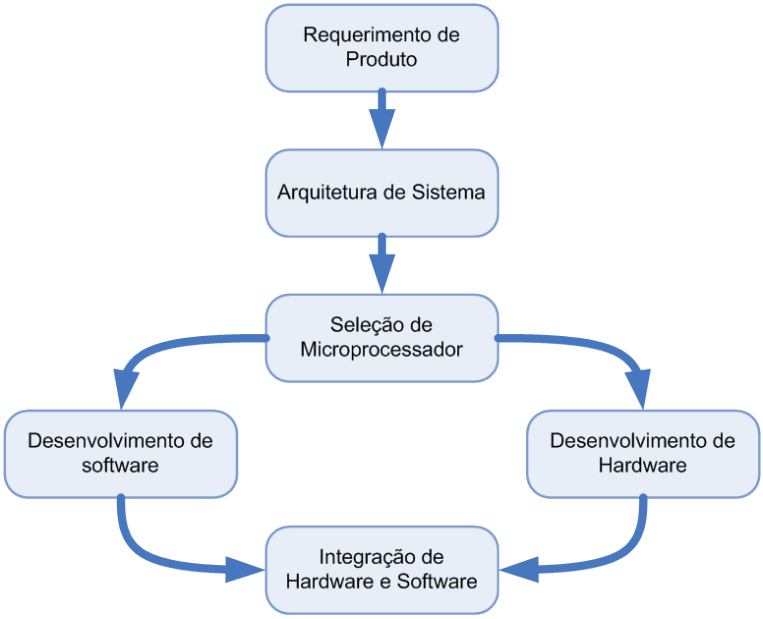

Figura 2.32: Processo de desenho de sistemas embarcados. Fonte: (DELMAR, 2004) (Modificado)

2.7.2 SBC

Single-board computers(SBCs) são computadores confeccionados em uma única placa PCI, incluindo em sua estrutura um microcontrolador ou um microprocessador, memórias RAM e ROM e, geralmente, vem com grande capacidade de IO, com entradas ADCs e DACs. Existem diversos SBC de diversas empresas e de diversas arquiteturas de processador (Zilog Z80, família Intel x86, Motorola 68000) (DELMAR, 2004), onde em quase a sua totalidade possuem a capacidade de executar alguma distribuição de Linux ou de Windows CE.

2.7.2.1 Exemplos de SBCs

Agora serão apresentados dois exemplos de SBCs, um da empresa *Vesta Tech* e um da empresa *Tecnology System*. Estes exemplos visam demonstrar a evolução dos SBCs.

1. *Vesta Tech:* um exemplo de SBC é o SBC da Vesta Tech (vide figura 2.33) que se utiliza de um microprocessador Motorola, com até 1 MB de RAM e até 1 MB de EPROM.

Figura 2.33: SBC da Vesta Tech. Fonte (DELMAR, 2004)

2. *TS7300:* um exemplo outro exemplo SBC é o TS7300 (vide figura 2.34) da empresa norte americana *Tecnology System* que é baseado no microcontrolador ARM920T de 200 MHz EP9302 da Cirrus que vem com uma FPGA e um Linux embarcado, onde o sistema operacional é instalado em uma das duas memória SD e que pode ser conectado a vários periféricos.

   Esta SBC pode ser utilizado para equipamentos de rede, como *gateways* e *firewalls*. Nesta dissertação, este STB representa um dos ambientes de teste de portabilidade do código do *Gateway* BACnet proposto.

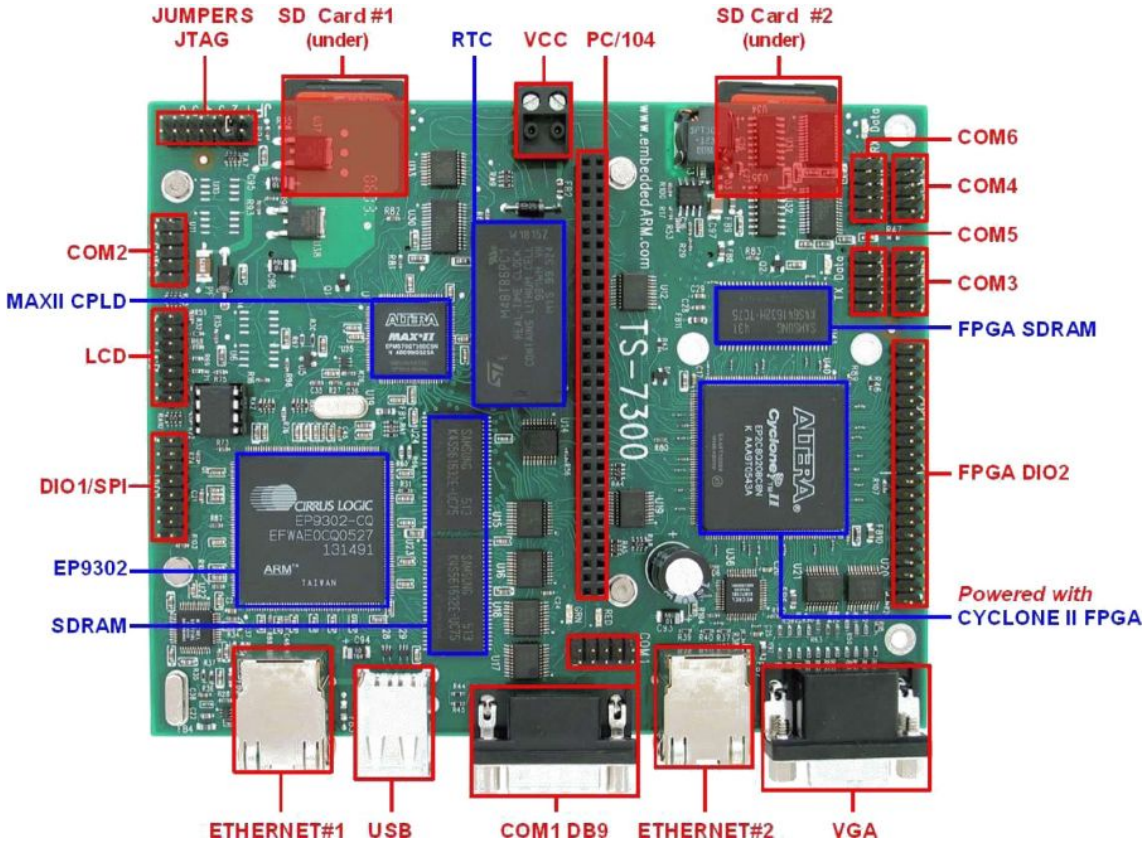

Figura 2.34: TS-7300 (Tecnology System SBC). Fonte (BTL, 2008)

2.7.3 PLC

Programmable Logic Controller ou em português CLP (Controlador Lógico Programável) é um sistema completo baseado em um microprocessador, desenhado especificamente para as tarefas de controle (DELMAR, 2004). O PLC geralmente está vinculado a automação industrial, devido ao seu histórico e custo mais elevado.

O PLC possui como elementos fundamentais: (a) unidade de processamento, geralmente um microprocessador, (b) interfaces de entrada e saída, divididas em acionadores digitais e analógicos, a relê ou a triac e (c) um padrão rígido de entrada de dados, definidos por padrões específicos. Existem PLCs de diversos tamanhos e capacidades, além de possuírem IPIs (Índice de Proteção Industrial) diferentes. Os PLCs geralmente possuem a facilidade de agregação de mais portas de Entrada e Saída, com a aquisição e instalação de novos "cartões" de expansão. A linguagem que se usa geralmente para a programação de PLCs é a Ladder, que é uma linguagem gráfica voltada para a tecnologia de acionamentos. Um exemplo de implementação em ladder de uma planta industrial é apresentada na figura 2.35.

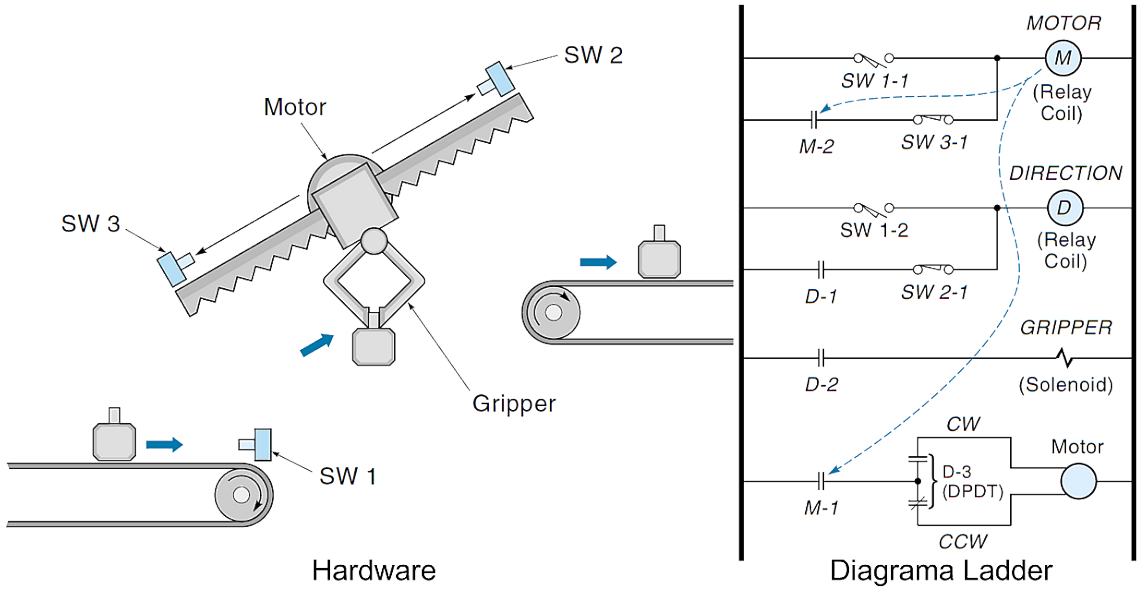

Figura 2.35: Implementação de automação com linguagem ladder. Fonte: (DELMAR, 2004)

2.7.4 Estudo dos microcontroladores utilizados nesta dissertação

Seguindo com a revisão bibliográfica de nivelamento de conhecimentos, esta dissertação apresenta uma pequena introdução aos microcontroladores, principalmente voltado para os leitores sem costume com a tecnologia embarcada, em especial aqueles oriundos da Engenharia de Redes. Esta introdução visa apresentar o conhecimento mínimo necessário e a familiarização com os principais termos utilizados, bem como a contextualização necessária para a compreensão dos microcontroladores utilizados nesta dissertação.

Os microcontroladores são circuitos integrados compreendidos como microcomputadores contidos em um único chip (*Single-Chip Microcomputers*) (DELMAR, 2004). Um microcontrolador é composto pelo menos das seguintes partes: (a)ULA (Unidade Lógica Aritmética), (b) memória RAM e ROM, ambas em pequena quantidade e geralmente com pinos voltados para a expansão deste espaço, (c) Portas digitais de entrada e saída, (d) portas analógicas de entrada ou ADC (Conversor Analógico Digital), (e) portas analógicas de saídas, geralmente PWM (Modulador por Largura de Pulso).

Um microprocessador não é capaz de executar um programa, devido a inexistência de itens básicos, tais como memória e capacidade de IO. Contudo, mesmo o microcontrolador mais simples é capaz de conter um programa a ser executado, ler informações do meio e devolver os resultados processados.

Os microcontroladores são conhecidos como controladores embarcados devido ao fato de serem utilizados no controle de algum componente, seja um robô industrial, um brinquedo, um estabilizador de corrente ou um portão automático, e se encontra dentro do próprio componente, daí a aplicação do termo embarcado. Um microcontrolador se diferencia de um microcomputador nas seguintes características:

- Um microcomputador é composto de diversas peças, placas e CIs, enquanto no microcontrolador todos os periféricos básicos já estão contidos dentro de um único CI, tendo dimensões muito menores, além de melhor aproveitamento energético.
- As palavras digitais de um microcontrolador são geralmente menores do que o de um microcomputador, devido a necessidade de menor precisão nos cálculos. Atualmente, os computadores possuem palavras de 32 ou 64 bits com algumas partes com barramento de 128 bits, enquanto a maioria dos microcontroladores possuem palavras de 8 bits, contudo vem crescendo a utilização dos microcontroladores em 16 e 32 bits.
- A frequência do relógio de um microcontrolador é geralmente menor de que um processador, mas atualmente esta relação está sendo revista, levando em conta os novos microcontroladores com frequência de cerca de 1 GHz (ARM11 Cortex) (ARM, 2000). Assim, os microcontroladores podem ser mais bem qualificados como tendo uma maior gama de frequência de relógio em relação aos microcomputadores (vide figura 2.36), contudo devido a limitações geométricas e térmicas a sua frequência de operação tende a ser menor do que a dos microcomputadores.

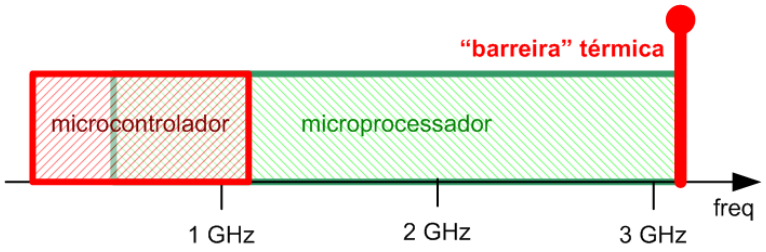

Figura 2.36: Comparativo de freqüências entre microcontroladores e microprocessadores.

- O conjunto de instruções *(instruction set)* de um microcontrolador possui geralmente menos instruções de acesso a memória do que um microprocessador, devido ao fato do primeiro possuir menos memória do que o segundo. Por outro lado, o conjunto de instruções do microcontrolador é rico em operações em nível de bit *(bit-handling)* devido ao fato de que uma saída paralela de um microcontrolador geralmente controla uma série de sub-dispositivos, assim estas funcionalidades facilitam o controle de cada parte do objeto no qual este está embarcado.

Os microcontroladores estão presentes em uma infinidade de equipamentos do nosso cotidiano, desde brinquedos eletrônicos como carrinhos até bonecas, eletrodomésticos (máquina de lavar roupas, geladeira, dentre outros) carros (o computador de bordo nada mais é do que um microcontrolador) portões eletrônicos, celulares e mp3/mp4 *players*. Delmar (DELMAR, 2004) indica que os microcontroladores tiveram grande aceitação na nova era de controle de sistemas via eletrônica devido a sua grande flexibilidade e seu crescente poder computacional.

2.7.4.1 Microcontroladores populares em automação domótica

Alguns microcontroladores conseguiram uma grande aceitação de mercado, tornando-se verdadeiros padrões de fato, como por exemplo o 8051 (DELMAR, 2004), que é comercializado por diversos fabricantes. Outro microcontrolador muito popular são os da família PIC comercializado pela empresa *Microchip Technology*, que mesmo contendo um grupo limitado de periféricos embarcados, dentre eles (vide figura 2.37) memórias ROM, EPROM, EEPROM, ADCs, temporizadores e portas seriais UART (DELMAR, 2004) são o alvo de diversos produtos de automação residência e predial.

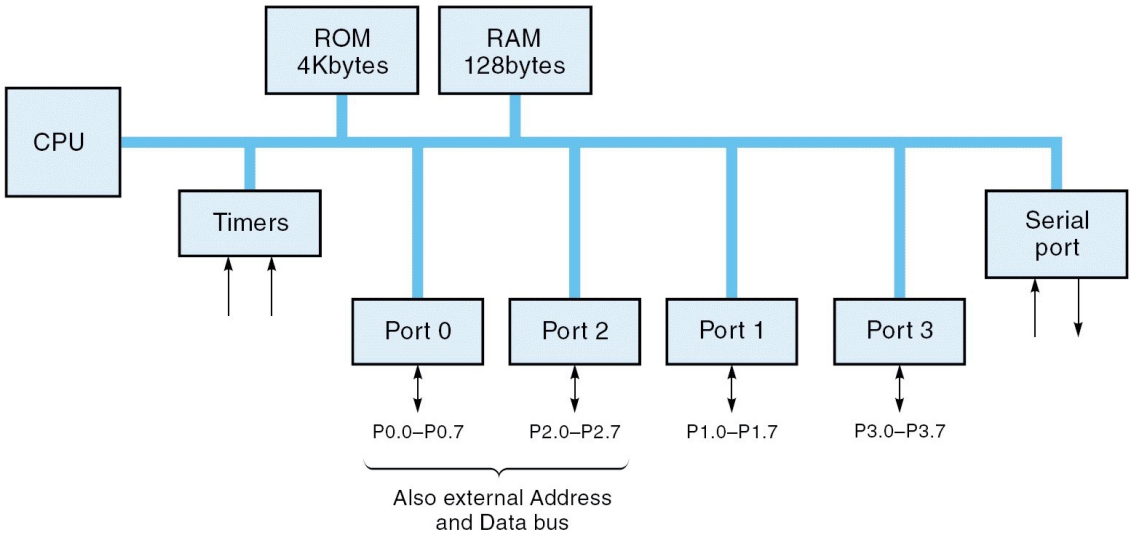

Figura 2.37: Diagrama de blocos de um Microcontrolador. Fonte: (DELMAR, 2004)

Os microcontroladores geralmente estão comercializados em versões com palavras de 8, 10, 16 e 32 bits, os quais se adéquam a um dado grupo de soluções. Contudo, é verificado uma tendência a utilização de microcontroladores de palavras maiores, geralmente de 32 bits, ou como o caso do ARM, que possui a possibilidade de operar em 16 ou 32 bits. Tal opção é motivada pelo barateamento dos microcontroladores bem como a necessidade da entrega ao mercado de produtos mais sofisticados e em

um tempo muito menor, levando a utilização de processadores que suportem sistemas operacionais modernos.

Nesta dissertação, será analisado com um pouco mais de profundidade o AVR, ARM e o ST40, que são os microcontroladores utilizados na elaboração dos protótipos e na implementação das softwares gerados nesta dissertação. É interessante verificar que o ARM e o ST40 são processadores de 32 bits com suporte a Linux, o que facilitou o desenvolvimento dos protótipos.

2.7.4.2 O microcontrolador AVR

AVR é um microcontrolador da família RISC desenvolvido pela empresa ATMEL, que está voltado para o mercado de baixo custo. Na automação o AVR é utilizado em pequenos dispositivos e como circuito de integração entre componentes, como por exemplo como controlador de um dispositivo de LCD. A sua família tradicional possui palavras de 8 bits, contudo existe uma sub-família chamada AVR32, que é de 32bits e pode executar sistemas operacionais modernos como por exemplo o Linux.

Uma dos pontos fortes dos AVRs é que o projeto do processador se deu no mesmo momento do que seu compilador, sendo assim, o tamanho de um código escrito em C e o seu equivalente lógico escrito em linguagem de montagem difere muito pouco.

A pilha BACnet foi compilada para o ATMega168 no projeto *BACnet Stach* (BACNET, 2009), que é um AVR com mais memória do que o ATMega8, porém com o mesmo conjunto de instruções. Existem também várias outras pequenas soluções de automação que se utilizam deste tipo de microcontrolador devido ao seu baixo custo e facilidade do obtenção no mercado.

Nesta dissertação foi implementada parte da pilha BACnet para dentro de um microcontrolador ATMega328P (vide seção 3.6), a fim de se ter um dispositivo BACnet de baixo custo, voltado para as atividades de iluminação. Para a concretização deste intento, foi confeccionado um conjunto simplificado de placas de referência com o ATMEGA328P e outros dispositivos passivos a fim de se ter um controle de um ponto de iluminação. Devido ao fato dos conjuntos de instruções do ATMEGA168 e do ATMEGA328P serem os mesmos, se fez necessário apenas a mudança do processador alvo e alguns poucos parâmetros para que o código fornecido pelo projeto *BACnet Stach*

fosse portado de um microcontrolador para outro. Também houve uma customização do dispositivos (*device*) BACnet para atender ao interesse de projeto.

2.7.4.3 ARM

ARM (*Advanced RISC Machine*) é um núcleo de processamento (*core*) comercial que é fornecido por diversos fabricantes de microcontroladores e que possui um grande número de produtos que os utiliza, sejam PDAs, *media players*, *smartphones* (IPHONE, 2010), impressoras, centrais de automação de carros, dentre outros.

Nesta dissertação, foi utilizado como ambiente de testes e validação de modelo de portabilidade do *gateway* BACnet um *Kit* de desenvolvimento ARM9, modelo TS7300. (vide fig. 2.34).

a) Historico: Em 1983 a empresa britânica Acorn Computers começa a desenvolver um computador pessoal (LEVY, 2005) de 16 bits. Porém ainda na fase de especificação do processador seus engenheiros encontraram somente processadores que possuíam um conjunto de instruções *instruction sets* muito complexo, aonde cada instrução chegava a demorar centenas de ciclos até ser executada. Desta forma, cada interrupção é penalizada com muito tempo gasto para completar a requisição, tornando assim o desempenho total do processador muito abaixo do que as memórias daquele tempo. Encontrando tais dificuldades, os engenheiros da Acorn Computers optaram pelo desenvolvimento de seu próprio processador, baseados no projeto do Berkeley RISC 1, desenvolvendo em apenas dois anos seu primeiro processador de 26-bits e 25000 transistores, o *Acorn RISC Machine* (ARM), sendo este lançado em 1985, com desempenho semelhante ao Intel 80286.

A escolha de desenvolver seu próprio microcontrolador foi audaciosa por parte dos engenheiros da Acorn Computers, visto que nos idos de 1980 o desenvolvimento de um processador demorava vários anos, e a empresa não possuía experiência na área. Em 1987 foi lançado o segundo processador da família, sendo que a principal melhoria foi o suporte a co-processadores, e logo depois foi acrescido memória cache no próprio CI.

Em 1990 a Apple decidiu-se pela utilização do ARM em seu PDA Newton, em uma decisão estratégica que levou a um consórcio que deu origem a empresa ARM, agora

acrônimo de Advanced RISC Machines. Nesta época o ARM se tornou 32-bits, ganhou suporte MMU e acumulador de instruções de 64-bits. A partir de então a ARM afastou-se do ramo de desktops indo para o mercado de sistemas embarcados.

Em 1996 foi acrescido ao ARM a arquitetura Thumb de 16-bit, que propicia uma redução do espaço médio do código em 40 por cento em relação aos códigos de 32-bits. Tanto o *iPod player* quanto o iPod vídeo utilizam os recém lançados core ARM7TDMI, com impressionantes 30000 transistores. Por esta ocasião, a Intel lança o processador StrongARM, que contém o core ARM cujo direitos autorais foram pagos a ARM.

Em 1999, se tem como principais inovações suporte a DPS e extensões para os *bytecodes* Java. Nesse ano a Intel lança XScale, contendo o novo core ARM. Nos outros anos que se seguiram foram acrescentadas novas melhorias no *core*, contudo a grande evolução ocorreu em 2009 foi introduzida a família Cortex, com velocidade de processamento na ordem de GHz, com a volta do ARM ao mercado de computadores, como o ATOM (vide figura 2.38).

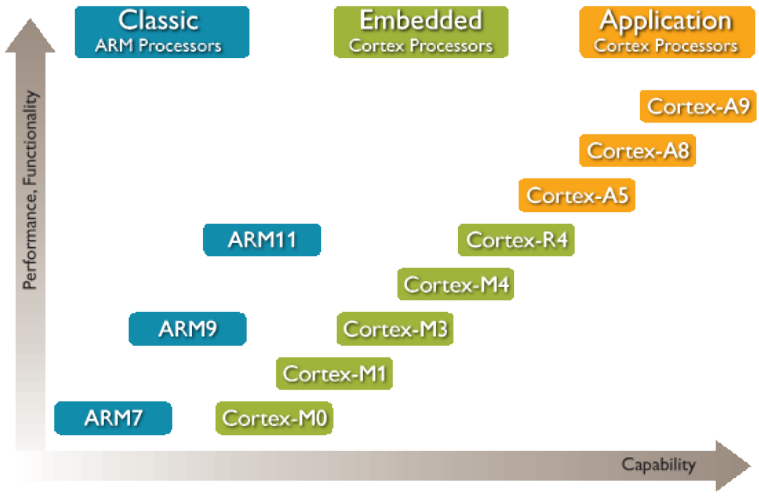

Figura 2.38: Realidade atual dos processadores ARM. Fonte: (ARM, 2010)

2.7.5 O *core* de microcontrolador ST40

ST40 é um microprocessador de 32-bits de arquitetura RISC (vide figura 2.39) *(Reduced Instruction Set Computer)* (HEATH, 2007) desenvolvido pela empresa ST (STMICROELECTRONICS, 2009) que é comercializado em uma série de microcontroladores com um grupo bem diversificado de periféricos. Este microprocessador vem com MMU, que

permite o seu uso com sistemas operacionais modernos e FPU, que permite ao processamento de vídeo. Nesta dissertação, foi utilizado o ST7100 (vide a seção 2.7.5.1) , que utiliza o core ST40.

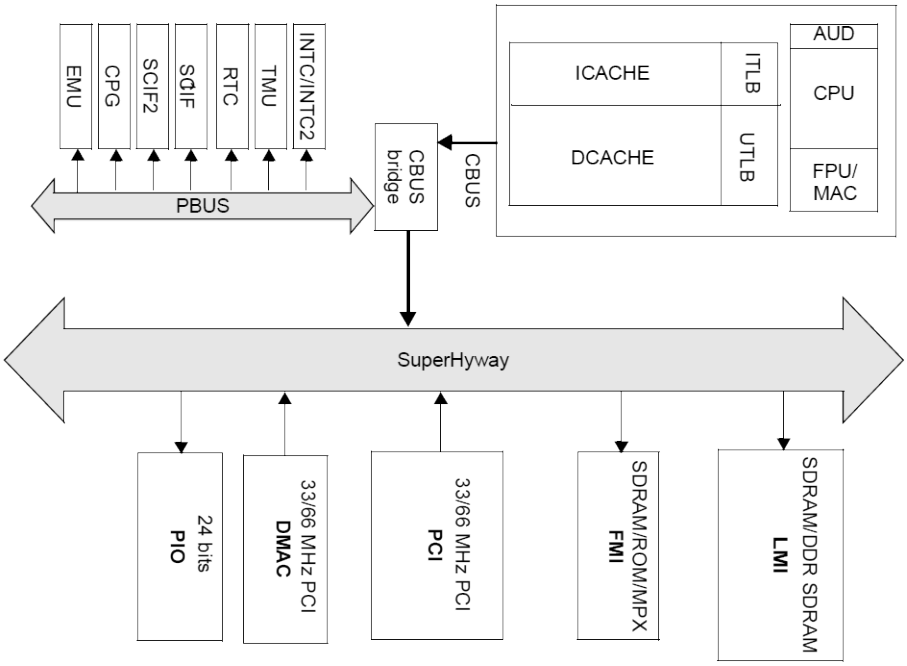

Figura 2.39: Arquitetura de um microcontrolador ST40. Fonte: (UM0339, 2007)

Assim como o ARM, o ST40 não é comercializado sozinho em um CI, mas sim com outros periféricos que o conferem as características de SOC, pelo fato de tornar possível a confecção de um sistema completo, no caso um receptor de TV digital, com a utilização de poucos componentes além do SOC.

O *core* ST40 possui uma CPU proprietária da *Hitachi* denominada arquitetura SuperH ou simplesmente SH. Com uma arquitetura de 32bits com *hardware* específico para cálculos de ponto flutuante de 64bits. Na versão de 200MHz chega a 360MIPS. Também possui a capacidade de endereçar até 4Gbytes de memória. No seu barramento é capaz de transferir dados até uma velocidade de 664 Mbytes/s.

Os dados são transmitidos internamente dentro do *core* ST40 através de um padrão proprietário chamado SuperHyway, que interliga o PIO de 24bits o PCI e as memórias a CPU através de uma ponte para o barramento CBUS, também proprietário.

O *core* ST40 possui alguns periféricos básicos, que são sempre fornecidos com o *core*, como por exemplo um RTC (vide figura 2.39). Estes periféricos básicos são interligados a CPU através de um barramento PBUS, passando contudo pela ponte de CBUS, visto

que este é o barramento da CPU.O ST40 tem dois MMU (*Memory Management Unit*), uma para cada memória cache. Esta arquitetura torna possível os microcontroladores que tem o ST40 como *core* executar sistemas operacionais modernos, como o Linux e o Windows. Possui versões comerciais de WindowsCE e livre de Linux (STLinux) a disposição no mercado (HEATH, 2007), (UM0339, 2007).

Segundo (STMICROELECTRONICS, 2005), o ST40 possui compiladores próprios em C/C++, depuradores e OS proprietário. Dentre os programas de outros fornecedores voltado para o ST40, existe o *WindowsCE* da *Microsoft*, o *JavaOS for consumers* da *Sun*, o *VxWorks, Tornado tools* da *WindRiver*, o Linux, a *Insignia JVM* e o *ANT browser*.

O core ST40 é utilizado em vários microcontroladores voltados para a confecção de STBs no mercado brasileiro, como por exemplo o DIGI TV da Positivo (POSITIVO, 2010), XPS 1000 da Proview (PROVIEW, 2010) e o DTH1900 da Century (BRASIL, 2010). Tal penetração no mercado brasileiro tornou o ST40 como família de processadores a serem adotados para a elaboração desta dissertação, no tocante a processadores de STBs, a fim de se ter uma grande gama de produtos comerciais compatíveis com a arquitetura proposta, podendo até mesmo serem aproveitados os arquivos binários fornecidos.

2.7.5.1 O microcontrolador ST7100

Dentre os microcontroladores que utilizam o ST40 como microprocessador existe o ST7100 que é voltado para aplicações de TV digital. Este SoC é utilizado por diversos STBs comerciais (POSITIVO, 2010; PROVIEW, 2010; BRASIL, 2010) além do *kit* de desenvolvimento MB442 (vide figura 2.40). Muitos STBs foram fabricados com base neste *kit*, tornando este uma escolha natural de pesquisa. O estudo deste microcontrolador vem a atender o objetivo específico (vide seção 1.3.2) de estudar plataformas abertas de *hardware-software* de STBs comerciais.

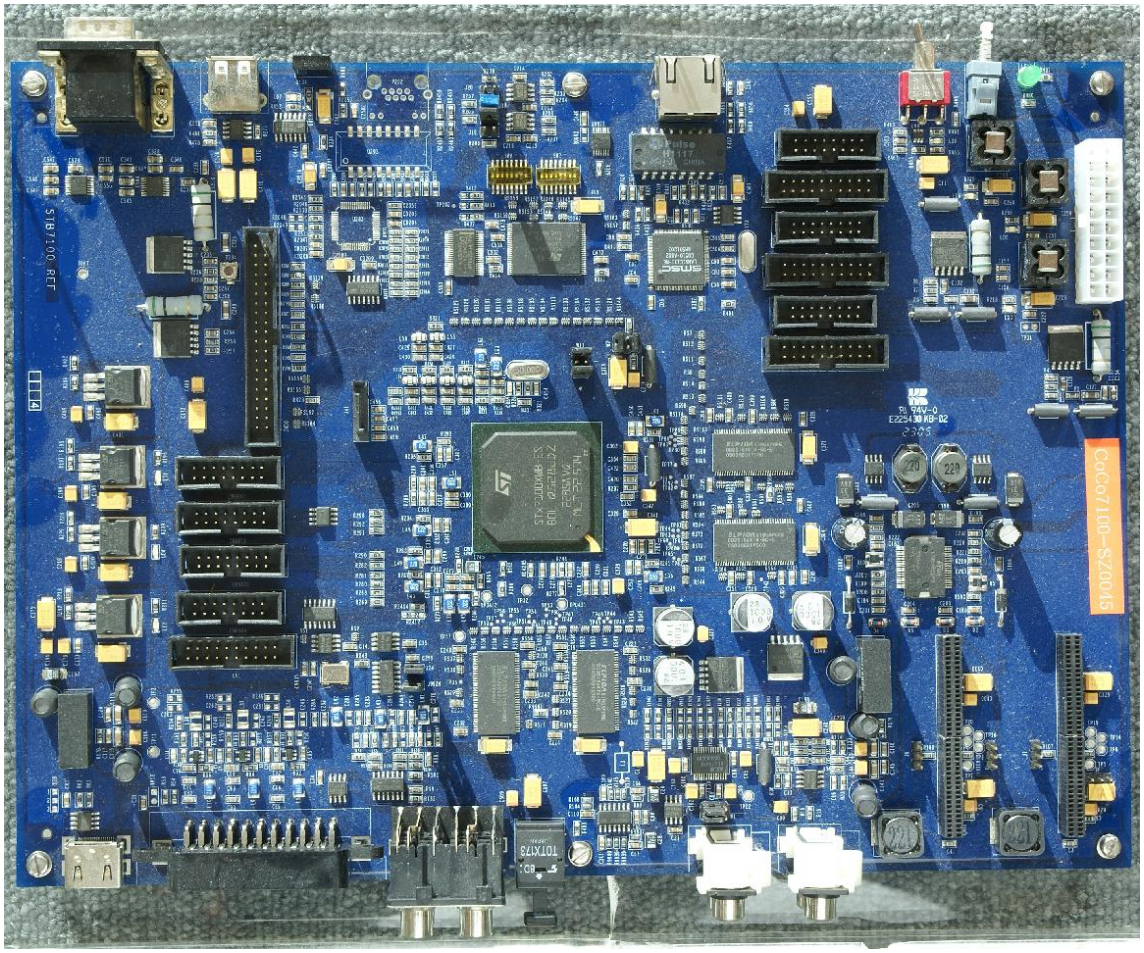

Figura 2.40: Kit de desenvolvimento mb442-revB-AP para desenvolvimento de microcontroladores ST7100. Fonte (STLINUX, 2009)

2.8 ESTUDO SOBRE *SOFTWARES* UTILIZADOS NESTA DISSERTAÇÃO

Os *softwares* foram utilizados nesta dissertação tanto como tanto como ferramenta de desenvolvimento quanto como servirem de apoio de testes e como ferramenta conceitual para o desenvolvimento do código (vide figura 2.41).

Dentre os *softwares* que foram utilizados como ferramenta de desenvolvimento encontra-se o Eclipse (ECLIPSE, 2010) como ambiente de desenvolvimento e o GCC (STALLMAN, 2002), como ferramenta de compilação. Como ferramenta de apoio cabe comentar o Wireshark (WIRESHARK, 2010), que foi utilizado para analisar os pacotes BACnet através da rede.

Alguns *softwares* foram disponibilizados como produto, ou seja, como parte integrante do sistema utilizado pelo usuário. Dentro deste grupo, têm-se o Linux, em especial o

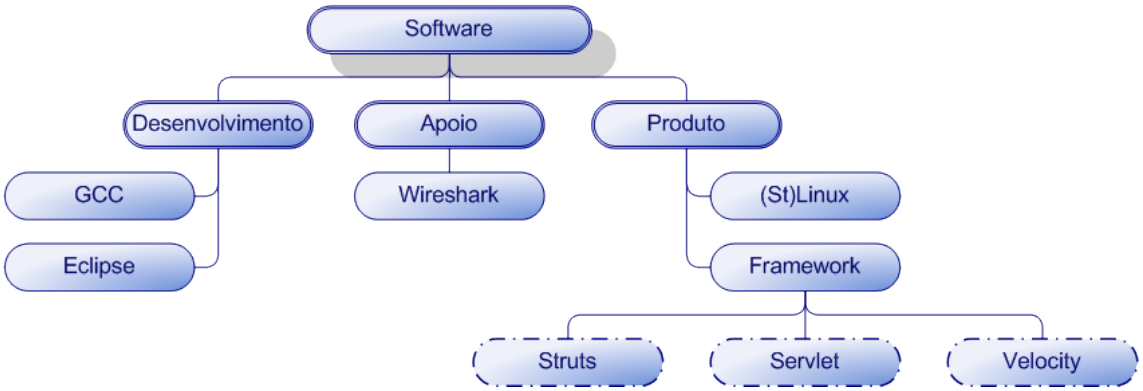

Figura 2.41: Estrutura do estudo de *Software*

STLinux disponibilizado como sistema operacional do STB e alguns *frameworks* que serviram de base teórica como os *Servlets* (SERVLET, 2010) do Java (JAVA, 2010), o *framework* Velocity (VELOCITY, 2010) e o *framework* Struts (STRUTS, 2010b),(STRUTS, 2010d) e (STRUTS, 2010c), dos quais muitas das decisões de implementação derivaram.

2.8.1 O sistema Operacional embarcado STLinux

O ST40 por possuir MMU é capaz de rodar o sistema operacionais Linux, sendo que existe um grupo que se dedica a distribuição de uma versão (vide figura 2.42) voltada especialmente para os microcontroladores que possuem esse microprocessador. O STLinux é o SO embarcado no STB utilizado nesta dissertação.

Figura 2.42: Logotipo do STLinux

O STLinux é uma distribuição Linux baseada no kernel 2.4 (STLINUX, 2009) e possui sua estrutura parecida com a do RedHat. A mesma equipe que distribui o STLinux também distribui um *Workbench* baseado no Eclipse para o desenvolvimento de códigos e alterações no kernel em um ambiente Linux para PC, onde se faz necessário a instalação de uma distribuição RedHat ou Fedora (STLINUX, 2009), além dos pacotes do STLinux, dentre eles o GCC.

2.8.2 A plataforma de desenvolvimento computacional GCC

Conforme (STALLMAN, 2002) o GCC é uma coleção de ferramentas de compilação que se pode ter como sigla de *GNU Compiler Collection* (Coleção de Compilador GNU) ou, no caso de se ter interesse de enfatizar a compilação de programas na linguagem C, se pode ter como abreviação de *GNU C Compiler* (Compilador C GNU).

O GCC é um conjunto de diversas ferramentas, dentre eles compiladores, ligadores, dentre outros, capaz de gerar códigos binários para várias arquiteturas, processadores e sistemas operacionais, sendo a ferramenta de compilação mais completa e configurável que existe atualmente. O GCC, geralmente, está vinculado aos sistemas operacionais UNIX, em especial o Linux, contudo existe possibilidade de gerar códigos binários para outros sistemas operacionais, como o Windows e o MACOS, além de se poder ir mais além, gerando código executável para outras arquiteturas, como para ARMs, AVRs, ST40 dentre outros.

2.8.3 O ambiente de desenvolvimento integrado Eclipse

O Eclipse é uma IDE (*Integrated Development Envoirement* Ambiente de desenvolvimento Integrado) (ECLIPSE, 2010) utilizado para a confecção de *softwares*, principalmente em Java, porém pode ser utilizado para qualquer outra linguagem, como por exemplo o LaTeXe o C.

As grandes vantagens do Eclipse é a sua tecnologia de *plugins*, que permite a sua expansão e configuração para cada caso, possuindo uma riqueza de ferramentas de edição de código, com o *code completion*, busca indexada de texto com expressão regular e identificação de erros e alarmes antes mesmo da compilação.

Nesta dissertação, o Eclipse foi utilizado como ambiente de desenvolvimento tanto do *gateway* quanto do *firmware* do dispositivo BACnet de referência, além de ser o ambiente de confecção desta dissertação, servindo de IDE para o LaTeX.

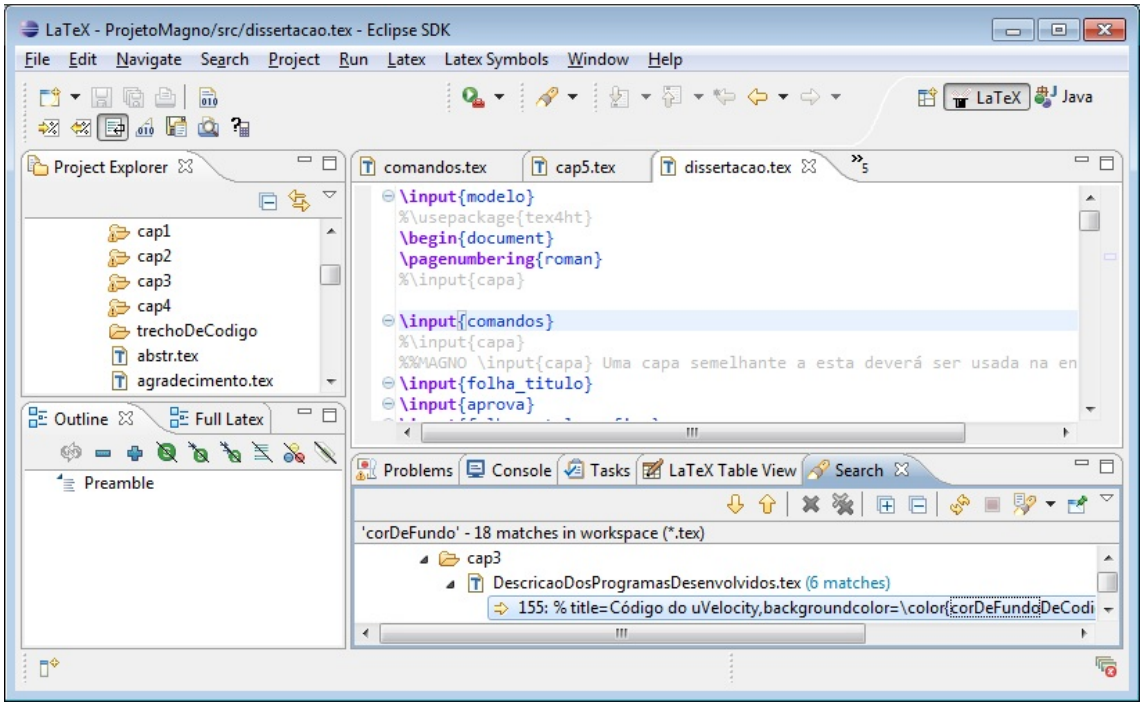

Figura 2.43: Execução da IDE Eclipse para geração de documentos em LaTeX

2.8.4 O analisador de protocolos Wireshark

O Wireshark (WIRESHARK, 2010) é um *software* que executa tanto em ambientes Windows (vide figura 2.44) quanto Linux/Unix que é capaz de capturar e analisar o tráfego (OREBAUGH, 2007) o tráfego pela placa de rede, com uma interface que torna possível a filtragem de alguns pacotes bem como a sua leitura bit a bit. O Wireshark possui suporte para mais de 750 protocolos (OREBAUGH, 2007), dentre eles o BACnet.

O Wireshark foi utilizado nesta dissertação para a análise dos pacotes BACnet/IP que eram enviados ou recebido nos protótipos em PC, para verificar a sua integridade e conformidade com o padrão definido pelo BACnet, fazendo parte da metodologia de estudo e desenvolvimento seguida nesta dissertação.

O Wireshark mostra todo o fluxo de dados da placa de rede, inclusive as mensagem que se destinam a outros IPs que estão no mesmo barramento. Para ser efetivo na análise de pacotes, o Wireshark fornece uma ferramenta de filtros avançada, capaz de identificar informações em diversos protocolos. Um exemplo de filtragem é o filtro de IP, definido por $ip.addr == XXX.YYY.ZZZ.KKK$. Para o BACnet é importante também fazer filtros de porta, que é definido pelo filtro $tcp.port == XXXX || udp.port == XXXX$, que filtra tanto o tráfego TCP quanto UDP.

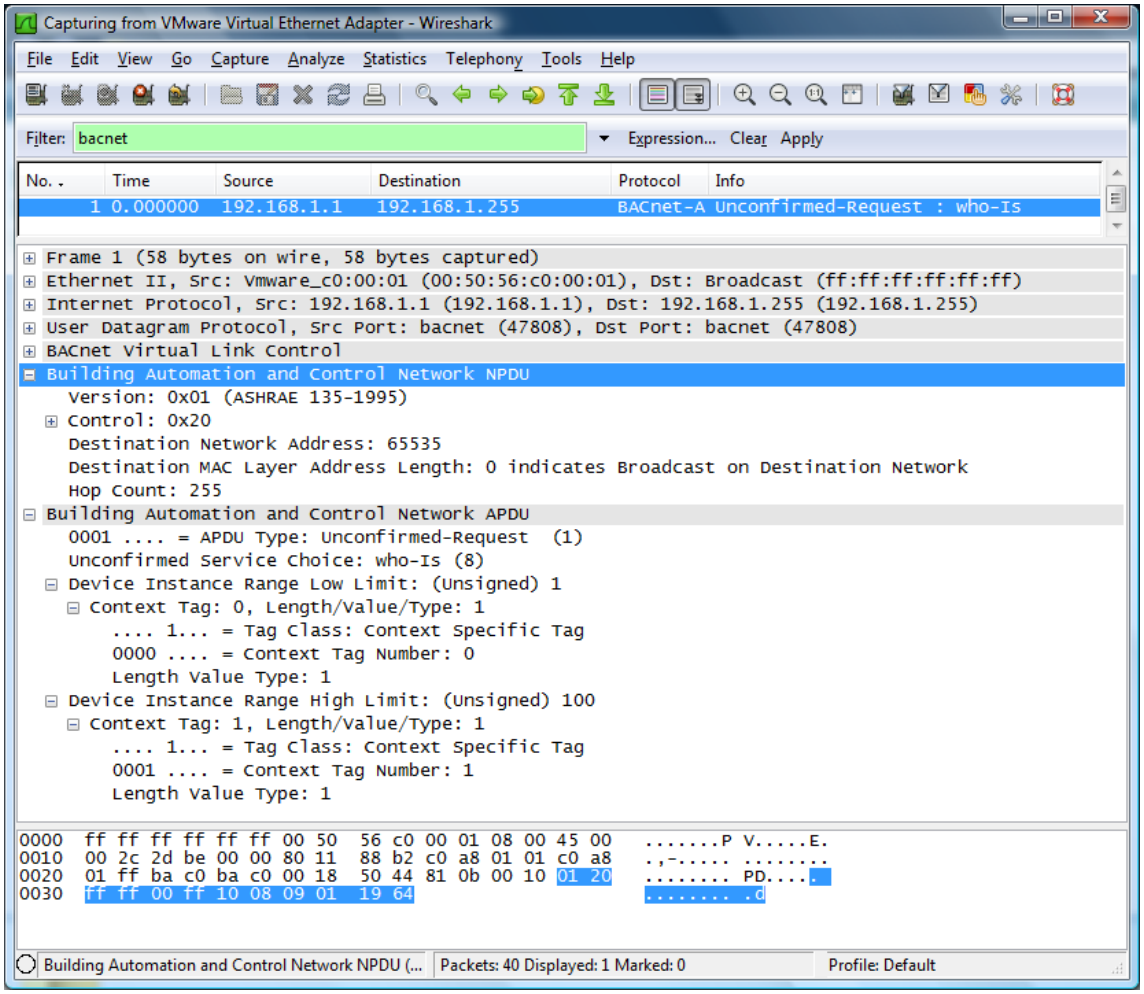

Figura 2.44: Tela do Wireshark executando no Windows e capturando um serviço BACnet *Who Is* nos endereços da faixa de 1 a 100

Um filtro importante, quando se está utilizando o Wireshark para interceptar pacotes BACnet, é o filtro *bacnet* o qual só intercepta pacotes bem formados do tipo BACnet (vide figura 2.45), que filtra somente os pacotes do tipo BACnet.

Outra função importante do Wireshark é a exportação dos pacotes capturados para um arquivo, seja para uma análise futura, ou para uma exportação dos pacotes octeto a octeto para a linguagem C.

Nesta dissertação, o Wireshark foi utilizado como analisador dos pacotes BACnet, verificando-se assim a integridade dos dados transmitidos, tanto utilizando na versão BACnet/IP quanto na versão BACnet/MSTP. Além do apoio na faze de desenvolvimento, o Wireshark foi utilizado também para a extração dos resultados do desenvolvimento do *gateway*, evidenciando a comunicação entre *gateway* e os dispositivos.

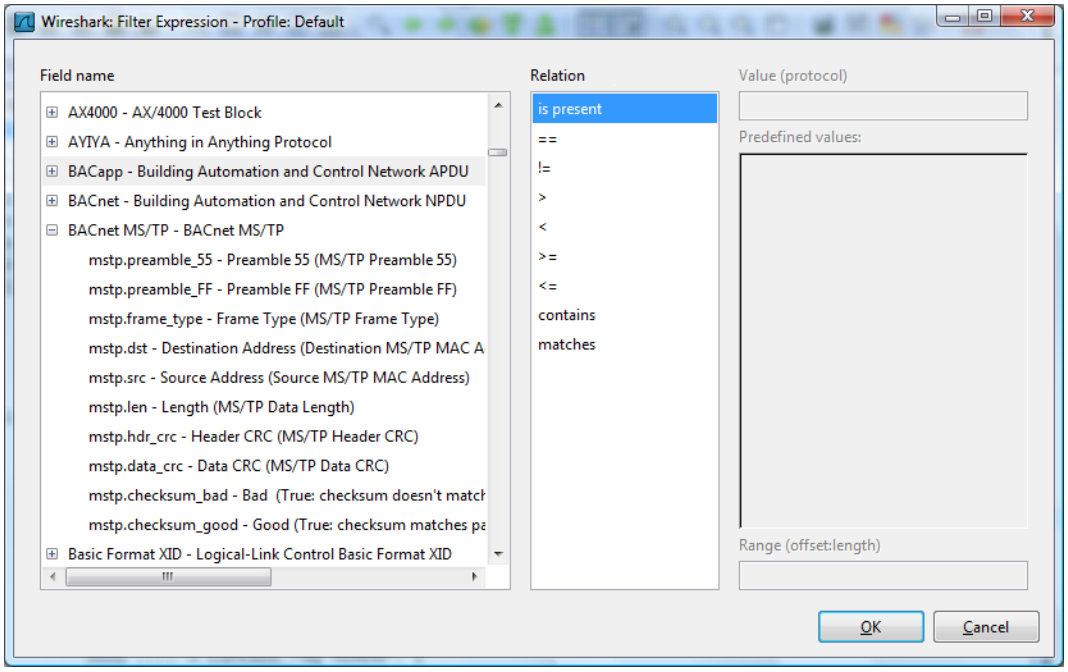

Figura 2.45: Tela de configuração do filtro do Wireshark para ser configurado para filtrar pacotes específicos BACnet

2.8.5 A tecnologia de desenvolvimento *web Servlet*

Servlets são componentes desenvolvidos pela Sun (atualmente de propriedade da Oracle (JAVA, 2010)), que são extensões do servidor que atuam na geração de conteúdo dinâmico para um aplicativo *web*. Seu nome se contrapõe ao nome Applet, que é executado no lado do cliente *(Application)*. Sua funcionalidade básica é processar requisições e retornar respostas, tornando factível a agregação de novas funcionalidades ao servidor.

Nesta dissertação, a tecnologia *Servlet* foi utilizado como referência no desenvolvimento da arquitetura do servidor web do *gateway*, provendo o mesmo nível de abstração propiciado pelo *Servlet* no Java, dentro do *gateway* em linguagem C. Esta abstração é provida pelas interfaces de *request* e *response*.

2.8.6 O *framework* de modelos Velocity

Velocity é um *framework* mantido pela *Apache Software Foundation*, muito utilizado para a geração dinâmica de documentos baseados em modelos *(templates)*, principalmente para conteúdo *web* (VELOCITY, 2010), porém pode ser utilizado para a geração de outros documentos. O desenvolvimento de soluções com o Velocity geralmente apresentam três elementos básicos, sendo eles: (a) o *template*, que é o modelo do arquivo a

ser gerado, (b) o código *server-side*, que será responsável pelo processamento dos dados a serem apresentados e (c) a *engine* do *framework*, que é responsável pela fusão dos dados no modelo. Do ponto de vista do desenvolvedor, o esforço a ser desempenhado para a geração de uma solução com o Velocity se encontra na elaboração do modelo e da codificação do lado do servidor.

O Velocity possuí uma linguagem de programação própria para o desenvolvimento dos *templates*, muito próxima ao HTML e ao JSP. Por padrão, os arquivos de *template* possuem a extensão ".vm".

Nesta dissertação, o Velocity foi utilizado como referência no desenvolvimento de um *framework* denominado uVelocity (lê-se micro-Velocity), que é provê a capacidade de fusão de um modelo com informações pontuais, sem estrutura de repetição.

2.8.7 O *framework* de MVC Struts

Struts é um *framework* mantido pela *Apache Software Foundation*, muito utilizado em aplicações *web* que provê uma forma de aplicação do padrão MVC (*Model View Controller*), onde o Struts é o *controler*. Para a parte de modelo (*model*), se tem a camada de persistência do sistema, geralmente se utilizando de Bancos de Dados Relacionais, arquivos, persistência em memória, dentre outros. Já na camada de apresentação (*view*) se tem o uso geralmente do Velocity, JSF, XSLT dentre outros. O Struts se comporta como uma ponte entre os dados contidos na camada Model para a parte de View *web* (STRUTS, 2010b).

2.8.7.1 Fluxo de execução básica do Struts

O caso de uso se inicia com o navegador realiza uma requisição para o servidor. O Struts inicia seus trabalhos executando os *interceptadores* próprios do desenvolvedor do sistema, que enriquecem ou tratam a requisição. Depois o objeto *action* é iniciado, e o *interceptador prepare* procura o *método prepare* dentro da *action* e, caso encontre, o executa.

Caso seja o Struts 2, ocorre a passagem dos parâmetros para dentro da *action*, e caso seja Struts 1, essa fase não ocorre, visto que o objeto *request* é passado para a *action* na

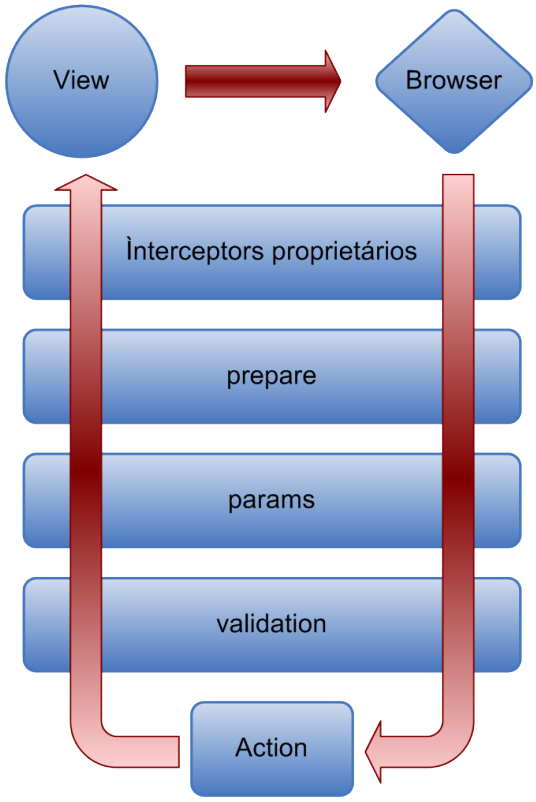

Figura 2.46: Fluxo de execução básica do Struts

figura da *ActionBean*. Por fim, é realizado as validações definidas pelo desenvolvedor do sistema. Como fluxo de exceção, caso em algum interceptador não se encontre as condições previstas pelo desenvolvedor, a solicitação é desviada para uma página de erro.

A última etapa é a execução do método *execute* da *action*, e no caso dos Struts 2, este método pode ser configurado para outro nome de método. Depois é retornado pela pilha de interceptadordes os resultados do processamento com o endereço de uma página de *view* a ser fundida com os dados provenientes da *action*. Após a fusão dos dados com o modelo de página, esta informação é enviada para o navegador.

Nesta dissertação o Struts foi utilizado como referência no desenvolvimento de um *framework* denominado uStrutus (lê-se micro-Struts), que é provê um alto grau de abstração na elaboração de novas funcionalidades ao servidor, não necessitando com esta ferramenta a compreensão de todo o *gateway* para o desenvolvimento de uma nova funcionalidade.

2.9 ASPECTOS BÁSICOS SOBRE PROJETOS DE NAVEGADORES

Nesta dissertação, é proposto o desenvolvimento de um navegador baseado em imagens, voltado para sistemas com limitações. Para a elaboração desta atividade, realizou-ze um estudo dos conceitos e aspectos de navegadores, a fim de se ter o suporte teórico necessário. Nesta dissertação, é definido como navegador todo e qualquer *software* destinado a apresentar um conteúdo interativo, provendo como mecanismo básico de saída um dispositivo visual (i.e. monitor, televisão tela de celular) e algum dispositivo de entrada para que ocorra a interação com o usuário (i.e. teclado, *mouse*, controle remoto). Uma definição mais simplista pode ser encontrada em (WHATBROWSER, 2010), que define navegador no âmbito de navegador de Internet, definindo navegador como *"O programa do computador que é usado para ver sites"*, sendo o *"programa mais importante do computador"* (WHATBROWSER, 2010).

2.9.1 Motivação para o desenvolvimento de um navegador

A utilização do *gateway* BACnet só é funcional para o usuário com a utilização de um aplicativo que apresenta de forma gráfica e intuitiva os dispositivos instalados, tornando assim possível o envio de comandos e a obtenção das informações da casa, sendo um canal bi-direcional entre o usuário e os diversos dispositivos automatizados que integram a solução de automação residencial.

Atualmente, diversos serviços e sítios de Internet (i.e Tweeter, Youtube, Facebook, dentre outros) possuem além de uma versão para ser utilizada em navegadores de internet para PCs uma versão para dispositivos limitados, como apresentado na figura 2.47, em especial celulares. A comunicação com estes aplicativos específicos é geralmente disponibilizados através de APIs públicas facilitando o desenvolvimento de uma grande variedade de aplicativos pelo público em geral. A necessidade do desenvolvimento de um navegador específico se deve ao fato de que um navegador de internet necessita de muita memória e processamento para apresentar uma informação, além da navegação em dispositivos embarcados serem diferentes de uma navegação comum em um PC, principalmente pelo fato do dispositivo embarcado não ter teclado nem *mouse*.

Para demonstrar o uso de memória na utilização de um navegador comercial voltado para a sua execução em computadores foram realizados 4 cenários de testes de utilização

Figura 2.47: Celular com navegador específico e de uso geral, com execução do Youtube em um navegador específico

em dois navegadores comerciais líderes de mercado, o Internet Explorer (IE, 2010) que utiliza o mecanismo de renderização Mosaic (IE-WIKI, 2010), (MOSAIC, 2010) e o Firefox (FIREFOX, 2010) que utiliza o mecanismo de renderização Gecko (GECKO, 2010). Os testes realizados consistiram-se na execução de um vídeo no Youtube (YOUTUBE, 2010) e no acesso de uma página de relacionamentos (FACEBOOK, 2010), utilizando como teste a utilização cotidiana dos navegadores. De acordo com os testes realizados (vide figura 2.48), se tem um consumo de memória sempre acima de 60MB, uma quantidade de memória muito superior do que a disponível na maioria dos dispositivos embarcados atualmente comercializados.

O procedimento de testes realizado consistiu executar o respectivo navegador diretamente na dada página a ser acessada, objetivando-se com isso a diminuição da interferência de outros sítio na utilização da memória por outros sítio. Para realizar esta medição foi utilizado o programa *ProcessExplorer* da *Sysinternals-Microsoft* (SYSINTERNALS, 2010). O Sistema Operacional utilizado foi um Windows Vista SP1. Devido

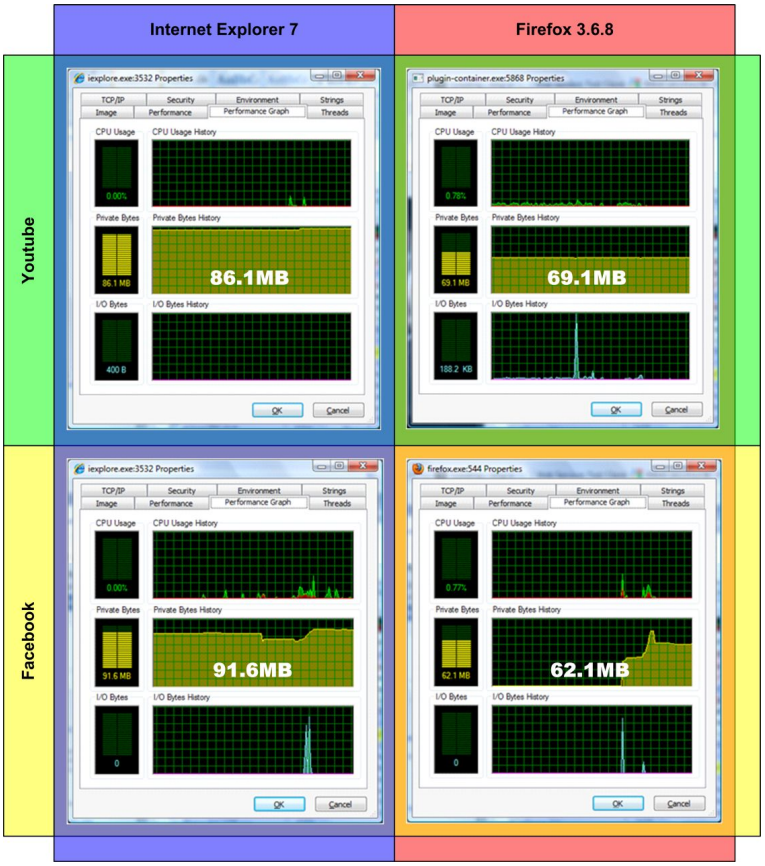

Figura 2.48: Comparativo de utilização de memória na execução de um vídeo no Youtube e uma página no Facebook em 2 navegadores

ao fato do Firefox delegar a um *plugin* a execução dos vídeos do Youtube, mediu-se diretamente a utilização de memória deste *plugin*.

2.9.2 Conceitos básicos no projeto de navegadores

Os navegadores são compostos de elementos básicos como ícones, janelas, blocos de texto. Estes elementos devem ser definidos e dispostos em uma interface gráfica de forma a facilitar a navegabilidade. O projeto do navegador proposto nesta dissertação segue as mesmas filosofias que norteiam os navegadores para sistemas embarcados, devido as características de limitação de *hardware* e de navegabilidade. Estas limitações são impostas, principalmente, pelo tamanho aparente da tela de TV e pela limitação do dispositivo de entrada, no caso o controle-remoto, muito semelhante a um teclado numérico de celular.

2.9.3 Conceitos básicos de navegadores para sistemas embarcados.

A realização de um paralelo entre os *smartfones* e os aparelhos de televisão nos quesitos de navegabilidade e a percepção visual é de grande importância devido principalmente ao fato de que os estudos e tecnologias em desenvolvimento de interface de usuário para celulares estarem muito mais evoluídos do que para televisores. Este cenário de evolução é fruto do longo tempo já transcorrido de estudos de interfaces de celulares. Outro fator que corrobora com a decisão de se traçar este paralelo é que ambas as classes de produtos possuem limitações nos dispositivos de entrada, com um teclado reduzido e a inexistência de *mouse*, carecendo de uma nova abordagem de navegabilidade da tela. Assim, como nos celulares, a navegação em televisores não pode se valer da utilização *mouse*, e deve se processar de forma tabular, onde a cada evento de clique no teclado um dado elemento visual recebe o foco.

2.10 CONCLUSÃO DO CAPÍTULO

Este capítulo buscou apresentar o ferramental teórico e técnico necessário para a realização desta dissertação, através da revisão de artigos e estudo de tecnologias referentes a automação residencial. Observa-se neste capítulo dois grandes blocos, um introdutório, mas voltado para o ferramental teórico, e um segundo mas voltado para o ferramental técnico.

O bloco introdutório buscou apresentar os principais conceitos de automação residencial e predial, através da releitura de artigos abordando tópicos como conforto, saúde e segurança. Foi apresentado também neste capítulo alguns trabalhos correlatos que serviram de base para a idealização da arquitetura de solução adotada nesta dissertação, como o PAUL e o *gateway* Konnex WebSevice. Esta releitura auxiliou na compreensão do que vêm a ser automação residencial e predial, seus conceitos e seu papel na sociedade. Os trabalhos correlatos serviram como parâmetros na solução proposta, tendo até algumas das características adotadas na elaboração desta dissertação.

Seguindo na revisão da literatura e do estado da arte, foram apresentados os principais protocolos de automação predial, com foco no BACnet, tema desta dissertação. Também foram apresentados os principais meios físicos e enlace de dados, abordando assim as principais camadas da pilha de referência OSI para a automação residencial

e predial. A revisão dos protocolos, principalmente do BACnet, foram determinantes na estratégia de integração de protocolos do *gateway* BACnet/HTTP proposto, com a definição da integração na camada de aplicativo. Quanto ao estudo dos meios físicos e camadas de enlace, este foi imprescindível na compreensão de alguns "fenômenos" ocorridos durante a fase de testes, como por exemplo a perda e repetição dos pacotes ao se tentar colocar o RS485 sobre o Zigbee.

Finalizando o bloco introdutório da revisão bibliográfica, foi apresentado os principais conceitos de TV digital, através da explanação dos principais padrões e da abstração proposta pelos *middlewares* de TV digital. Foi apresentado também algumas soluções de integração de TV digital com automação residencial, que auxiliaram nas decisões de projeto da solução proposta. Os conceitos de TV digital foram importantes para a definição da arquitetura de software adotada, buscando um código enxuto e não vinculado a linguagem C, buscando facilitar a portabilidade deste código para o *middlewares* de TV digital assim que este for consolidado. Esta decisão se expressa, além da estruturação do código em proto-objetos, a definição de tipos como o String, muito comum ao Java, e que é a linguagem que está sendo adotada nos *middlewares* brasileiro e europeu.

Após este bloco introdutório, foi realizado um estudo detalhado dos *hardwares* e *softwares* utilizados e referenciados nesta dissertação, como por exemplo os microcontroladores, SoCs, ferramentas de desenvolvimento e padrões de codificação e metodologias de decisão. O estudo dos *hardwares* foram imprescindíveis tanto para a compreensão das capacidades de cada componente adotado como na escolha destes componentes para a elaboração dos subprodutos desta dissertação, como por exemplo será o caso do AVR, que será apresentado como solução para a confecção de um dispositivo BACnet de baixo custo. Quanto ao estudo dos *softwares*, estes estão permeados dentro de todo o desenvolvimento, além de algumas decisões, como a adoção de algumas filosofias do Strutus e do Velocity no desenvolvimento de ferramentais de software, chamados no capítulo 3 de uStruts e uVelocity.

3 DESENVOLVIMENTO

Para o desenvolvimento do *gateway* BACnet/HTTP se fez necessário a realização de diversas atividades que geraram subprodutos desta dissertação. Estas etapas foram divididas da seguinte forma:

1. *Estudo de soluções consolidadas no mercado*: esta etapa trata do *hardware* que veio a servir de ambiente de desenvolvimento do aplicativo.

2. *Desenvolvimento de um navegador de aplicação específica baseado em imagens*: esta etapa trata do desenvolvimento do aplicativo que deverá servir de interface visual entre o usuário e o *gateway* BACnet/HTTP, sendo o aplicativo cliente deste.

3. *Desenvolvimento de um gateway BACnet/HTTP*: Esta etapa trata do desenvolvimento de um *gateway* de camada de aplicação voltado para a interligação de redes BACnet e HTTP. Pela amplitude do desenvolvimento realizado, optou-se pela divisão desta fazes em duas subfases, sendo elas:

   (a) *Aspectos de implementação do gateway BACnet/HTTP*: esta etapa trata dos macro-aspectos das implementações da implementação do *gateway* BACnet/HTTP e da interpretação de alguns tópicos referentes a implementação.

   (b) *Descrição do aplicativo desenvolvido*: esta etapa trata do desenvolvimento do código do *gateway*, numa abordagem com diferentes níveis de abstração, até a apresentação de algoritmos das principais funcionalidades.

4. *Desenvolvimento de um dispositivo BACnet de referência*: esta etapa trata do desenvolvimento de um dispositivo BACnet em um microcontrolador AVR (vide seção 2.7.4.2), para servir de referência a novas implementações.

O STB utilizado como plataforma de desenvolvimento consiste em um receptor de TV digital do padrão brasileiro (vide seção 2.5.2.4), sem *middleware* de TV digital (vide seção 2.5.5). Esta escolha de *hardware* restringe a decisão de onde deverá ser implementado o aplicativo de automação residencial, cabendo apenas a implementação

diretamente sobre o sistema operacional, que neste receptor é o STLinux (vide seção 2.8.1). Este *hardware* possui como peça principal um SOC (vide seção 2.7.1) ST7100 (vide a seção 2.7.5.1). Para realizar as alterações no *firmware* foram estudadas algumas soluções de mercado e as plataformas de desenvolvimento disponíveis. As ferramentas de software utilizadas para a geração do novo *firmware* foram o compilador GCC (vide seção 2.8.2) e a IDE Eclipse (vide seção 2.8.3).

3.1 ESTUDO DE SOLUÇÕES CONSOLIDADAS NO MERCADO

Devido o fato de se ter desejado que o *gateway* desenvolvido nesta dissertação pudesse ser utilizado em STBs comercializados no mercado brasileiro, foi realizado um estudo das soluções de STB comercializadas. Foi escolhido um grupo de STBs em que o centro da solução fosse o CI ST7100 (vide seção 2.7.5.1), a fim de que os programas compilados neste trabalho pudessem ser utilizados sem a necessidade de uma nova etapa de compilação. Nesta fase foram escolhidos 3 STBs comerciais, sendo eles o (a) DigiTV positivo da Positivo (POSITIVO, 2010), o XPS1000 da Proview (PROVIEW, 2010) e o DTH1900 da Century do Brasil (BRASIL, 2010). Para realizar o estudo destas soluções de uma forma mais sistemática, foi definido uma metodologia com base nos procedimentos propostos pelo pesquisador Ian McLoughlin em (MCLOUGHLIN, 2008). A metodologia proposta está baseada em três atividades, sendo elas: (a) escolha de produto, (b) conhecimento do produto e (c) conhecimento dos desenvolvedores do produto. Estas atividades são descritas a seguir:

3.1.1 Escolha do produto:

A escolha dos STBs comerciais partiu da necessidade de ser baseada no CI ST7100, devido a existência da plataforma de desenvolvimento e ao reaproveitamento dos programas compilados. Foram observados também os produtos que sofreram grande pressão para entrar no mercado, visto que tal postura leva mecanismo de atualização que torna viável a inserção dos código fontes (MCLOUGHLIN, 2008).

3.1.2 Conhecendo e analisando o produto:

Após a escolha dos produtos comerciais a serem utilizados, foi dado procedimento ao reconhecimento do sistema, bem como as peculiaridades do mesmo. Esta análise foi dividida em duas grandes atividades, onde uma é responsável pelo levantamento dos componentes e outra é responsável pelo levantamento das portas de acesso, tal como será descrito a seguir:

3.1.2.1 Geração do BOM (*Bill Off Material* lista de materiais):

Nesta atividade ocorreu o levantamento cuidadoso dos componentes, prestando principal atenção aos CIs utilizados. Dentre os CIs contidos em cada aparelho, os de maior tamanho geralmente são os mais importantes, como por exemplo micro-controladores, processadores, memórias, conversores ADCs e DACs. Muitos CIs utilizados em produtos comerciais não possuem *datasheets* abertos à utilização pública, carecendo de contratos com os fornecedores para a sua obtenção. Contudo, caso ocorra a escolha por um sistema com sistema operacional, o conhecimento profundo do *hardware* pode não se fazer necessário, visto as abstrações proporcionada pelo SO. Alguns CIs possuem dissipadores que dificultam a sua remoção, e para tanto foi recorrido também a fóruns de alteração em *hardware* para a complementação das listagens de materiais.

3.1.2.2 Identificação das portas de entrada e saída:

Na metodologia proposta foram categorizadas as portas de entrada e saída do sistema nos seguintes grupos: (a)portas oficiais de usuário, (b) portas oficiais de manutenção e (c) portas extra-oficiais ou de desenvolvimento.

1. *Portas oficiais de usuário*: compreende as portas comumente documentadas e oferecidas ao usuário, sendo algumas destas as atualização de *firmware*, saídas de vídeo, entradas de antena, entrada de *joystick*, dentre outros. Devido à necessidade de se por no mercado um produto e manter a qualidade alta do mesmo frente à concorrência, muitos produtos comerciais, dentre eles todos os STBs analisados, possuem uma porta de atualização de *firmware*. Essa porta é particularmente

importante, pois a existência desta porta torna possível a inserção de arquivos binários a serem executados dentro do produto comercial.

2. *Portas oficiais de manutenção*: são as portas definidas para a utilização dos profissionais no momento da manutenção do sistema. Estas portas geralmente estão documentadas nos diagramas esquemáticos dos aparelhos, que na sua maioria das vezes só são fornecidos a empresas de assistência técnica autorizada. Este grupo de portas geralmente se alteram de acordo com o fabricante e o modelo do equipamento, mas geralmente são portas de verificação de sinal, portas seriais e conectores de equipamentos proprietários.

3. *Portas extra-oficiais ou de desenvolvimento*: são aquelas não documentadas e comumente utilizadas nas fazes que antecedam a produção em larga escala, que foram mantidas de alguma forma,seja por causa da dificuldade de confecção do leiaute das placas, seja por mera distração. Estas portas geralmente não contém os conectores, somente as pontos de soldagem. Algumas vezes não possuem alguns componentes como, por exemplo, resistores e CIs de acoplamento.

3.1.3 Conhecimento dos desenvolvedores do produto

Segundo Sun Tzu em (TZU, 2010): *"Se você se conhece, você tem metade da guerra ganha, se conhece o seu inimigo, tem metade, se você conhece os dois terá a guerra ganha."* Estas palavras mostram a postura a ser adotada na abordagem dos sistemas a serem analisados, onde se optou pelo levantamento da aptidão natural das empresas que desenvolveram os produtos e os indicativos na imprensa ou nas atualizações de *software*, sobre os nomes dos responsáveis. Este estudo preliminar das pessoas, sejam físicas ou jurídicas, visou identificar as decisões de projeto e o modo de pensar que nortearam o desenvolvimento do produto.

3.2 PROJETO DO NAVEGADOR *X*

Neste tópico será apresentado as fazes de desenvolvimento de um navegador específico orientado a imagem, nomeado de navegador X. Serão definidos os elementos visuais a serem apresentados, o mecanismo de navegação pelo qual o usuário irá interagir com o navegador, bem como será definido o mecanismo de apresentação gráfica. Será dado

uma atenção especial ao padrão de cores adotado, tanto para o armazenamento de imagens quanto o definido pelo STB, com o algoritmo de conversão de um padrão para outro, sendo também explanado o formato de arquivo adotado, tanto para o armazenamento de imagens quanto no STB.

3.2.1 Definição dos elementos visuais a serem apresentados na tela do navegador.

A definição do tamanho mínimo dos elementos visuais deve ser analisada a partir do ângulo visual de cada elemento, baseado na distância média de utilização do equipamento. Para a obtenção destes valores, tomou-se como parâmetro um celular do tipo *smartphone* em relação a um aparelho receptor de TV. Estimou-se a utilização de um celular a uma distância média de 0.4m (vide figura 3.1) e, para o caso de uma televisão, estimou-se uma distância de média de 2m. Para a definição do tamanho dos elementos visuais, optou-se por calcular primeiramente a proporção entre os elementos visuais apresentados em um celular e os apresentados em uma televisão.

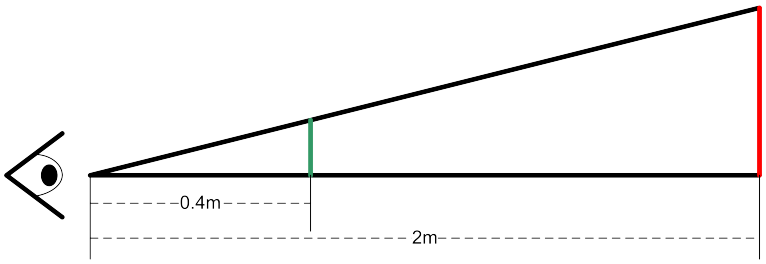

Figura 3.1: Cálculo da proporção entre elementos gráficos de um celular e de uma TV

O cálculo desta razão é apresentado na eq. 3.1 (HALLIDAY; RESNICK, 2010) onde para o caso do par de distancias 2m/0.4m o resultado é 5. Isso significa que, neste exemplo, um elemento visual apresentado no celular tem que ser 5 vezes maior do que em uma televisão, para que seja enxergado da mesma forma por um observador. Um ícone de $1cm^2$ apresentado em um celular teria 5cm de lado, ou seja, $25cm^2$ em um aparelho de TV.

$$Raso = \frac{D_1}{D_2} = \frac{Distncia_{TV}}{Distncia_{Celular}} = \frac{2m}{0.4m} = 5 \tag{3.1}$$

Nesta proporção, um *smartphone* do modelo Iphone que possui uma tela com 3.5" (IPHONE, 2010) observado a uma distância de 0.4m teria a mesma impressão visual para um usuário com um televisor de 17.5" a uma distância de 2m.

3.2.2 Definição do mecanismo de navegação do usuário

O mecanismo de navegação foi projetado como máquinas de estados finitos (*Finite State Machines* - FSMs), onde cada estado representa um dado ícone. Cada um desses estados é um *estado de seleção*, onde as operações realizadas pelo usuário, através do teclado, serão processadas apenas dentro do universo de possibilidades do dado ícone.

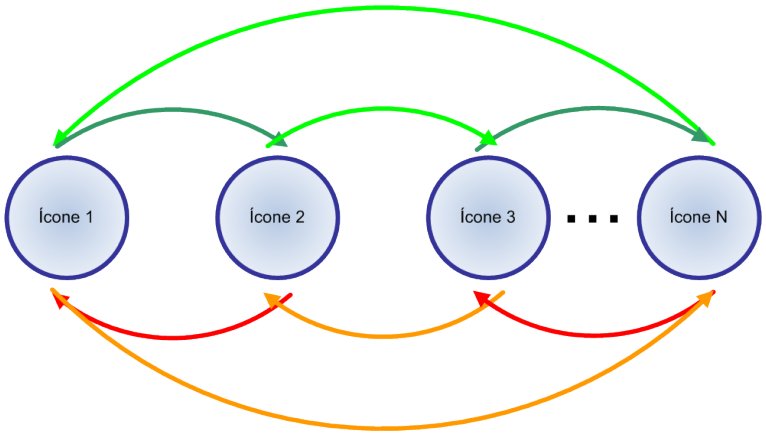

Figura 3.2: Máquina de estados de Navegação entre ícones

A figura 3.2 mostra as transições entre os ícones, a partir de um modelo de navegação com duas teclas, onde em um dado ícone pode se tomar dois sentidos, cada um referente a cada tecla. Cabe ressaltar que neste modelo, um usuário com o domínio da utilização de apenas uma tecla de direcionamento pode chegar a qualquer ícone. Esta observação visa atender as pessoas mais idosas e menos familiarizadas com a tecnologia, buscando atender as necessidades apresentadas na seção 2.2.2.

```
1  typedef void (*fnBotao)(unsigned int estado);
2  static unsigned int ESTADO ICONE DESABILITADO ⇐ 0 ;
3  static unsigned int ESTADO ICONE LIGADO ⇐ 1 ;
4  static unsigned int ESTADO ICONE DESLIGADO ⇐ 2 ;
5  typedef struct
6      RGBImage imgDesabilitado;
7      RGBImage imgLigado;
8      RGBImage imgDesligado;
9      unsigned int estado;
10     fnBotao funcao;
11 Botao;
```

Listagem 1: Estrutura do objeto Botão

Cada ícone possui sub-estados que podem ser divididos em dois grandes grupos: (a) *habilitado* e (b) *desabilitado* (vide listagem 1 linha 2). O grupo de estados habilitados tem seus estados definidos de acordo com o dispositivo físico de automação que o ícone representa. Para o caso do elemento *BACnet DigitalOutput*, apresentado no capítulo 2, página 27, se pode ter 2 estados possíveis: *ligado*(vide listagem 1 linha 3) e *desligado* (vide listagem 1 linha 4) (vide figura 3.3), desconsiderando-se o estado *desconhecido* ou *desabilitado*. Na estrutura do objeto *Botão* apresentado na listagem 1 foi colocado também uma imagem para cada estado possível (linhas 6,7,8), a fim de se ter uma identificação visual de cada estado.

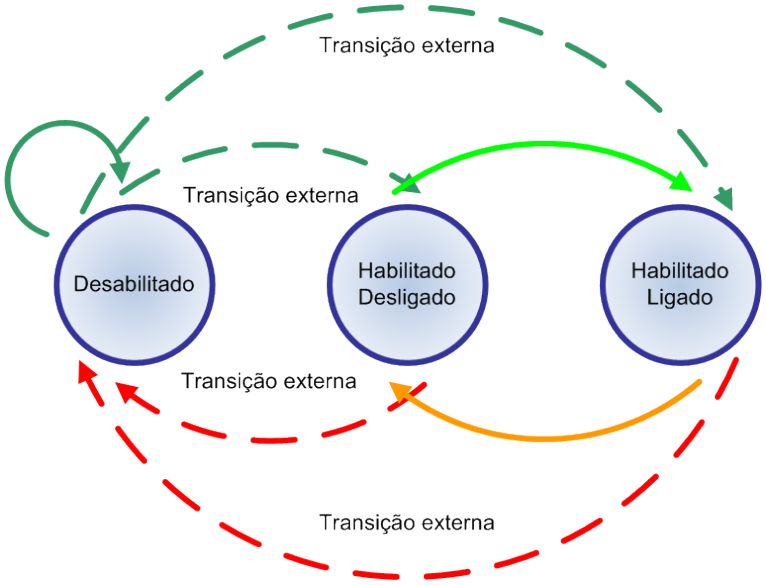

Figura 3.3: Máquina de estados interna do ícone representando o BACnetDigitalOutput

O estado *desabilitado* foi projetado para ser mantido independente da operação do usuário (vide figura 3.3 e listagem 2 linhas 4 e 5), sendo que este estado só é transitado caso ocorra um evento externo, como, por exemplo, um dispositivo é descoberto na rede BACnet (passando do estado *desconhecido* para o estado no qual o dispositivo foi encontrado). Também um dispositivo pode passar para o estado *desabilitado*, caso se perca a comunicação com o dado dispositivo. Na listagem 2 é apresentado o algoritmo do mecanismo das funções do *Botão* para o caso do elemento *BACnet DigitalOutput*, onde ocorre a transição do estado *ligado* para o *desligado* (linhas 6 e 7) e vice-versa (linhas 8 e 9) e é executada a funcionalidade do novo estado (linha 12).

A operação a ser realizada por ocasião da passagem para o dado estado é realizado pela chamada da funcionalidade descrita na listagem 2 linha 12. Estas chamadas são descritas de uma forma mais detalhada na listagem 3, onde se realiza o direcionamento da funcionalidade de cada estado. Diferentemente do que ocorre no mecanismo dos botões, o algoritmo apresentado na listagem 3 não executa a transição de estados, mas

```
   Entrada: Botao *botão
   Saída: Sem retorno
1  estado ⇐ estado do botão
2  funcao ⇐ função do botão
3  selecione estado faça
4      caso ESTADO ICONE DESABILITADO
5          estado ⇐ ESTADO ICONE DESABILITADO;
6      caso ESTADO ICONE LIGADO
7          estado ⇐ ESTADO ICONE DESLIGADO;
8      caso ESTADO ICONE DESLIGADO
9          estado ⇐ ESTADO ICONE LIGADO;
10     senão
11         ...
12 funcao(estado);
13 estado do botão ⇐ estado;
```

Listagem 2: Mecanismo da máquina de estados das funções do botão

sim realiza a função definida para cada estado.

```
   Entrada: estado
   Saída: Sem retorno
1  se estado == ESTADO ICONE LIGADO então
2      Realizar operações de ligar Dispositivo;
3  senão se estado == ESTADO ICONE DESLIGADO então
4      Realizar operações de desligar Dispositivo;
5  senão se estado == ESTADO ICONE DESABILITADO então
6      Realizar operações ao se ter a Dispositivo desabilitado;
```

Listagem 3: Função do botão do dispositivo

Na figura 3.3, as transições realizadas pelo usuário foram desenhadas com linha cheia e as transições realizadas de forma alheia ao usuário do navegador (i.e, falhas de rede, ação de agentes da rede, usuários externos) foram marcados com linhas tracejadas.

3.2.3 Definição do mecanismo de apresentação gráfica.

A apresentação de elementos gráficos em dispositivos com sistema operacional pode se dar de diferentes formas, sendo as mais utilizadas o *framebuffer* e o OpenGL. O OpenGL (OPENGL, 2011) é utilizado quando se tem *hardware* dedicado a renderização de objetos 2D ou 3D, e acesso via API a estas funcionalidades. Já o *framebuffer* é o mais simples e de mais fácil utilização no nível de *hardware*.

Para o desenvolvimento do navegador optou-se o desenvolvimento de um *driver* de acesso ao *framebuffer*, com funções voltadas para o desempenho e facilidade de utilização na geração de elementos visuais. No Linux o *framebuffer* é mapeado em um dispositivo que, como qualquer outro, é um arquivo que deve ser operado para se enviar ou receber informação do mesmo. Este dispositivo foi mapeado em uma região de memória que facilita as operações de escrita não seqüencial, o que é pretendido para a renderização de apenas uma parte da tela, objetivado se ter melhor desempenho.

O sistema de coordenadas da tela se inicia no canto superior esquerdo, crescendo para baixo e para a direita, onde na horizontal se tem o *eixo x* e na vertical o *eixo y* (vide figura 3.4).

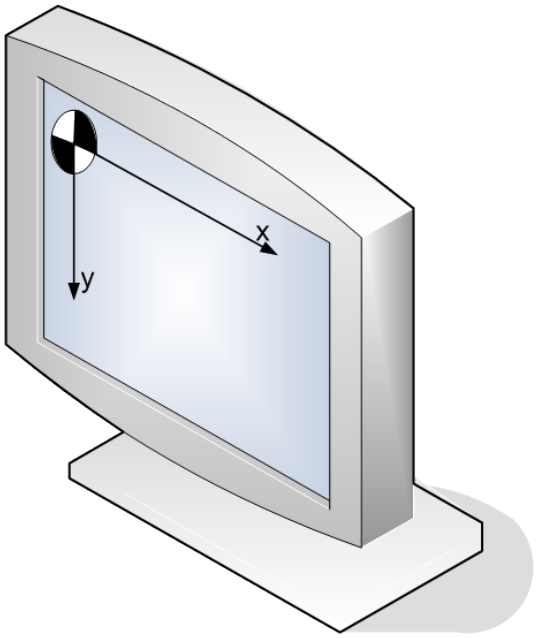

Figura 3.4: Sistemas de coordenadas adotado nos monitores e televisores

3.2.4 Padrões de cor

A expressão de cada pixel de cor deve seguir um padrão de cor. Dentre os padrões de cor atualmente utilizados para apresentação de vídeo se tem os padrões de 16 bits e os 32 bits.

3.2.4.1 Padrões de 16Bits

- *YCrCb:* é baseado no antigo padrão imposto pela evolução da televisão preto-e-branco para colorida (vide seção 2.5), o qual é baseado em um canal de *luminância* (Y) que representa a imagem em (preto-e-branco) e dois canais de *cromância* (Cr e Cb), sendo as *cromâncias* vermelha e azul. Neste padrão, 8 bits são alocados para o canal Y, visto que o olho humano é mais sensível a variações de claro e escuro. Por outro lado, 4 bits são disponibilizados para cada um dos canais de cor.
- *RGB565:* esta é um padrão baseado nos bastonetes, células responsáveis pela captação de cor no olho humano, que possuem 3 tipos, cada um baseado em uma das cores (R= vermelho, G=verde, B = azul). O canal G tem 6 bits, pois o olho humano é mais sensível a luz verde, provavelmente devido a fatores de evolução humana.

3.2.4.2 Padrões de 32Bits

Os padrões de codificação de cor em 32 bits geralmente codificam as cores em tons de vermelho, verde e azul, onde cada canal de cor é representado por 8 bits. Dentre as formas de codificação também tem especial importância a ordem de codificação da cor, geralmente representada na ordem RGB (vermelho, verde e azul), porém também se pode ter um padrão com um *swap* das cores, sendo representado, por exemplo, pelo padrão BRG. A conversão do padrão RGB32 para RGB565 se dá da forma apresentado na figura 3.5 e na listagem 4:

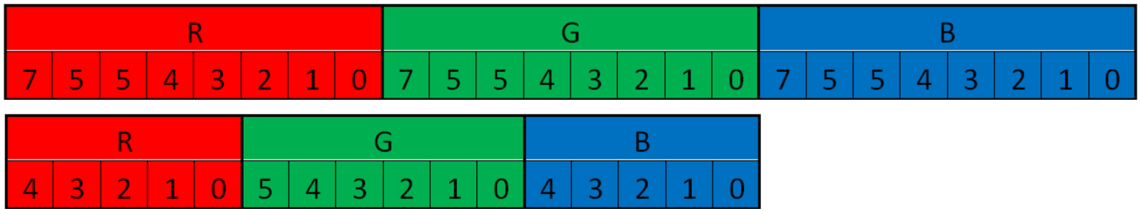

Figura 3.5: Conversão de RGB32 para RGB565

Entrada: Pixel (3Bytes) em RGB32

Saída: 2Bytes RGB565

1 $Sada =$

2 $R \Leftarrow (8bits)$ e aplicação da máscara $0xf800$ $\otimes$G $\Leftarrow (3bits)$ e aplicação da máscara $0x07e0$ $\otimes$B $\Rightarrow (3bits)$.

Listagem 4: Conversão de RGB32 para RGB565

3.2.5 Formato de arquivo das imagens

Um dos elementos visuais mais necessários a qualquer navegador é a apresentação de imagens. Existem alguns navegadores baseados em caracteres, que não apresentam imagens de forma alguma como, por exemplo, o *lynx* (LYNX, 2010) que é executado no console de sistemas Unix; porém a sua interface é menos amigável e voltada a um propósito de prover acesso a páginas de Internet através de um console. Contudo, um navegador que apresenta somente imagens vem a ser mais amigável, tal como ocorre com o OperaMini (OPERAMINI, 2010) que é voltado para dispositivos portáteis. O OperaMini renderiza a página como uma imagem em um servidor, "tirando uma foto" da página (vide figura 3.6), exibindo uma parte desta na tela, permitindo assim um melhor aproveitamento da rede e da qualidade de renderização, que é feita em um servidor PC externo. Assim, como se optou pela qualidade da apresentação visual, tendo em vista também o número reduzido e definido de ícones, foi considerada como elemento principal de renderização a imagem.

Existem diversos formatos de arquivo de imagem, que podem ser divididos de acordo com o seu nível e formato de compactação. Um primeiro grupo seria os dos formatos descompactados, chamado de *raw* onde não há qualquer compactação. Neste grupo se encontra o RAW, o BMP, o PPM, dentre outros. Já os formatos que apresentam compactação são aqueles no qual se abre mão de alguma informação para aperfeiçoar o transporte de informação. Do grupo dos formatos de imagem com compactação temos como melhores exemplo o formato GIF que é voltado para imagens com um universo de cores reduzido (gravuras) e o JPG que pode representar qualquer profundidade de

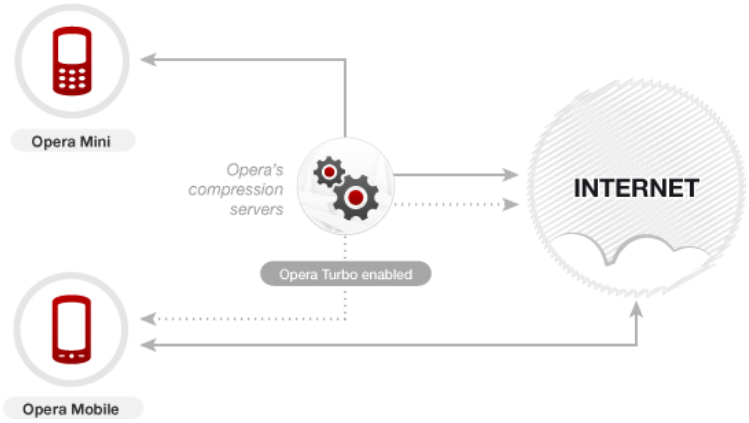

Figura 3.6: Diagrama informativo do Opera Mini e Opera Mobile. Fonte: (OPERAMINI, 2010)-página specs

cor possui um algoritmo mais sofisticado de compactação, no qual se pode definir o nível de compactação com perdas.

Para a definição do padrão de imagem a ser utilizado foi tomado às seguintes premissas:

- Ser um padrão popular, não criando um "padrão" próprio.
- Ser um padrão para o qual existam conversores e editores para este formato de cores;
- Ser um formato de fácil interpretação, não exigindo grande poder computacional do hardware que executa o navegador.

Para estes propósito foi escolhido o padrão PPM binário, que é um padrão sem compactação, com um cabeçalho que identifica o padrão, adiciona comentários e informa a profundidade de cores e as dimensões x e y da imagem. Contudo, caso se deseje o suporte para outros formatos de imagem, pode-se utilizar das várias bibliotecas de *software*, como por exemplo a libjpeg (IJG, 2010).

A definição do arquivo PPM binário (PPM, 2010) é apresentado na listagem 5. Inicialmente se tem uma sequência de caracteres mágicos, o "P6" indicando que o arquivo é do formato PPP Binário, depois comentários precedidos do caractere "sustenido", seguido das dimensões x e y no formato ASCII com a profundidade de cor para cada bit, limitada desde Abril de 2000 a 255 (PPM, 2010). Por fim, uma sequência de dados binário no formato RGB contendo a informação da imagem.

```
1 P6
2 Comentários
3 Mais comentários
4 Dimensão_X Dimenso_Y
5 Profundidade de Cor
6 Dados binários R G B ...
```

Listagem 5: Formato de arquivo PPM Binário

Para guardar a informação em memória, foi definido estrutura de dados definido na listagem 6. A primeira estrutura (RGBPixel) é responsável por armazenar um pixel no formato RGB32. Já a estrutura RGBImage guarda a imagem como um todo, agregando a informação de translação da origem da imagem (int *x*, *y*) e as suas dimensões de largura e altura (*width, height*).

```
1 typedef struct
2     unsigned char red, green, blue;
3 RGBPixel;
4 typedef struct
5     int x, y;
6     int width, height;
7     RGBPixel *data;
8 RGBImage;
```

Listagem 6: Estrutura do objeto de imagem

3.2.6 Aspectos da implementação do navegador *X*

Segue um descritivo das funções mais importantes dos navegador *X*, descrevendo principalmente o driver de *FrameBuffer*. Esta implementação é baseada nas definições do arquivo de imagem apresentado na listagem 5 e na definição do objeto imagem, apresentado na listagem 6. Estas funções são as seguintes:

- *drawPixel:* desenha um pixel RGB nas coordenadas definido. Para implementar a transparência de 1 bit, foi considerado a cor preta (R+G+B=0) como a transparência. Este método necessita conhecer como é definido o *framebuffer*, como por exemplo o padrão de cores adotado (vide listagem 7);

```
Entrada: FBPixel pixel, int x, int y
Saída: Sem retorno
1 long int location ⇐
2 (x+vinfo.xoffset) * (vinfo.bitsPerPixel/8) +
3 (y+vinfo.yoffset) * finfo.lineLength;
4 *(fbp + location + 0) ⇐ pixel.blue;
5 *(fbp + location + 1) ⇐ pixel.green;
6 *(fbp + location + 2) ⇐ pixel.red;
```

Listagem 7: Função que desenha um pixel na posição definida

```
Entrada: RGBImage *img, int xOffset, int yOffset, int wOffset, int hOffset
Saída: Sem retorno
1 width ⇐ img.width;
2 height ⇐ img.height;
3 x ⇐ img.x;
4 y ⇐ img.y;
5 para j ⇐ yOffset até j < (yOffset+hOffset)  j < height faça
6     para i ⇐ xOffset até i < (wOffset+xOffset)  i < width faça
7         pixel ⇐ img.data[width*j + i];
8         drawPixel(pixel,x + i,y + j);
```

Listagem 8: Função que desenha uma porção da imagem na posição definida

- *drawSubImage:* desenha uma porção da imagem, sendo útil para re-pintar uma imagem após como limpeza da tela (vide listagem 8);
- *drawImage:* desenha uma imagem na tela, se utilizando dos métodos anteriores (vide listagem 9);
- *initFramebuffer:* inicializa o *framebuffer*, reservando o dispositivo, mapeando este em memória, obtendo as informações do dispositivo;
- *finalizeFramebuffer:* finaliza o dispositivo, liberando o recurso e as regiões de memória.

```
Entrada: image *img
Saída: Sem retorno
1 FBPixel pixel;
2 width ⇐ img.width;
3 height ⇐ img.height;
4 x ⇐ img.x;
5 y ⇐ img.y;
6 para j ⇐ 0 até height faça
7     para  i ⇐ 0 até width faça
8         pixel ⇐ img.raw[width*j + i];
9         drawPixel(pixel,x + i,y + j);
```

Listagem 9: Função que desenha uma imagem na posição definida

3.3 ASPECTOS DE IMPLEMENTAÇÃO DO *GATEWAY* BACnet/HTTP

A implementação do *gateway* BACnet/HTTP, bem como o navegador *X*, foram implementados em um computador pessoal, porém serão executadas em um STB comercial. A implementação pode ser dividida em duas grandes etapas: (a) definição da estratégia de integração e (b) implementação do código em si.

3.3.1 Definição de estratégia de integração

O BACnet como protocolo está definido nas camadas de rede e aplicativo com base na pilha de referência OSI. Um *gateway* é um conversor de protocolos que pode ser implementado em qualquer um dos sete níveis da pilha OSI. Caso seja possível implementar um *gateway* em um nível mais baixo, melhor será o desempenho do *gateway*, porém se tem uma solução muito acoplada e pouco abstrata e mais difícil de ser desenvolvida. Adotou-se para esta dissertação o desenvolvimento de um *gateway* de camada 7, ou um *gateway* de camada de aplicativo, por restrição natural do protocolo BACnet (vide seção 2.3.1.4, página 27), que está definido até a camada de aplicativo, além de se ter uma maior facilidade de desenvolvimento, devido ao maior grau de abstração. Obteve-se com esta estratégia uma maior facilidade de reuso e expansão do *gateway*, bem como a possibilidade de implantá-lo em diferentes arquiteturas. O foco desta seção é a definição das diretrizes de criação desse *gateway* e da forma de ligação das diversas pilhas.

Neste contexto, para o desenvolvimento do *gateway* de BACnet IP/HTTP adotou-se a camada de aplicativo como o suporte para o *gateway*, utilizando uma camada de código como "cola" entre o BACnet/IP e o HTTP.

Para o desenvolvimento da etapa BACnet do *gateway* foi utilizada como pilha de BACnet a implementação do BACnet Stack (BACNET, 2009), que se mostrou uma das mais completas do SourceForge (GEEKNET, 2010) no segmento, com uma comunidade muito ativa e receptiva a novos desenvolvedores, respondendo prontamente aos questionamentos feitos.

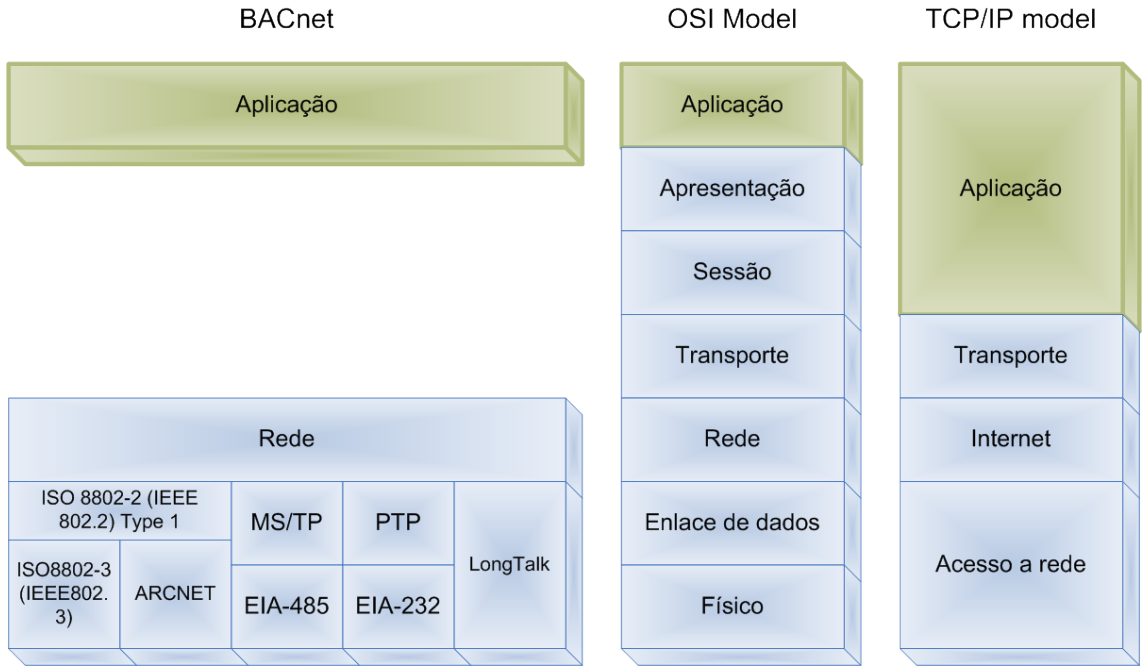

Figura 3.7: Comparativo das pilhas BACnet e TCP/IP com a referência OSI. Fonte:(ASHRAE135, 1995), modificado

Os STBs possuem um navegador que provê uma interface com o usuário das funcionalidades do STB, tais como mudança de canal, ajuste de cor, programação e restrição de canais. Contudo, com o *gateway* BACnet sendo executado dentro do STB, os desenvolvedores do navegador do STB poderão agregar as funcionalidades de acesso ao barramento de automação residencial, seja por BACnet/IP ou BACnet/MSTP ou Zigbee.

O *gateway* foi definido para poder ser composto de uma pilha BACnet compilada em dois modos, uma voltada para o padrão BACnet/IP e outra para o BACnet/MSTP. Por outro lado, se tem a implementação de duas implementações distintas de acesso ao HTTP (vide figura 3.8). Uma é a utilização do AXIS2, um servidor de *webservice* de boa receptividade no mercado. Outra é um servidor HTTP 1.0, desenvolvido especialmente

para aplicações que necessitem de uma implementação mais enxuta. Este servidor visa uma melhor utilização de espaço em memória e a possibilidade de comunicação de clientes com implementações mais simplificadas, visto que estes clientes não precisariam implementar um cliente *webservice* e sim apenas um cliente HTTP 1.0.

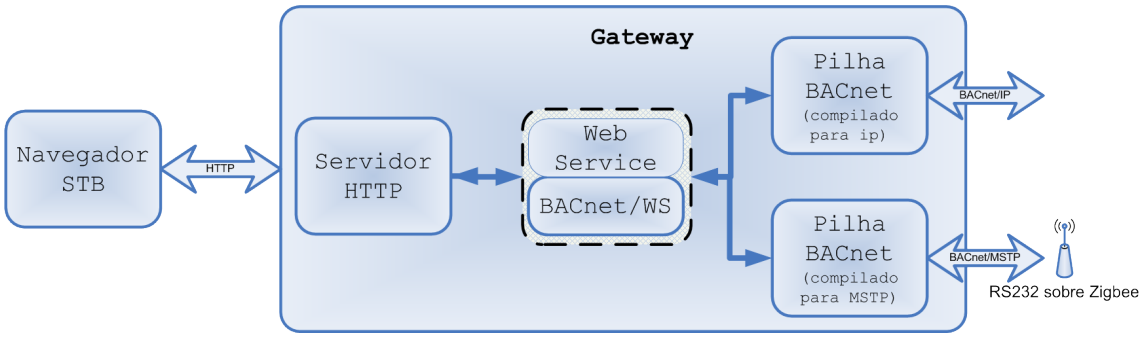

Figura 3.8: Arquitetura do *gateway*

3.3.2 Implementação do código

Para a implementação do código foi buscado a facilidade de compreensão da implementação e a aplicação de abstração para que novas funcionalidades sejam acrescidas ao *gateway*, sem a necessidade da compreensão da solução como um todo. Para alcançar este objetivo, o *gateway* foi dividido em 4 grandes módulos, descritos a seguir:

- HTTP-Server;
- BACnet-Controler;
- Web-Framework;
- Util-Fucntions.

Esta abordagem torna possível a manutenção do *gateway* e o acréscimo de funcionalidades mais fácil e a solução mais desacoplada, visto que as funcionalidades de um módulo estão todas agrupadas dentro do mesmo módulo, podendo operar de forma independente dos outros módulos.

3.4 DESCRIÇÃO DOS *GATEWAYS* DESENVOLVIDOS

Nesta seção serão apresentadas as principais partes dos *gateway* desenvolvido. Como metodologia da apresentação desta etapa, foi adotado uma abordagem em níveis de abstração, onde os cada nível foi nomeado em termos da aviação, ou seja, na unidade pés, partindo de um nível de abstração mais alto (nomeado de 100.000 pés) até um nível mais baixo (100 pés). Nesta abordagem, é apresentado simultaneamente características funcionais e técnicas, buscando a cada passagem para um nível mais baixo, o aumento do entendimento da solução, desde uma compreensão geral da solução até os detalhes das implementações de diversas funções.

3.4.1 100K

Foi adotada como estratégia de integração a separação lógica do *gateway* em módulos. Tal abordagem facilita tanto a codificação quanto a depuração do código, sendo que cada uma destas atividades (codificação e depuração) podem ser realizadas somente no dado módulo, sem sofrer interferência de problemas relacionados com outros módulos.

Outro aspecto positivo da abordagem da separação do código em módulos é que a codificação está voltada para um forte desacoplamento das diversas partes que compõem o *gateway*, o que vem facilitar tanto a compreensão do código por futuros desenvolvedores quanto a reutilização dos módulos para novas soluções que venham a ser projetadas e que se desejam funcionalidades parecidas.

A parte em *software* que define o *gateway* foi concebida em quatro grandes módulos, tal como apresentado na figura 3.9. As mesmas serão descritas a seguir:

3.4.1.1 Servidor HTTP (HTTP-Server)

Este módulo representa a implementação de um servidor aderente a definição de servidor HTTP 1.0 (RFC1945, 1996) básico, capaz de disponibilizar quaisquer recursos solicitados por quaisquer clientes HTTP (i.e. Firefox, Internet Explorer). Este módulo está subdivido em dois módulos menores, a fim de facilitar a compreensão e a manu-

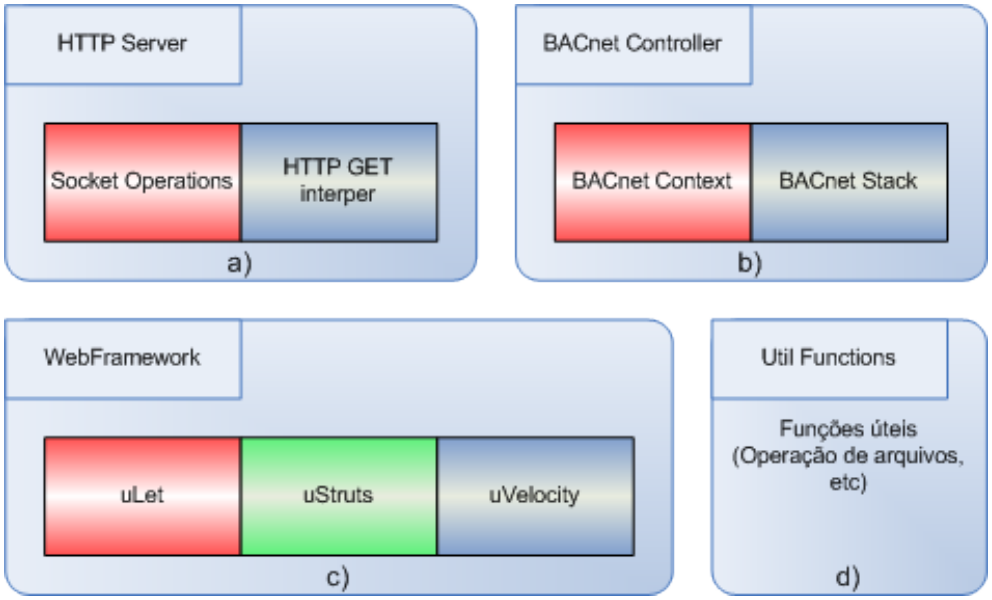

Figura 3.9: Implementação do *gateway* 100k pés. (a) módulo *HTTP-Server*, (b) módulo *BACnet-Controler*, (c) módulo *Web-Framework* e (d) módulo *Util functions*

tenção do código, sendo estes os módulos de *Operações com socket* e *Interpretador de funções HTTP*:

a) Operações com *socket* (*Socket-Operations*): realiza todas as operações necessárias com os *sockets* de comunicação com os clientes HTTP, tal como as funcionalidades de abrir um *socket*, responder uma dada solicitação e fechá-lo ao término da conexão.

b) Interpretador de funções HTTP: é um interpretador das funções definidas da (RFC1945, 1996) que definem o padrão HTTP 1.0. Para esta dissertação foi definido que seria implementado um interpretador de chamadas ao método GET, a fim de tornar o servidor mais simples. Esta abordagem não causa perda significativa de funcionalidade, visto que, por exemplo, o método POST acrescenta apenas a possibilidade do cliente HTTP enviar mais parâmetros e valores maiores, pois este método não se limita ao tamanho da URL. Segundo a RFC-3986 (RFC3986, 2005), o tamanho da URL é limitado, porém este limite é definido por cada implementação de servidor e cada implementação de cliente, sendo que a menor URL de cliente é de cerca de 2048 caracteres(Internet Explorer) (MICROSOFT, 2010).

3.4.1.2 Controlador-BACnet (*BACnet-Controler*):

o controlador BACnet é a parte do *gateway* voltada para a rede de automação predial e que é capaz de se comunicar através do protocolo BACnet apresentado na seção 2.3.1.4. O Controlador BACnet está para a rede BACnet assim como o Servidor HTTP está para a rede HTTP. O controlador BACnet está dividido logicamente em duas partes, que serão descritas a seguir (vide figura 3.9b):

a) Contexto-BACnet(*BACnet-Context*): contém as implementações em alto nível das funcionalidades da rede BACnet, provendo meios de acesso a esta rede, aos seus dispositivos e as propriedades destes dados dispositivos. O contexto-BACnet provê uma forma intuitiva de leitura e escrita de parâmetros em dispositivos, bem como a análise de status de erro e de aplicação de *reset* de um dado dispositivo, dentre outras funcionalidades.

b) Pilha-BACnet (*BACnet-stack*): é a parte do código que contém as funcionalidades de geração e interpretação dos APDUs e NPDUs, bem como prover o acesso ao meio físico adotado. Dentre as pilhas analisadas, foi eleita a (BACNET, 2009) e personalizada para que esta viesse a atender as funcionalidades necessárias ao *gateway*.

3.4.1.3 *Framework*-Web (*Web-Framework*)

Visando facilitar a implementação de novas soluções que objetivam utilizar o *gateway* como estratégia de acesso a rede BACnet, buscando que seus códigos se tornem portáveis e pequenos o suficiente para caber dentro de um sistema embarcado, foi desenvolvido um *framework* simplificado com base em filosofias e ferramentas comumente adotadas em projetos *web*, levando-se em conta projetos de servidores *web* em Java. Este *framework* provê uma camada de abstração do *gateway*, tornado possível o desenvolvimento de soluções de automação predial utilizando comunicação BACnet, sem a necessidade do conhecimento do protocolo em si. Devido ao fato de todas as funcionalidades terem sido implementadas em um contexto mais reduzido, e somente alguns fundamentos serem aplicados e não todo o padrão, todas as funcionalidades do *framework* foram nomeadas com o prefixo *u* (proveniente da letra grega μ=micro). Para tanto, foram desenvolvidas as seguintes funcionalidades:

a) *uLet*: baseada nas filosofias do *servlets* (com.javax.Servlet(SERVLET, 2010)) apresentada na seção 2.8.5, possui de entrada e saída equivalente aos objetos de entrada e saída de *HttpServlets*, também nomeadas de *request* e *response*, com um domínio de atributos muito parecido. Somente a função *doGet* (equivalente ao método *doGet* do *HttpServlet*) foi implementada, sem prejuízos relevantes a solução, visto que o servidor somente consegue interpretar métodos HTTP GET. Tal escolha é justificável visto o fato de que a maioria das implementações de *servlets* aponta o método doPost para o *doGet*.

b) *uStruts*: baseado no *framework* Struts (STRUTS, 2010b; STRUTS, 2010d; STRUTS, 2010c) apresentado na seção 2.8.7, prove uma solução de implementação do Padrão de Projeto (*design Pattern*) MVC (*Model View Control*), atuando na camada de controle. Na implementação do *gateway* deste *framework*, os dados (*model*) são montados em estruturas chamadas *actions* sendo que a parte de apresentação (*view*) é provida pelo *framework uVelocity*.

c) *uVelocity*: baseando no *framework* Velocity (VELOCITY, 2010) apresentado na seção 2.8.6, esta funcionalidade realiza a análise léxica de uma cadeia de caracteres (*string*) que pode tanto estar no sistema de arquivos do sistema operacional quanto dentro do próprio código, e então realiza a fusão (*merge*) dos dados passados em um contexto gerando no final uma nova cadeia de caracteres, que no caso de serviços *web* é um novo *recurso*, que pode ser, por exemplo, uma página HTML (vide figura 10).

O analisador léxico do uVelocity é apresentado na listagem 10 e vai varrendo todos os caracteres da entrada (linhas 1 e 2) em busca dos *tokens* definidos pelo Velocity. O *caso* apresentado nas linhas 3 até 8 implementam o processador de entradas do tipo "formal" do Velocity (VELOCITY, 2010), ou seja, aqueles que se encontram entre caracteres {} (chaves) precedidos do caractere $ (cifrão) (por exemplo ${nomeDispositivo}). Caso os caracteres encontrados não representem nenhum *token* válido, estes são acrescidos no texto de saída (vide listagem 10 linha 12), o que garante a inserção do restante do modelo de entrada, seja qual for a linguagem adotada.

O algoritmo adotado para o processamento da entrada formal do *framework* uVelocity é apresentado na listagem 11. Tomando como exemplo a entrada formal *${nomeDispositivo}*, ao entrar na função descrita na listagem 11, o valor de *i* refere-se à posição do

```
   Entrada: String entrada, String saida, ListaPar contexto
   Saída: Sem retorno
1  para i⇐0 até tamanhoDaEntrada faça
2      selecione caractere_Posicao[i] faça
3          caso $
4              se ehEntradaFormal(entrada, i) então
5                  linhasConsumidas ⇐ processaEntradaFormal(entrada, i, saida,
                   contexto);
6                  i+=linhasConsumidas;
7              senão
8                  acrescentarCaractere(saida,caracterePosicao[i]);
9          caso ...
10             ...
11         senão
12             acrescentarCaractere(saida,caracterePosicao[i]);
```

Listagem 10: uVelocityEngine

caractere *n*. O laço apresentado nas linhas 1 a 4 da listagem 11 destina-se a percorrer o texto até obter todo o conteúdo da variável do tipo *entrada formal*, que no exemplo apresentado é o texto *nomeDispositivo*. Na linha 5 é realizada a chamada da função que realizará a transformação da entrada formal em seu respectivo valor, caso este exista no contexto recebido como parâmetro. Esta atividade é melho descrita na listagem 12.

O algoritmo adotado para a transformação da entrada formal em seu respectivo valor é apresentado na listagem 12, que consiste num algoritmo de busca simples, tendo como condição de parada o encontro da chave (linha 3 da listagem 12) ou o fim do grupo de valores possíveis (linha 1). O parâmetro de entrada *contexto* nada mais é do que uma lista de *pares* de *chave* e *valor*. Por fim, para manter íntegro o modelo (*template*) recebido como entrada da *engine* do uVelocity, independente do conteúdo deste modelo, foi adotado a postura de retornar o próprio valor recebido (vide linha 4), caso não se encontre nenhuma chave igual a *função* passada como parâmetro de entrada.

```
Entrada: String entrada, int i, String saida, ListaPar contexto
Saída: Número de caracteres processados
Dados: int acc; Acumulador de caracteres processados
1 enquanto caractere Posicao[i] faça
2     acrescentarCaractere(tmp,caractere Posicao[i]);
3     acc++;
4     i++;
5 saida+=parseValor(tmp, contexto);
6 retorna acc
```

Listagem 11: Processador de entrada formal(processaEntradaFormal)

```
Entrada: String função, ListaPar contexto
Saída: String, contendo o valor da tradução
1 enquanto houver contexto a processar faça
2     se função = chave do contexto então
3         retorna valor da contexto para a dada chave;
4 retorna função;
```

Listagem 12: parseValor

3.4.1.4 Funções úteis (*Util functions*)

As funções descritas aqui são o cerne do *berço de código* da aplicação. Este módulo provê funcionalidades que podem ser aproveitadas por todos os outros módulos do sistema.

3.4.2 10K

Para a integração dos módulos do *gateway* foi utilizada a arquitetura apresentada na figura 3.10. Quando o *gateway* é iniciado, este abre duas linhas de execuções (*treads*) onde em uma se inicia o Controlador BACnet e na outra o servidor *web*.

O *controlador BACnet* realiza uma varredura (*scan*) da rede BACnet, identificando todos os dispositivos e suas respectivas propriedades e valores, montando assim uma estrutura de dados que possui o contexto da rede BACnet.

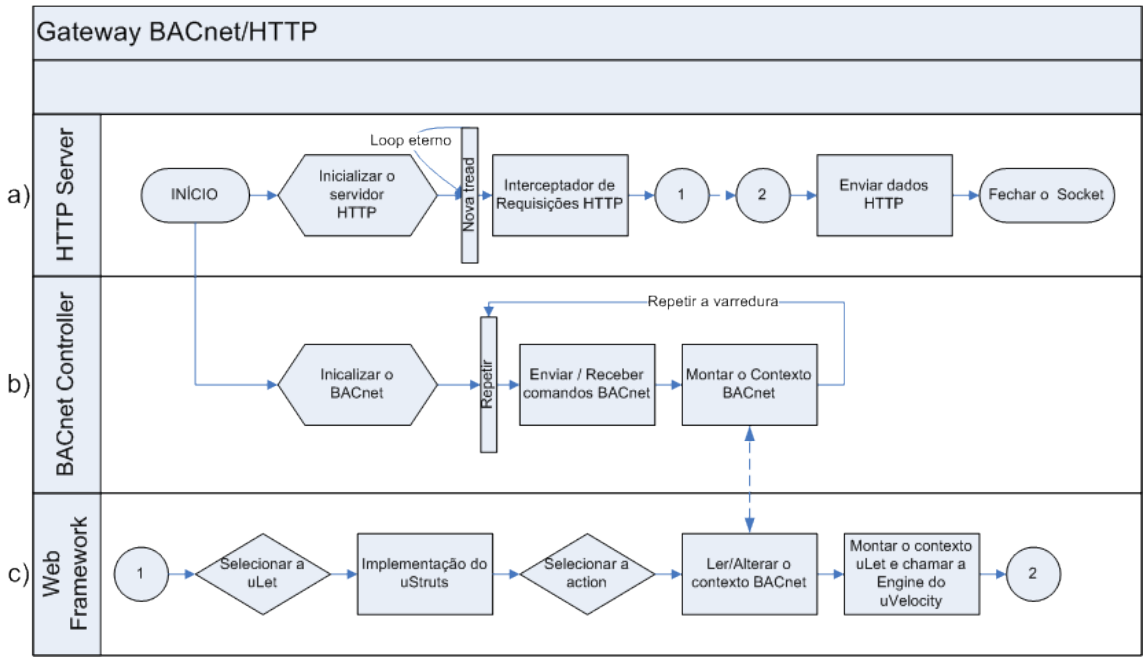

Figura 3.10: Implementação do *gateway* 10000 pés. a) HTTP Server, b) BACnet Controller, c) Web Framework

O *servidor web* espera as requisições HTTP oriundas dos clientes e, quando estas solicitações ocorrem, abre-se um fluxo de execução (*tread*) para atender a cada uma destas solicitações. Esta requisição é tratada pelo *framework web*, que devolve o resultado do processamento para o servidor *web*, a fim de que este envie o resultado ao cliente, finalizando o *socket* de comunicação HTTP.

Esta arquitetura limita a solução do *gateway* a sistemas que suportem o multiprocessamento (*multitreading*), tal como o Windows, Linux, RTOS, dentre outros. Contudo, esta é uma limitação natural de servidores *web*, pois estes têm que aceitar múltiplas conexões simultâneas, o que só é possível com o uso de *treads*. Porém, cabe ressaltar também que, devido à arquitetura em módulos e blocos em que o *gateway* foi proposto, é possível uma linearização do fluxo de execução do programa, tornando-o *monotread*. Cabe contudo ressaltar que tal abordagem levará a limitação do atendimento a apenas uma requisição de clientes *web* por vez.

A figura 3.11 ilustra como seria a arquitetura do sistema *monotread*. Esta solução não foi implementada devido ao fato das plataformas escolhidas possuírem suporte a programação paralela, porém é factível o aproveitamento dos mesmos blocos codificados para o *gateway* para a confecção do sistema apresentado, caso seja estritamente necessário.

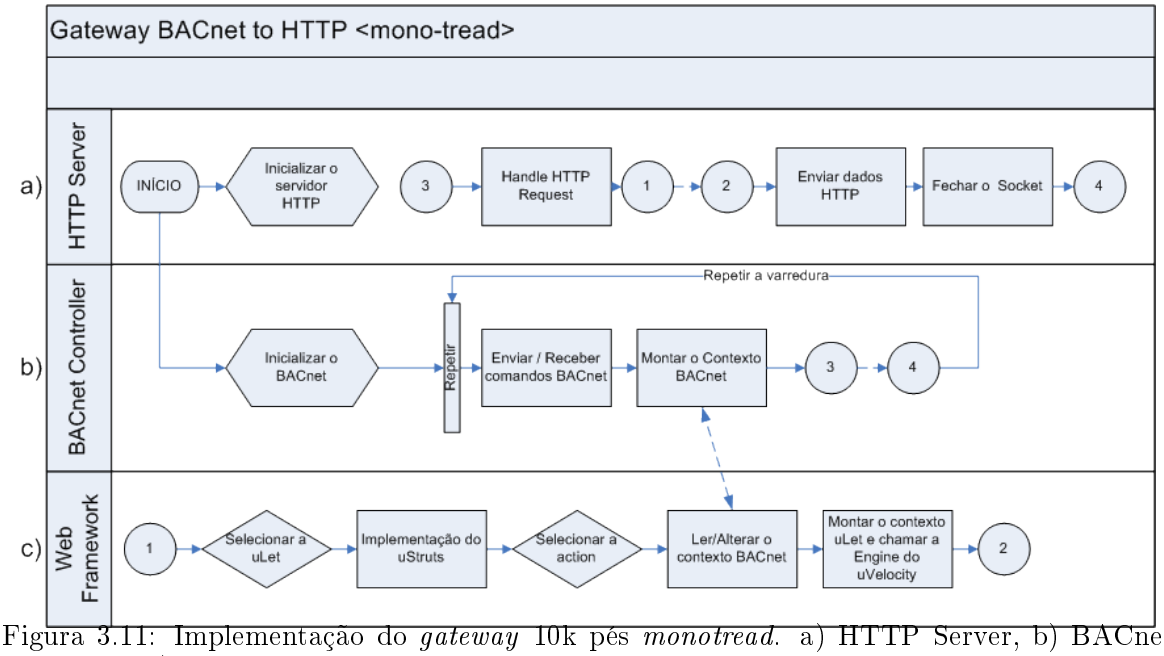

Figura 3.11: Implementação do *gateway* 10k pés *monotread*. a) HTTP Server, b) BACnet Controller, c) Web Framework

O *framework web* (*web framework*) inicia as atividades de processamento tanto no caso *multitread* (vide figura 3.10) quanto no caso *monotread* (vide figura 3.11) recebendo os dados das solicitações dos clientes *web*. Estas solicitações são recebidas pelo servidor *web* e enviadas para a fábrica de *uLets* (*uLets factory*), onde é definido qual *uLets* irá atender a solicitação através da URL enviada.

A *uLet action* irá buscar qual a *action* irá montar o *contexto uLet* daquela solicitação, através da busca de variáveis internas do servidor ou mesmo através de leitura ou alteração de parâmetros do contexto BACnet. Por fim, a *uLet action* (vide figura 3.10) chama a *engine* do *uVelocity* para realizar a fusão (*merge*) do *"contexto uLet"* com o modelo (*template*) VM definido para aquela dada *action*, passando uma estrutura/objeto *response* para que o servidor *web* se encarregue de enviar os dados para atender a solicitação do cliente.

3.4.2.1 Análise dos detalhes da codificação

Após uma abordagem superficial da mecânica do *gateway*, segue um aprofundamento de como foi implementado cada uma das funcionalidades.

Ao executar o *gateway* o mesmo inicializa os processos fundamentais Servidor HTTP e o Controlador BACnet. Seguindo esta abordagem simples, é possível acrescentar

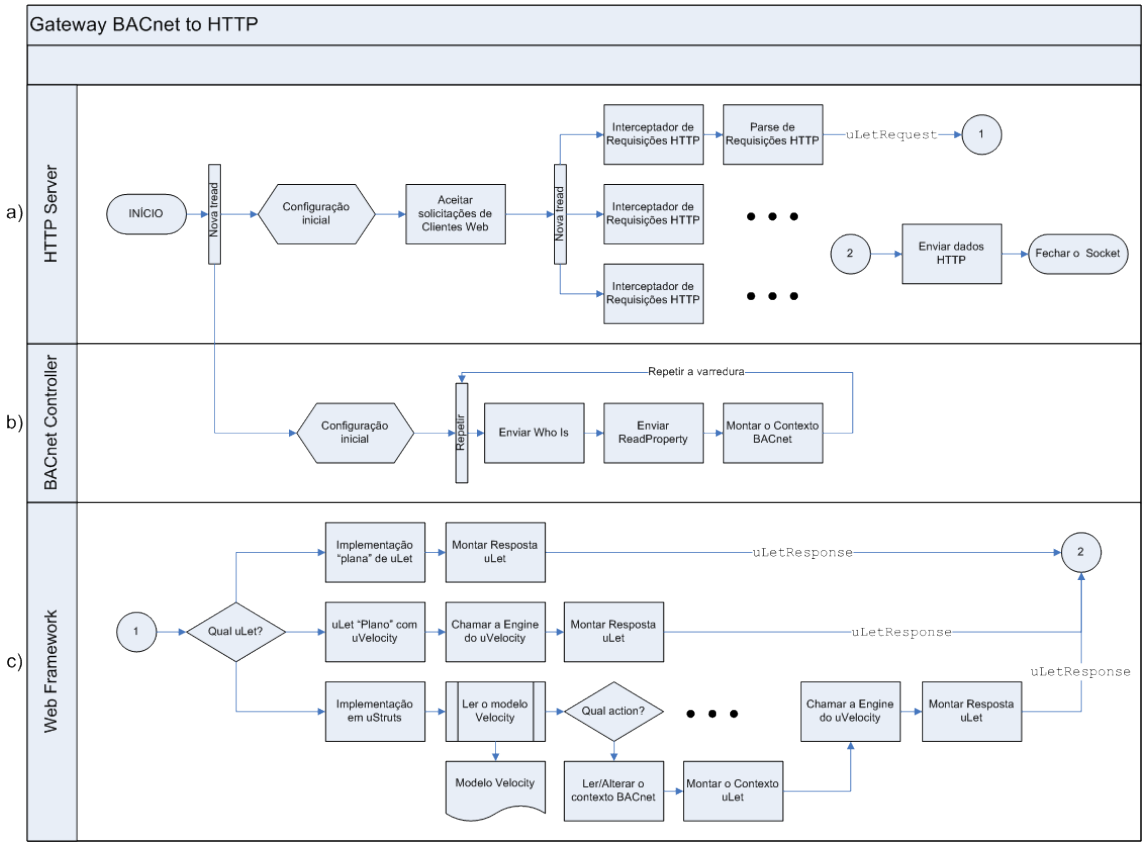

Figura 3.12: Implementação do *gateway* 1000 pés. a) HTTP Server, b) BACnet Controller, c) Web Framework

outros protocolos no conversor de protocolos, sendo que cada um será apenas uma nova entrada nesta inicialização (vide listagem 13).

```
   Entrada: Parâmetros do sistema
   Saída: Código de finalização do gateway
1  para cada Protocolo i faça
2      inciaNovaTread(protocolo(i));
```

Listagem 13: Inicialização do *gateway*

Na etapa da inicialização do servidor HTTP é criado o *socket* servidor (*server socket*) configurado para que este seja aberto de uma forma compreensível para o HTTP, definindo-se a porta e o IP do servidor. Durante a abertura do *socket* podem ocorrer alguns erros tratáveis como, por exemplo, a porta escolhida já estar sendo usado por outro processo do sistema operacional. Esta classe de erro deverá ser mostrado para o usuário e terminado o processo, visto que não se tem condições de continuar o processo.

3.4.2.2 Acept Web Client

Após todos os ajustes realizados, o servidor fica esperando as requisições no *socket* servidor. Quando ocorre a chegada da requisição, esta é tratada por um *tratador de requisições HTTP* (*handle HTTP request*), que é executado em uma *thread* específica para cada requisição, deixando assim o servidor livre para receber outra solicitação (vide figura 3.10).

3.4.2.3 Tratador de requisições HTTP (*Handle HTTP request*)

```
   Entrada: Objeto data
   Saída: Sem retorno
 1 inicializacao();
 2 se size == ERROR ou size == 0 então
 3 |   OutputScreenError("Error calling recv()");
 4 senão se !ParseHTTPHeader(receivebuffer,request) então
 5 |   OutputHTTPError(socketCliente, BAD REQUEST);
 6 senão se request.metodo igual a GET então
 7 |   response.status = OK;
 8 |   doGet(request,response);
 9 |   se response.status == OK então
10 |   |   sendbuffer += HTTP/1.0 200 OK /r/n;
11 |   |   sendbuffer += Content-Type: “response.mineType";
12 |   |   sendbuffer += /r/nContent-Length:“response.contentLength”/r/n/r/n;
13 |   |   send(socketCliente,sendbuffer,strlen(sendbuffer),0);
14 |   |   send(socketCliente,response.output,response.contentLength,0);
15 |   senão
16 |   |   OutputHTTPError(socketCliente, response.status);
17 senão
18 |   OutputHTTPError(socketCliente, NOT IMPLEMENTED);
19 fechar o socket socketCliente;
```

Listagem 14: HandleHTTPRequest

O *tratador de requisições HTTP* (*Handle HTTP request*) recebe como informações de entrada os dados do cliente que solicitou um dado recurso do servidor e a instância do *socket*, que foi aberta pelo cliente e lê as informações transmitidas nesta instância de

socket, passando esta informação para um decodificador de requisições HTTP (*HTTP request parser*) (vide listagem 14). Na linha 1 da listagem 14 ocorre a inicialização das principais variáveis a serem utilizadas, como o *uLetRequest*, *uLetResponse* e *socketCliente*. Das linhas 10 a 12 é escrito o cabeçalho da mensagem HTTP ("HTTP/1.0 200 OK ◦Content-Type ...") no caso de sucesso (RFC1945, 1996). As linhas 17 e 18 da listagem 14 podem ser alteradas para a implementação das outras funções do HTTP.

Entrada: String receivebuffer, uLetRequest *request
Saída: Booleano, indicando se o parser foi possível de ser realizado

```
1  char urlParam[MAX PATH];
2  String pos;
3  int sizePos;
4  sizePos ⇐ 0;
5  pos ⇐ strtok(receivebuffer,);
6  se pos == NULL então
7      retorna FALSE;
8  sizePos ⇐ strlen(pos)+1;
9  request.metodo ⇐ (String) malloc(sizePos);
10 strncpy(request.metodo,pos,sizePos);
11 pos ⇐ strtok(NULL,' ');
12 se pos == NULL então
13     retorna FALSE;
14 //Inicia a analise da url e do(s) parametro(s) strcpy(urlParam,pos);
15 pos ⇐ strtok(NULL,'/r');
16 se pos == NULL então
17     retorna FALSE;
18 sizePos ⇐ strlen(pos)+1;
19 request.protocolo ⇐ (String) malloc(sizePos);
20 strncpy(request.protocolo,pos,sizePos);
21 decomporUrlParametro(urlParam, request);
22 retorna TRUE;
```

Listagem 15: Parser de cabeçalho HTTP

a) *Parser* de requisições HTTP: (vide listagem 15 e vide figura 3.12 a) O *parse* foi desenvolvido com base na (RFC1945, 1996) e os dados do cabeçalho processado são armazenados em uma estrutura/objeto chamado uLetRequest. Após o

parser do cabeçalho HTTP começa a execução do método HTTP solicitado. Caso o método solicitado não esteja implementado pelo servidor, este retorna um erro HTTP 501 que significa *Não Implementado* (*Not Implemented*). Para o processamento desta solicitação, o servidor chama a fábrica de *uLets*, que recebe o *uLetRequest* e devolve o *uLetResponse*.

3.4.2.4 Fábrica de *uLets* (*uLet Factory*)

Ao receber a solicitação do servidor para processar uma dada requisição, a *uLetFactory* agrega a informação da URI em termos do sistema operacional, visto que foi delegado a *uLetFactory* a responsabilidade de deter a informação do diretório base (*wwwroot*) do servidor HTTP.

Neste ponto, o servidor tem que ser aderente a seção 12.5 da (RFC1945, 1996) (*Attacks Based On File and Path Names*) e se prevenir de ataques baseados em nomes de arquivos. Este ataque pode ser realizado colocando caminhos relativos na URL, se utilizando do conjunto de caracteres "../" que levaria a um nível acima no Sistema de Arquivos do Sistema Operacional (vide listagem 16, linhas 1 a 5).

Após agregar as informações do possível caminho do recurso no Sistema Operacional, a *uLetFactory* identifica, com base na URL passada, qual é a *uLet* que irá atender a solicitação(vide listagem 16 linhas 7 a 12 e vide figura 3.13).

Nas linhas 7 e 8 da listagem 16 ocorre a primeira tentativa de delegação é para o *framework uStruts*. Para esta implementação foi utilizado como padrão as extensões de URL ".do" (STRUTS, 2010a), comum entre os desenvolvedores de Struts. Assim, toda URL terminada em ".do" irá ser direcionada para a *uLetAction*, uma interpretação simplificada da ActionServlet (STRUTS, 2010a).

Caso a URL não termine com ".do", estão a URL é comparada com as outras *uLets* implementadas e, caso não esteja entre as *uLets* cadastradas, o *uLetFactory* tentará ele mesmo buscar o recurso solicitado no sistema de arquivos do sistema operacional (vide listagem 16 linhas 13 e 14), tornado assim viável a existência de recursos estáticos no servidor, como por exemplo figuras, pdfs, css (folhas de estilo), HTMLs estáticos, dentre outros.

```
   Entrada: uLetRequest
   Saída: uLetResponse
1  String path = wwwroot + request.url;
2  fullpath(request.path,path,MAX PATH);
3  se request.path estiver fora de wwwroot então
4  |  response.status ⇐ SC FORBIDDEN;
5  |  retorna;
6  senão
7  |  se url termina com .do então
8  |  |  acionarFrameworkStruts;
9  |  senão se url=="networkDevices.vm" então
10 |  |  acionarULetNetworkDevices;
11 |  senão se url "..." então
12 |  |  ...;
13 |  senão
14 |  |  leiaDoSistemaOperacional;
```

Listagem 16: Algoritmo de seleção de uLets

Basicamente, existem três tipos de *uLets* implementados no *gateway* atualmente, Os quais são explicados a seguir:

a) *uLets* Planos (Plain uLets) A filosofia dos *uLets* torna factível a construção de diversas funcionalidades e recursos se utilizando das informações contidas no *gateway*, recursos estes que podem ser classificados da seguinte maneira (vide figura 3.13 a):

- Páginas HTML simples, como por exemplo páginas de erro que mostrem informações em um contexto mais reduzido, ou mesmo trechos de página como as requisitadas por solicitações Ajax (vide figura 3.13 b).
- Geração de páginas com *mimetype* diversos, como por exemplo XMLs, imagens,textos planos, dentre outros. Esta liberdade de geração se deve ao fato dos *uLets* terem uma alta abstração do restante do sistema, sendo assim os *uLets* as "células tronco" do *gateway*(vide figura 3.13 b e listagem 17).
- Pode ser distribuidor para outros recursos, *uLets* e até mesmo chamadas a funções

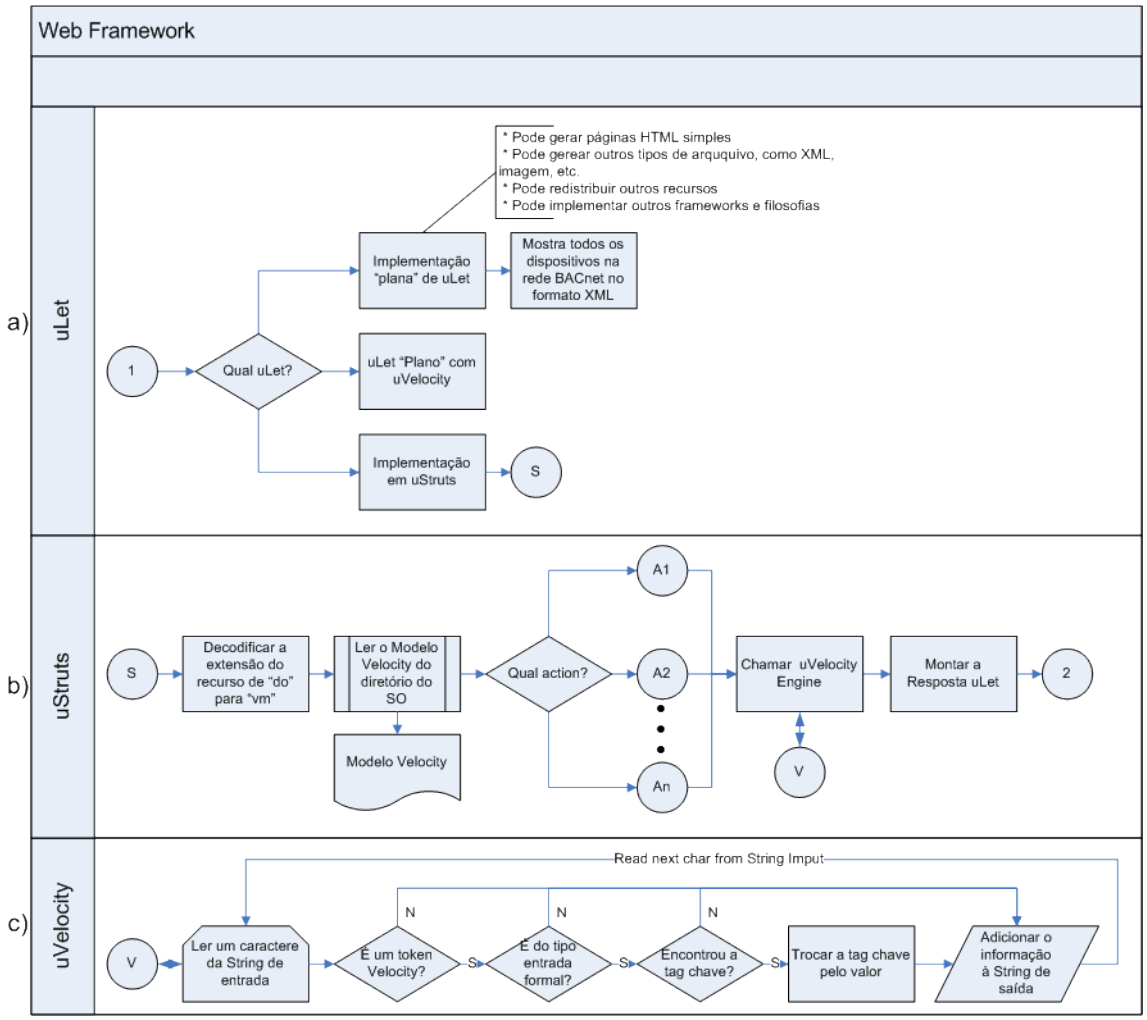

Figura 3.13: implementação do *web framework* do *gateway* 100 pés. a) uLet, b) uStruts, c) uVelocity

de código, sendo assim possível a criação de concentradores de URL como é o caso de digestores (*digests*) de Segurança de Acesso (vide figura 3.13 b).

- Pode-se implementar outros *frameworks* ou filosofias de programação, tal como ocorreu nesta dissertação com a implementação do *framework uStruts* (vide figura 3.13 c).

b) *uLets* Planos com uso de *uVelocity*: Devido ao fato dos *uLets* poderem utilizar qualquer função desenvolvida no código do *gateway*, estes podem se valer, por exemplo, das funcionalidades fornecidas pelo *uVelocity* que acrescenta a possibilidade da criação de conteúdo elegante e que disponham informações relevantes do contexto de execução do código. Seguindo esta abordagem foi criado o *uStruts* que vem a substituir este tipo de *uLet* (vide figura 3.12 c e figura 3.13 b).

```
Entrada: uLetRequest request, uLetResponse *response
Saída: Sem retorno
1  String resposta;
2  int sizeDevices ⇐ 1;
3  int sizeProps ⇐ SIZE VETOR;
4  Dispositivo ALL DEVICES ⇐ getALLDEVICES();
5  resposta ⇐ "< xml >";
6  para  i ⇐ 0 até sizeDevices faça
7      resposta += "< device >" ;
8      sizeProps ⇐ ALL DEVICES[i].last;
9      para  j ⇐ 0 até sizeProps faça
10         resposta += "<property id=";
11         resposta += ALL DEVICES[i].pars[j].id;
12         resposta += ">";
13         resposta += bacappPrintValueToString(ALL DEVICES[i].pars[j]);
14         resposta += "< /property>";
15     resposta += "< /device >";
16 resposta += "< /xml >";
17 response.output ⇐ resposta;
18 response.contentLength ⇐ strlen(resposta);
19 response.mineType ⇐ "text/xml";
```

Listagem 17: uLet de exemplo, gerando um XML com elementos da rede

c) *uStruts*: A partir do padrão de que toda URL com a extensão ".do" devesse ser tratada pelo *uStruts* (STRUTS, 2010a) e de que todo o modelo de Velocity teria a extensão ".vm"(VELOCITY, 2010), torna-se necessário que a primeira ação a ser tomada pelo *framework uStruts* é a tradução das extensões dos arquivos informados pelo *uLetFactory* de ".do" para ".vm", para que estes sejam efetivamente encontrados no Sistema de Arquivos do Sistema Operacional (vide figura 3.13 b).

Após esta etapa, o *uStruts* lê o modelo *uVelocity* instância o contexto através de uma chamada a uma dada uAction escolhida através da URL passada, em um procedimento semelhante ao realizado pelo *uLetFactory* para encontrar as *uLets* (vide figura 3.13 b).

Os futuros desenvolvedores que optarem por acrescentar funcionalidades ao *gateway* e optarem pela filosofia do *uStruts*, terão que apenas implementar uma nova uAction, endereçá-la no *uStruts* e criar um novo modelo *uVelocity* contendo o modelo que re-

presenta como os dados serão apresentados.

Após a montagem do contexto, o *uStruts* chama a mecanismo do *uVelocity* que irá realizar a fusão (*merge*) dos dados processados pela uAction e colocadas no contexto, com o modelo *uVelocity* guardado no Sistema de arquivos (vide figura 3.13 b).

No final do processamento, é montado o *uLetResponse*, com o conteúdo a ser retornado e a indicação do seu tamanho. Este *uLetResponse* é devolvido para o Servidor *web* que desempacota a resposta e transmite via *socket* para o cliente solicitante (vide figura 3.12 a).

3.5 SOLUÇÃO DE COMUNICAÇÃO ZIGBEE

A solução adotada para a conexão do módulo Zigbee ao STB visou ser a menos intrusiva no *hardware* do STB. Para atingir este objetivo, foi utilizada uma porta USB disponível, seja esta disponível no *kit* de desenvolvimento ou para atualizações de *firmware*(no caso de STBs comerciais). Esta escolha foi motivada pela possibilidade de uma melhor *aceitação de mercado*, bem como a manutenção da garantia dos usuários, além da possibilidade de se atender a vários STBs, de diversas marcas e modelos.

O dispositivo USB-Zigbee foi confeccionado de tal forma que se tem um conversor USB-Serial ligado a um conversor Serial-Zigbee (vide figura 3.14). Assim o STB ou qualquer computador que não tenha um módulo Zigbee poderá, através de uma porta Virtual COM, comunicar-se pela rede BACnet através do protocolo serial. Neste caso, optou-se pelo chip da FTDI, que já possui os *drivers* compilados para o STLinux.

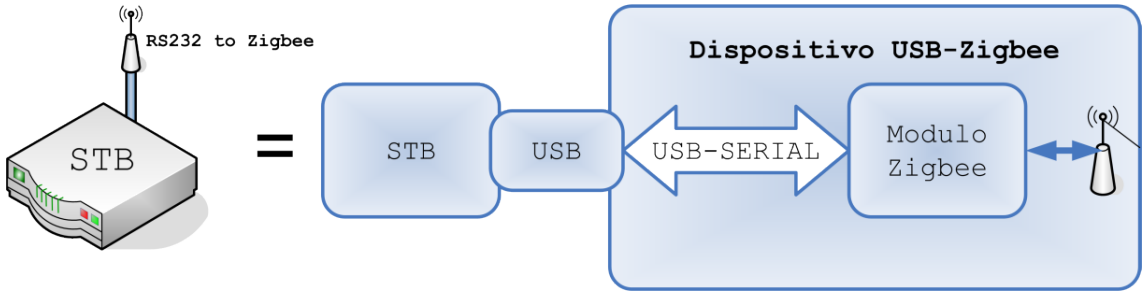

Figura 3.14: Ligação STB Zigbee via USB

3.6 DESCRIÇÃO DOS DISPOSITIVOS DESENVOLVIDOS

Nesta seção será apresentado um dispositivo USB confeccionado como um dos produtos desta dissertação. Este dispositivo serve como circuito de referência e ambiente de testes para as metodologias de integração entre o STB e a rede Zigbee. Na figura 3.15a é apresentado o esquemático do circuito de referência proposto. Já na parte 3.15b e 3.15c são apresentados a confecção do circuito, a frente e o verso. Este circuito é composto por um condicionador de sinais, visto que a alimentação fornecida pela porta USB é de 5V e o sinal D+ e D- é de 3V3 (USB, 2010). Este circuito é baseado em um dos circuitos de referência proposto pelo VUSB (VUSB, 2010).

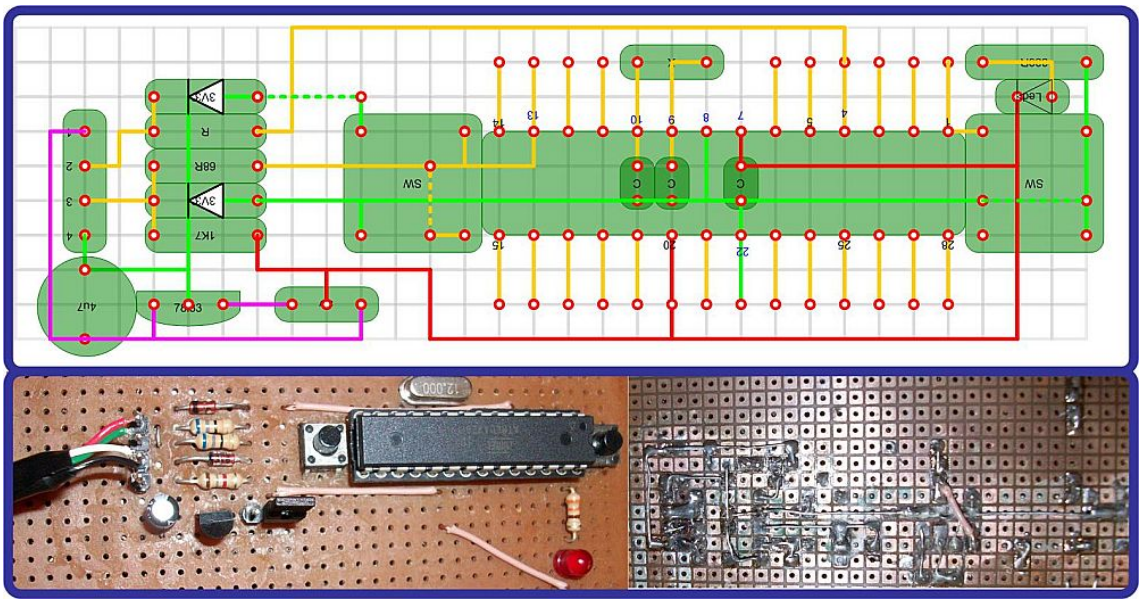

Figura 3.15: Esquemático do roteamento e soldagem do protótipo

3.7 INTEGRAÇÃO HARDWARE-SOFTWARE

Para a integração *hardware-software*, foram adotadas três estratégias de integração, de acordo com a porção do *gateway* que deverá estar contido dentro do STB. Estas estratégias podem ser definidas como: (a) *Gateway* BACnet completamente dentro do STB, (b) *Gateway* BACnet completamente dentro do STB com o desenvolvimento de *hardware* específico USB e (c) *Gateway* BACnet parcialmente dentro do STB e parcialmente dentro do *hardware* específico.

3.7.1 *Gateway* BACnet completamente dentro do STB

A primeira solução estratégia de integração adotada consistiu em colocar o *gateway* BACnet completamente dentro do STB, se valendo do poder computacional ocioso, utilizando para a ligação com a rede de automação residencial um conversor USB-serial, tal como apresentado na figura 3.16.

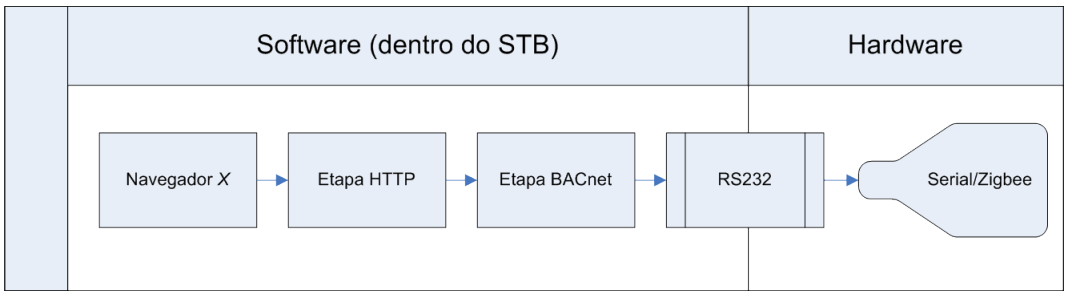

Figura 3.16: Modelo de integração *hardware-software* com o *gateway* totalmente no STB.

Esta abordagem contou como ponto positivo o desenvolvimento de apenas um programa, que viria a trazer uma série de benefícios, dentre elas:

- *Montagem de apenas um ambiente de desenvolvimento*: como se tem a geração de apenas um binário, se faz necessário apenas uma IDE, que no caso da execução do *gateway* BACnet para execução dentro de SoCs da empresa ST, já se tem o STWorkbench, que precisa apenas ser configurado para o projeto específico.
- *Método único de testes*: com o desenvolvimento de apenas um programa, pode-se adotar um método de testes mais simplificado, sendo necessário o teste deste único programa, mesmo apresentando todas as funcionalidades.
- *Redução do código*: por não se ter a necessidade do desenvolvimento de interfaces entre programas distintos, se tem uma menor quantidade de código a ser desenvolvido.

Contudo, a abordagem do desenvolvimento do *gateway* BACnet para ser executado apenas dentro do STB apresentou alguns pontos negativos, sendo que esta abordagem se mostrou inviável tecnicamente. São os pontos negativos encontrados nesta abordagem:

- *Necessidade de compilação de todo o código fonte para diversas plataformas*: partindo do interesse de atender um número maior de STBs, a estratégia da utilização

de um código único levaria a necessidade de recompilação de todo o *gateway* para cada arquitetura ou família de soluções.

- *Baixa prioridade da porta serial dentro dos sistemas operacionais modernos*: a porta serial possui uma prioridade muito baixa nos principais sistemas operacionais modernos tanto para a arquitetura PC quanto para os STBs. Dentro de muitas versões de SOs, a prioridade da interrupção da porta serial é inferior ao do acesso a disco. O protocolo MS/TP possui um requerimento de tempo muito forte, na ordem de dezenas de milissegundos (ASHRAE135, 1995), (BACNET, 2009). Ao verificar a principal lista de discussão aberta de BACnet, foi proposto o aumento da prioridade do processo para tentar mitigar este problema, através do comando *ionice* no Linux e o aumento da prioridade no Windows, com o uso de, por exemplo o ProcessExplorer (SYSINTERNALS, 2010). Foi testada estas duas opções, tanto na arquitetura PC quanto no STB, sem resultados satisfatórios.

Este modelo de solução para o casos do *gateway* (se ligando a uma rede BACnet/MSTP) foi mantido a fim de comparação com os outros. Contudo, cabe ressaltar que este ainda continua sendo o modelo mais interessante, no caso de se utilizar o *gateway* ligado a uma rede BACnet/IP, pelos seus pontos positivos e pela existência de viabilidade, ao não se utilizar a porta serial para comunicação e sim a porta de Ethernet.

Para prova de conceito de portabilidade do código fonte (para o caso do *gateway* se ligando a uma rede BACnet/IP) e levando em conta que o STB comercial que se dispunha para os testes não possuía porta Ethernet, se utilizou o SBC com processador ARM9 apresentado na figura 2.34 (BTL, 2008). A figura 3.17 mostram a execução do *gateway* HTTP - BACnet/IP dentro do SBC TS7300, sendo que o programa cliente é um navegador HTTP Firefox (FIREFOX, 2010), executando num computador com o sistema operacional Windows.

3.7.2 *Gateway* BACnet completamente dentro do STB com o desenvolvimento de *hardware* específico USB

Para tentar contornar a baixa prioridade da porta serial com os sistemas operacionais, buscou-se uma nova estratégia de integração do *gateway* HTTP-BACnet/MSTP para a rede de automação residencial, através do desenvolvimento de um *hardware* específico de comunicação. Para isto, se utilizou o *hardware* apresentado na figura 3.15. O

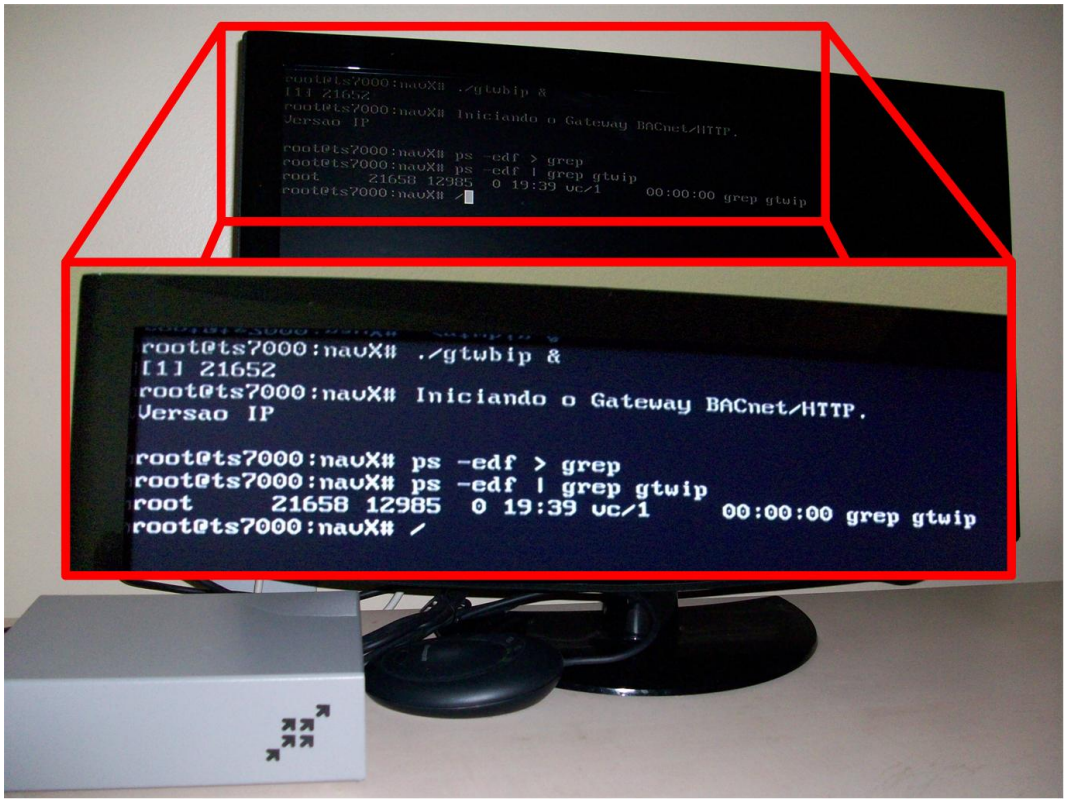

Figura 3.17: Execução do *gateway* HTTP - BACnet/IP dentro do SBC TS7300

hardware específico desenvolvido não visou impactar na estrutura do *gateway*, mas sim contornar as limitações do sistema operacional no tratamento da porta serial.

Contudo, antes de se realizar as alterações no *gateway*, foi realizado um teste de desempenho de velocidade de comunicação, a fim de verificar a viabilidade na utilização desta abordagem de integração hardware-software para o caso do protocolo BACnet/MSTP. Os resultados deste teste são apresentados na figura 3.18.

O tamanho máximo da palavra que se pode transmitir por HID pela USB1.1 é de 254 (levando em conta o cabeçalho e o CRC). Com os dados apresentados, realizou-se uma interpolação linear do tempo gasto para transmitir um pacote pelo número de palavras do pacote. A interpolação linear foi aproximada para a equação 3.2, onde o *tempo* é dado em milissegundos e o *tamanho* da palavra é dado em bytes. Pode-se interpretar a equação 3.2 tendo que a cada 4 bytes enviados se tem um aumento de 1 ms no tempo total de transmissão do pacote, com 5.8 ms para o controle do pacote. Para completar a análise (verificando a figura 3.18), observa-se que o tempo de comunicação não seria menor que 8 ms para a configuração adotada nos testes.

$$tempo = \frac{tamanho}{4} + 5.8 \tag{3.2}$$

Tamanho	Tempo (ms)	kbps
1	8.000	-
2	8.000	-
8	8.000	-
16	9.930	25.780
32	13.950	36.703
64	21.730	47.124
128	37.930	53.994
200	55.995	57.148
254	69.870	58.165

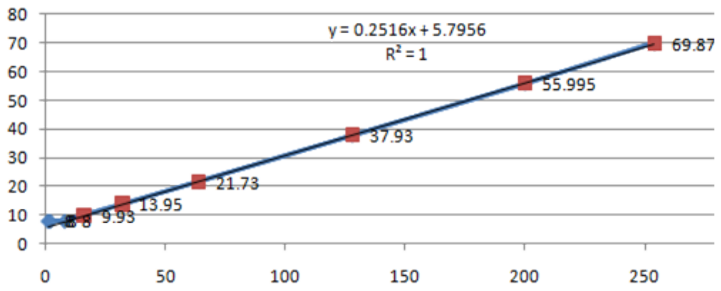

Figura 3.18: Dados do desempenho de comunicação com o dispositivo USB desenvolvido

Com a taxa observada na equação 3.2 (verificada na figura 3.18) observa-se que caso se consiga manter o tamanho do pacote NPDU pequeno pode serem mantidos os quesitos de tempo real do protocolo BACnet MS/TP, porém ao custo de sempre se ter nomes de dispositivos e parâmetros muito pequenos, o que poderia levar a inutilidade do *gateway* em alguns casos. Outro ponto a ser ressaltado é que todos os testes foram realizados com a maior prioridade possível de sistema operacional, além de se ter o computador sem uma carga massiva do processamento durante a realização dos testes. Alguns testes que não respeitam este cenário foram realizados, o que levou a uma inconstância nas execuções simultâneas, com alguns atrasos esporádicos. A partir desta verificação, optou-se pelo interrupção dos testes deste modelo de integração e a tentativa da adoção do modelo apresentado na seção 3.7.3.

3.7.3 *Gateway* BACnet parcialmente dentro do STB e parcialmente dentro do *hardware* específico

A fim de atender os requisitos de tempo real impostos pelo protocolo BACnet MSTP, optou-se pelo desenvolvimento de um *firmware* com toda a pilha BACnet (vide figura 3.19. Neste modelo de integração, utilizaram-se os seguintes recursos: (a) pilha USB1.1 VUSB (que é própria para microcontroladores AVR) (b) pilha BACnet portada para microcontroladores ATmega168 e (c) um microcontrolador ATmega 328p. Cabe relembrar que foi mantido o *bootloader* da gravadora USBasp, que ocupa 4K de memória, restando portanto 28K para o *firmware* final, no caso do ATmega 328p.

Por outro lado, foram realizados alguns testes de compilação, a fim de se obter um ambiente mais fácil de dar manutenção visando atender o tamanho disponível dentro

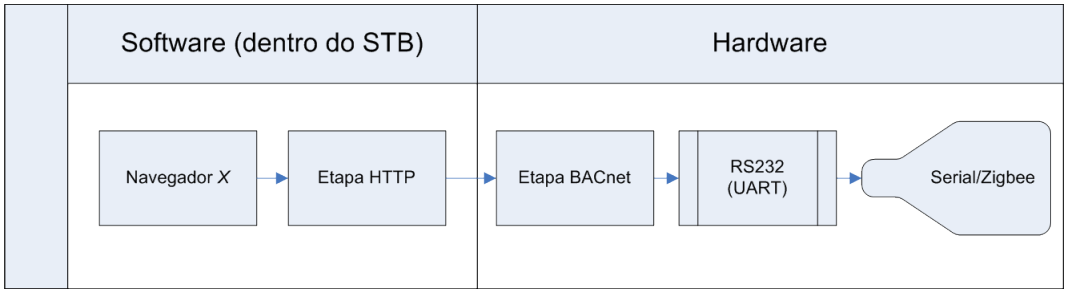

Figura 3.19: Modelo de integração *hardware-software* com o *gateway* parcialmente no STB.

do microcontrolador. Nestes testes (ao se colocar todos os arquivos da pilha BACnet em um único e simples passo de compilação) obteve-se um arquivo binário com cerca de 30K de tamanho, mesmo utilizando otimização de compilação para espaço (-Os (STALLMAN, 2002)), o que não cabia dentro do espaço disponível no *hardware* com o *bootloader*. Portanto, utilizou-se o mesmo procedimento adotado na configuração padrão da pilha BACnet (BACNET, 2009) para o microcontrolador ATmega168. Este procedimento realiza a geração de uma biblioteca estática, que na ligação do código pelo (STALLMAN, 2002), importa apenas as funções realmente utilizadas no código final.

A abordagem da geração da biblioteca e depois da sua ligação com o código final resultou em um *firmware* com os seguintes tamanhos: (a) cerca de 16KB para a pilha BACnet, (b) cerca de 18KB para a pilha BACnet com a pilha USB. No pior caso de utilização da memória (18KB) (levando-se em conta a utilização do microcontrolador ATmega328P) se tem ainda cerca de 10KB para o restante da aplicação a ser desenvolvida. Este espaço este muito maior do que o disponível dentro de um microcontrolador ATmega8 (com os seus 8K de memória flash), utilizado em soluções mais simples de automação residencial.

4 RESULTADOS

Neste capítulo são apresentados alguns cenários de uso do *gateway* BACnet/HTTP desenvolvido, bem como dos outros subprodutos, demonstrado a aplicação dos métodos e técnicas analisados nesta dissertação. A apresentação deste capítulo começa com a configuração das ferramentas de *software*, passando pelas alterações no *hardware* do STB do tipo comercial. Posteriormente é apresentado alguns casos de uso dos dois softwares mais relevantes, o navegador *X*e o *gateway* BACnet/HTTP em si. Por fim, é apresentado o dispositivo BACnet desenvolvido para testes.

4.1 O QUE FOI DESENVOLVIDO

4.1.1 Configuração de ambientes desenvolvimento de software embarcado

Como subproduto desta dissertação tem-se a configuração de uma máquina virtual (vide figura 4.1) com a versão do kernel 2.6.17.14-stm22-0038 e versão do GCC 4.3.4 20091123 específica para o STB adotado nesta dissertação. A partir desta configuração tornou-se possível a inserção de novos módulos dentro do STB adotado, como por exemplo o *driver* do conversor USB-Serial da FTDI utilizado como porta de saída dos comandos BACnet para a rede doméstica.

Figura 4.1: *Printscreen* de uma máquina virtual executando o aplicativo de programação fornecido pela ST

4.1.2 Resultados sobre STB adotado

Para a obtenção dos dados referentes as configurações de fábrica do STB adotado fez-se necessário o estudo de soluções consolidadas no mercado descrita na seção 3.1. Isto resultou numa alteração do STB adotado, onde foi disponibilizado uma porta serial (vide figura 4.2) através do qual se tem o acesso como *root* ao sistema operacional do STB. Foi também instalado dentro do STB um conversor USB-serial a fim de se ter maior confiabilidade da alteração, evitando o uso de fiação externa.

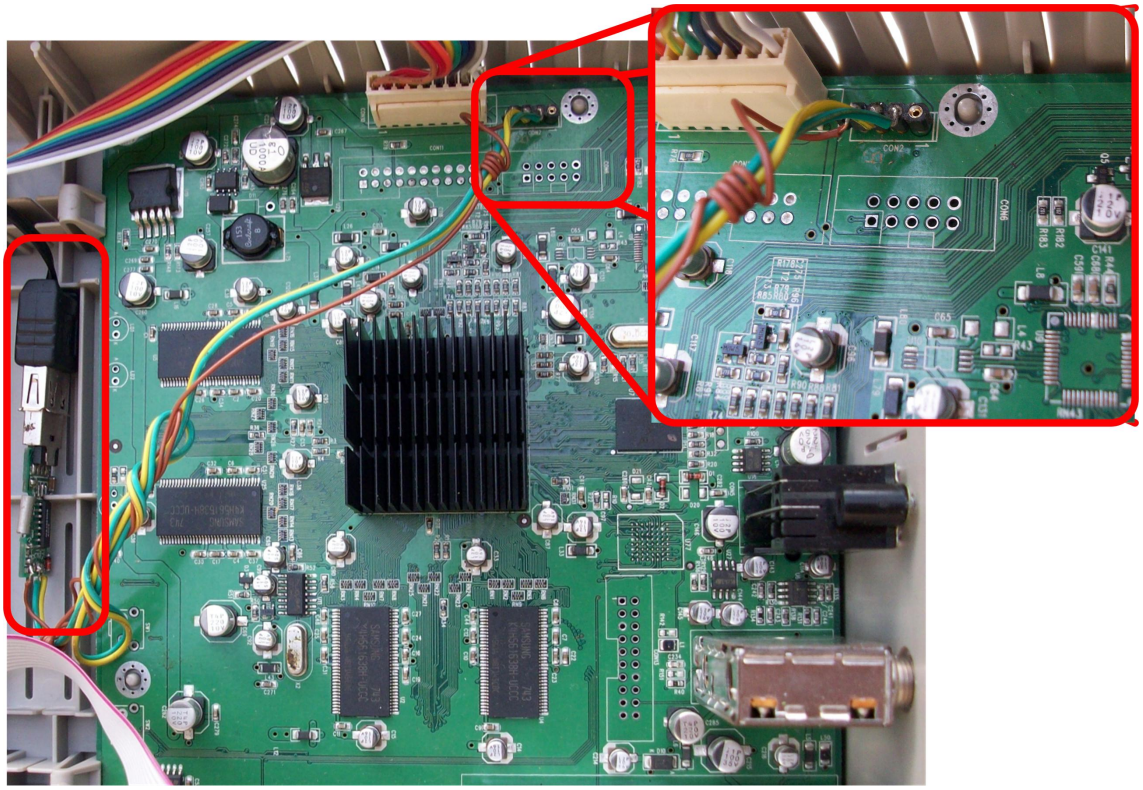

Figura 4.2: Fotografia do STB adotado com a porta serial disponibilizada. No detalhe, a soldagem do conector.

Também foi desenvolvido uma atualização de *firmware*, a partir de versões antigas do *firmware* original, com a finalidade de disponibilizar o acesso como *root* ao sistema pela porta serial e inserir os aplicativos desenvolvidos nesta dissertação (como o *gateway* BACnet/HTTP) o navegador e os *drivers* do controle remoto e do conversor USB/serial para a rede BACnet, bem como quaisquer outros programas desenvolvidos em trabalhos futuros.

4.1.3 Navegador orientado a imagem de finalidade específica

O navegador orientado a imagem de finalidade específica (navegador *X*) desenvolvido tem dois pontos importantes a serem apresentados: (a) o mecanismo de entrada de comandos e (b) o mecanismo de saída de informação.

4.1.3.1 O mecanismo de entrada de comandos do navegador *X*

O mecanismo de entrada de comandos do navegador *X*disponibilizada nesta dissertação é um controle remoto (vide figura 4.3). Buscou-se, originalmente, utilizar o mesmo controle remoto disponibilizado com o STB adotado, porém devido ao fato de não ter-se conseguido obter as informações de configuração do mesmo, optou-se pela aquisição e utilização de um controle-remoto melhor documentado, sendo encontrado no comércio local o controle remoto SHARP modelo G1324SA (vide figura 4.3). Este controle possui suas teclas já mapeadas no site do lirc (LIRC, 2010). Com esta escolha, demonstrou-se a possibilidade de utilização de qualquer controle-remoto que opere com uma frequência portadora igual ou próxima a utilizada por estes controle-remotos, dando maior versatilidade a solução.

Cabe ressaltar que dentro da versão original do sistema operacional do STB comercial adotado existe um arquivo *lircd.conf* que não se refere ao controle remoto disponibilizado, sendo este arquivo igual ao disponibilizado com o kit de desenvolvimento no qual este STB se baseia, apresentado na seção 2.7.5.1 (página 64).

Figura 4.3: Controle remoto SHARP modelo G1324SA utilizado para enviar comandos ao navegador *X*

4.1.3.2 O mecanismo de saída de informação do navegador *X*

No contexto deste trabalho foi também desenvolvido um navegador orientado a imagem conforme descrito na seção 3.2. Este navegador exibe as informações da rede BACnet através da tela da TV, sendo executado dentro do STB. Este navegador se comunica com a rede doméstica através do *gateway* BACnet/HTTP, através de um *socket* HTTP. Na figura 4.4 é apresentado uma foto mostrando a execução do navegador *X* dentro do STB comercial, onde o mecanismo de saída é uma TV LCD e o de entrada é um controle remoto.

Figura 4.4: Tela do Navegador *X* em execução no STB comercial apresentado em uma TV LCD

4.1.4 *Gateway* BACnet/HTTP

Para executar o desenvolvimento do *gateway* BACnet/HTTP de uma forma que ele fosse plenamente portável entre diversas plataformas, os códigos foram testados em três plataformas distintas, uma arquitetura PC, um SOC do tipo TS7300 (vide seção 2.7.4.3) e no STB adotado com processador ST7100 (vide seção 2.7.5.1) e STlinux (vide

seção 2.8.1). Como resultado tem-se uma versão do *gateway* BACnet/HTTP portável e compilada para estas arquiteturas, as quais serão apresentadas a seguir.

4.1.4.1 Execução do *gateway* BACnet/HTTP em PC

Um dos resultados desta dissertação foi uma versão do *gateway* BACnet/HTTP compilado para ser executado na arquitetura PC. Esta foi a primeira versão do *gateway*, devido principalmente a grande gama de ferramentas de desenvolvimento, o que tornou as fases de desenvolvimento mais ágeis. A versão para PC pode ser executada tanto sobre o sistema operacional Windows quanto no Linux. Na figura 4.5 é apresentada a interceptação dos pacotes BACnet no Wireshark (seção 2.8.4), sendo que os pacotes foram transmitidos pelo protocolo MS/TP, na versão MS/TP do *gateway* BACnet/HTTP. Estes pacotes foram interceptados por um programa fornecido juntamente com a pilha BACnet utilizada (BACNET, 2009).

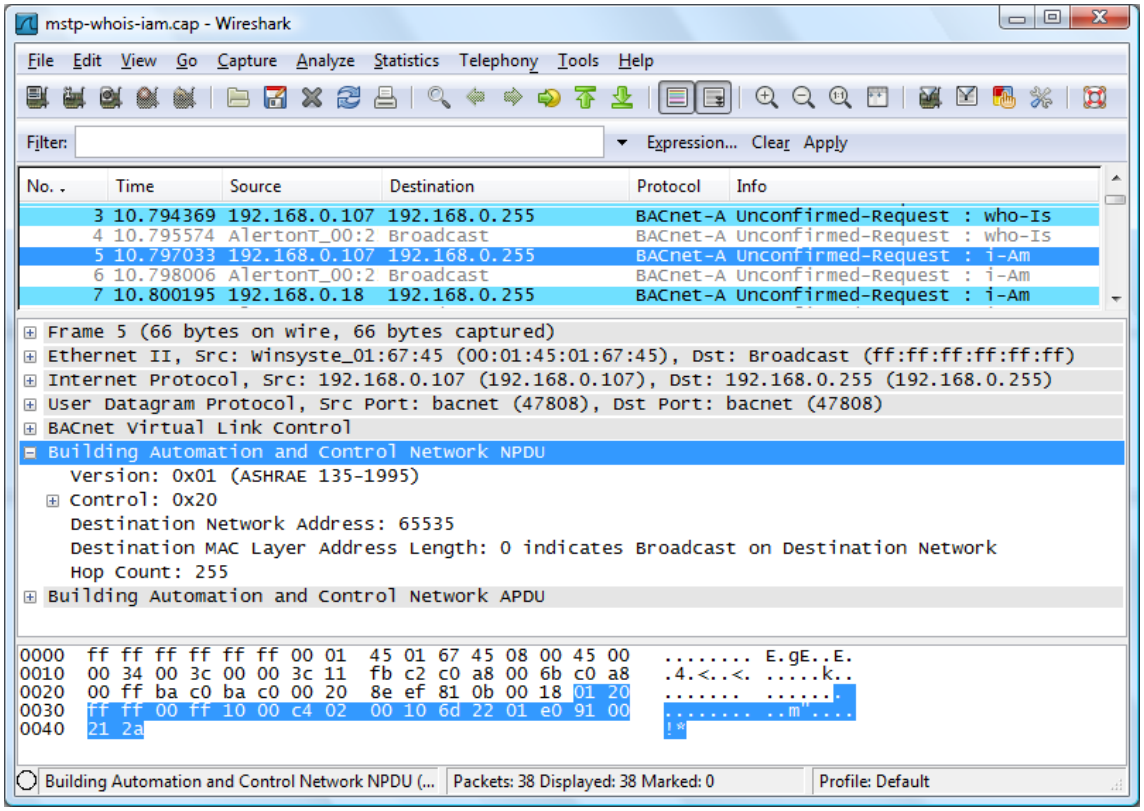

Figura 4.5: Wireshark interceptando o tráfego de pacotes BACnet

Já na figura 4.6 é apresentado o navegador *web* Firefox enviando solicitações para a rede de automação BACnet, através de uma interface de páginas desenvolvido especialmente para navegadores de uso geral. A funcionalidade de geração de páginas voltadas para

navegadores mais modernos (com mais recursos dos que os geralmente encontrados em sistemas embarcados) visa demonstrar os capacidades da solução proposta nesta dissertação para a geração de conteúdo para sistemas com mais recursos computacionais, como os encontrados na arquitetura PC.

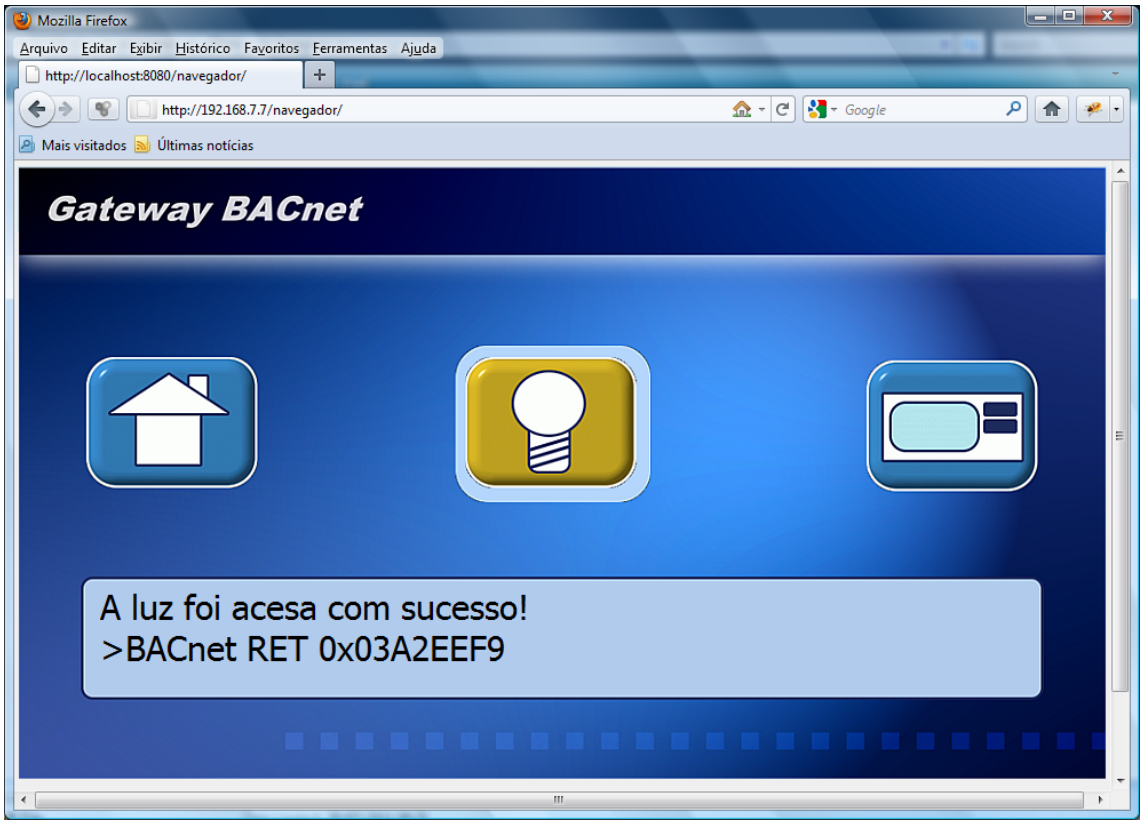

Figura 4.6: Execução de uma página do *gateway* BACnet/HTTP no navegador Firefox

4.1.4.2 Execução do *gateway* BACnet/HTTP no SBC TS7300

O *gateway* BACnet/HTTP também foi portado para o SBC TS7300 apresentado na seção 2.7.4.3, o qual possui um ARM9 como processador. Este resultado busca demonstrar a portabilidade do *gateway* em uma arquitetura mais acessível, tanto pela existência de mais recursos computacionais quanto pelos dispositivos de entrada e saída disponíveis. O uso desta arquitetura visou a facilitação da elaboração de trabalhos futuros, além deste SOC servir de um "meio termo" entre a arquitetura PC e os existentes nos STBs.

Na figura 4.7 é apresentado o arranjo para execução dos testes em laboratório, onde foi utilizado o SBC TS7300 ligado a um monitor e teclado, se comunicando por Ethernet com notebook. No computador portátil encontra-se o Wireshark executando a

Figura 4.7: Fotografia do arranjo de testes do *gateway* BACnet/HTTP

interceptação dos pacotes trafegados na rede local, assim como um navegador de uso geral enviando os comandos e recebendo os resultados do *gateway* BACnet/HTTP. O *gateway* está sendo executado dentro do SBC TS7300.

4.1.4.3 Execução do *gateway* BACnet/HTTP no STB

Outro resultado de portabilidade do *gateway* BACnet/HTTP foi a execução deste aplicativo dentro de um STB. As características do *gateway* permaneceram inalteradas das apresentadas para os casos de portabilidade para a arquitetura PC e para o SBC TS7300. No caso do STB utilizado neste trabalho, não foi possível a execução do *gateway* na configuração BACnet/IP, devido a inexistência de porta Ethernet neste equipamento na versão comercializada.

Para demonstrar a execução do *gateway* BACnet/HTTP no STB adotado, foi tirado um *printscreen* (vide figura 4.8) de uma listagem de processos do sistema operacional, onde se pode observar a execução dos processo do *gateway* BACnet/HTTP (processo PID 78 - gtwbm) e do navegador *X* (process PID 79 - navx).

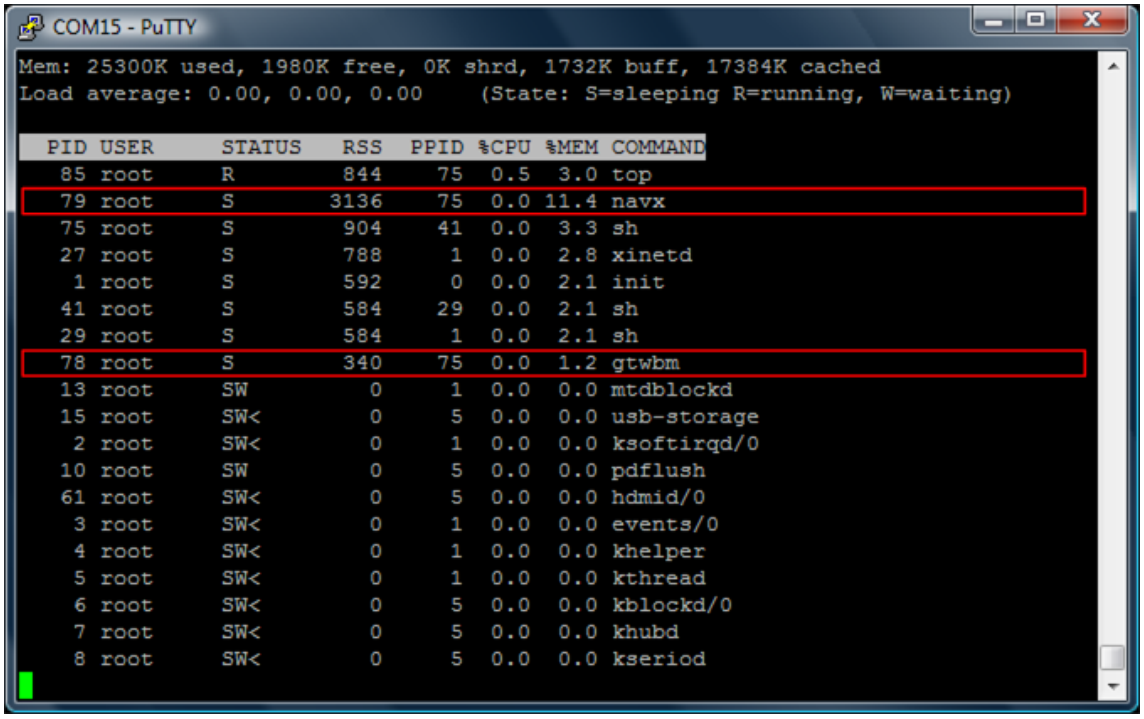

Figura 4.8: *Gateway* BACnet/HTTP executando dentro do STB adotado

4.1.5 Dispositivo BACnet (BACnet *device*) desenvolvido para testes em microcontroladores

Um dos resultados desta dissertação é um dispositivo BACnet de baixo custo baseado no microcontrolador ATMEGA328P apresentado na seção 2.7.4.2. Este dispositivo foi confeccionado em dois módulos, sendo eles: (a) placa de controle USB e (b) placa de potência a relê (vide figura 4.9).

A placa de controle USB possui a parte de *hardware* semelhante a apresentada na figura 3.15, diferenciando principalmente pela exposição do barramento de controle e pelo *firmware* contido no microcontrolador AVR. Esta versatilidade do *hardware* se foi alcançada, principalmente, pela simplicidade da solução adotada. Uma preocupação no tocante a configuração do *firmware* e na confecção do *hardware* foi a definição das funcionalidades dos pinos utilizados pelo USB. Teve-se como orientação no momento da escolha das funcionalidades dos pinos a manutenção da maior gama de funções, a fim de ser possível o reaproveitamento do *firmware* disponibilizado nesta dissertação para novos dispositivos sem a necessidade de realocação de pinos.

Quanto a placa de potência, foi adotado a solução mais simples e comum de mercado, ou seja, o acionamento a relê. A escolha por este tipo mais simples de acionamento

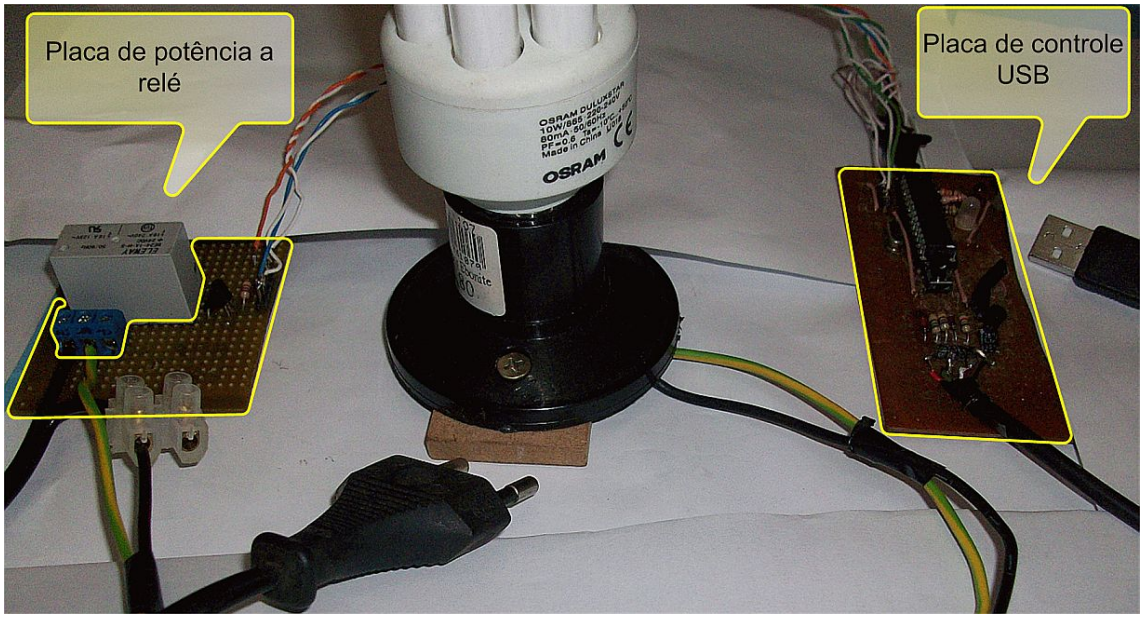

Figura 4.9: Dispositivo (*device*) BACnet MSTP para ligar e desligar lâmpadas

foi motivado pela grande gama de acionamentos que podem ser realizados, como por exemplo lâmpadas de todos os tipos, cargas indutivas, como motores e martelos eletromagnéticos, voltados para o acionamento de persianas e portas, bem como o desenvolvimento de tomadas de uso geral, onde não se tem o controle de qual o tipo de carga será conectada a tomada.

O circuito proposto de acionamento foi ligado a uma lâmpada florescente, do modelo mais comum a residências brasileiras, onde o controle da intensidade luminosa ("dimerização") só é possível com o uso de reatores específicos e caros. Assim, o caso de testes do dispositivo BACnet proposto nesta dissertação foi voltado para um dos pontos onde se acredita poder atender o maior número de pessoas e classes sociais, ou seja, no acionamento controlado do tipo binário.

4.2 ANÁLISE DOS RESULTADOS

O ambiente de desenvolvimento de aplicativos para STBs configurado nesta dissertação produziu diversos programas que são plausíveis de execução em STBs do tipo comercial. Este ambiente viabilizou o processo de desenvolvimento e compilação de aplicativos para STBs que possuam o CI ST7100 de uma forma facilitada. A montagem deste ambiente em uma máquina virtual facilita a portabilidade e replicação do mesmo, podendo este ser disponibilizado para diversos pesquisadores e estudantes a um custo

zero. A utilização da IDE Eclipse também facilita o desenvolvimento e sistemas mais complexos, devido a riqueza de ferramentas disponibilizadas por esta IDE.

Os estudos realizados em STBs do tipo comercial, além de agregarem conhecimento sobre esta arquitetura, demonstraram que este tipo de *hardware* pode ser utilizado como uma opção barata de ambiente de desenvolvimento. O uso de STBs comerciais e subsidiados pelo governo é demonstrado nesta dissertação e que, com poucas alterações, pode vir a fomentar um número maior de pesquisas em TV digital, através do barateamento de custos. O uso de *hardwares* comerciais para a pesquisa pode ser vantajoso tanto no quesito comercial quanto na adoção de uma abordagem mais cativante dos estudantes de graduação. Esta dissertação provou que esta abordagem é viável.

O navegador *X*vem a dar suporte a codificação de novos aplicativos voltados para a automação residencial, apresentando um modelo de codificação que pode ser utilizado diretamente em diversos dispositivos embarcados. Os conceitos de conforto e foco no usuário apresentado no capítulo 2, nortearam o desenvolvimento do navegador *X*, que visou a simplicidade dos comandos e do formato de uso, baseado em máquinas de estados. Também o uso de um controle remoto diferente do original do equipamento mostrou as facilidades da abstração proposta por ferramentas computacionais.

O *gateway* BACnet/HTTP proposto neste trabalho foi testado em diversas arquiteturas, demonstrando a portabilidade do mesmo. Este *gateway* pode ser utilizado da forma que é apresentado nesta dissertação. Também o estudo de algumas de suas característica podem dar suporte ao desenvolvimento de novas soluções conversão de protocolos de automação. As implementações dos *frameworks* uVelocity, uStruts e uLets propiciam uma simplicidade de codificação para ambientes Web desenvolvidos em linguagem C, geralmente encontrados em linguagem Java. Estas implementações foram realizadas de uma forma compacta, voltada para aos quesitos de portabilidade.

Por fim, o dispositivo BACnet desenvolvido neste trabalho, foi voltado ao baixo custo, contendo apenas um único CI e poucos circuitos passivos. A escolha da API de USB1.0 VUSB também foi motivada pela funcionalidade de gravação direta do microcontrolador Atmel sem a necessidade de *hardware* adicional (gravadora). Este dispositivo, mesmo com as respectivas limitações, apresentou-se funcional em diversas situações, como em iluminação e outros controles do tipo binário. O dispositivo BACnet ainda possui disponível entradas e saídas analógicas, com os quais podem ser implementados diversos outros tipos de controle.

5 CONCLUSÃO

Esta dissertação propôs uma metodologia de integração de automação residencial com receptores de TV digital, através da elaboração de um *gateway* BACnet/HTTP, com a confecção de um navegador específico para o caso de STB, além da elaboração de um dispositivo BACnet de baixo custo. A abordagem deste problema de integração de uma forma que pudesse ser reutilizada, total ou parcialmente em outros trabalhos, levou ao desenvolvimento de uma série de subprodutos, que podem ser utilizados das mais diversas maneiras.

No âmbito acadêmico, esta dissertação pode ser apresentada como uma leitura inicial para os trabalhos que envolvam automação residencial e TV digital, podendo ser lida tanto por pesquisadores provenientes da engenharia mecatrônica (que estão mais voltados para aos aspectos de programação embarcada e micro-eletrônica) quanto por engenheiros de redes (mais voltados para aos aspectos de rede e de TV digital) bem como por cientistas da computação interessados em produzirem soluções de automação residencial mais elegantes e com interfaces mais amigáveis.

Esta dissertação não visou uma explicação extremamente densa dos temas abordados, tendo o seu foco mais voltado para ser um ponto de referência inicial aos trabalhos que o sucederão. Buscou-se resolver os principais problemas geralmente encontrados nas fases iniciais dos projetos como, por exemplo, a configuração de ambientes, definição de padrões e protocolos, definição de ambientes de desenvolvimento e confecção primária de produtos que realizam a atividade fim, que é a automação residencial através do protocolo BACnet, executado sobre um receptor de TV digital, o STB.

5.1 Trabalhos futuros

Devido à característica inerente desta dissertação de ser um precursor no desenvolvimento conjunto de automação residencial e TV digital dentro da Universidade de Brasília, esta dissertação tem como finalidade a utilização em trabalhos futuros desta natureza. Com isto pode-se apontar alguns trabalhos a serem realizados dentro do

próprio *gateway*, acrescentando novas funcionalidades que lhe confeririam uma maior versatilidade no atendimento de suporte a novos trabalhos. Dentre estes cabe citar:

1. *Integrar o Navegador Xcom o programa de EPG*: atualmente não é possível executar o Navegador *X*e assistir TV ao mesmo tempo pois não foi feita a integração com o programa de sintonia e guia de programação do STB. Pode ser interessante o desenvolvimento de um EGP com o uso dass funcionalidades do navegador *X*.

2. *Desenvolver o gateway BACnet/ BACnet WS*: dando seguimento ao desenvolvimento do *gateway*, o desenvolvimento da possibilidade de comunicação pelo padrão BACnet webservice.

3. *Mapear o controle remoto específico do STB*: onde não se faria necessário a utilização de outro controle remoto que não o original de fábrica. Seria interessante também o desenvolvimento de um mecanismo de auto-mapeamento de controles remotos.

4. *Desenvolvimento novas funcionalidades do navegador*: a evolução do navegador, como por exemplo conteúdo de texto, melhor animação de ícones, dentre outros.

5. *Desenvolvimento de trabalhos utilizando o gateway*: como por exemplo o teste de carga numa rede BACnet com vários dispositivos, ou ainda o desenvolvimento de uma solução de automação com o uso de celulares, baseados no sistema operacional Android.

6. *Desenvolvimento de um supervisório*: seguindo na evolução do navegador, a criação de um supervisório, com a definição de uma série de elementos visuais padrões, voltados tanto para a automação de prédios quanto para automação de indústrias.

7. *Desenvolvimento de mecanismo para a visualização de câmeras de circuito interno de TV*: desenvolvendo assim uma solução integrada de segurança.

8. *Confecção de novos atuadores e sensores BACnet*: com base no dispositivo BACnet apresentado nesta dissertação, pode-se desenvolver diversos atuadores e sensores BACnet.

REFERÊNCIAS BIBLIOGRÁFICAS

15606-1, N. *Televisão digital terrestre - Codificação de dados e especificações de transmissão para radiodifusão digital Parte 1: Codificação de dados*. 4 2008. 30 p. Disponível em: <http://www.dtv.org.br/download/pt-br/ABNTNBR15606_2D1_2007Vc_2008.pdf>.

ARAYA, A. A. Questioning ubiquitous computing. In: *Proceedings of the 1995 ACM 23rd annual conference on Computer science*. New York, NY, USA: ACM, 1995. (CSC '95), p. 230–237. ISBN 0-89791-737-5. Disponível em: <http://doi.acm.org/10.1145-/259526.259560>.

ARM. Arm architecture reference manual (ddi0100e). *ARM*, v. 1, n. 1, p. 128, 2000.

ARM. *ARM - Processors*. 2010. Disponível em: <http://www.arm.com/products-/processors/index.php>. Acesso em: 11 abril 2010.

ASHRAE135. *A Data Comunication Protocol for Building Automaition and Control Networks*. 12 1995. 614 p. Disponível em: <http://www.bacnet.org>.

ASHRAE135. *ANSI/ASHRAE Addendum g to ANSI/ASHRAE Standard 135-2008*. 7 2010. 116 p. Disponível em: <http://www.bacnet.org/Addenda/Add-135-2008g.pdf>.

ASHRAE135. *ANSI/ASHRAE Addendum i to ANSI/ASHRAE Standard 135-2008*. 7 2010. 116 p. Disponível em: <http://www.bacnet.org/Addenda/Add-135-2008i-PPR4-Draft+_chair-approved_.pdf>.

ASHRAE135. *ANSI/ASHRAE Addendum k to ANSI/ASHRAE Standard 135-2008*. 1 2010. 11 p. Disponível em: <http://www.bacnet.org/Addenda/Add-135-2008k.pdf>.

ATSC. *Advanced Television System Committee*. 2010. Disponível em: <http://www-.atsc.org/>. Acesso em: 15 novembro 2010.

BACNET, P. *BACnet Stack*. 2009. Disponível em: <http://www.bacnet.sourceforge-.net>. Acesso em: 01 setembro 2009.

BRASIL, C. do. *Century do Brasil*. [S.l.], 2010. Disponível em: <http://centurybr-.com.br>. Acesso em: 10 setembro 2010.

BTICINO. *Bticino*. 2010. Disponível em: <http://www.bticino.com.br>. Acesso em: 01 setembro 2010.

BTL. *Tecnology System*. 2008. Disponível em: <http://www.tecnologysystem.com>. Acesso em: 01 setembro 2008.

BTL. *BTL Product Listings*. 2010. Disponível em: <http://www.bacnetinternational-.net/btl/>. Acesso em: 01 setembro 2010.

CENTER, M. D. *Visual Basic Developer Center*. 2010. Disponível em: <http://msdn-.microsoft.com/pt-br/vbasic>. Acesso em: 01 junho 2011.

CORRIGAN, S. *Introduction to the Controller Area Network (CAN)*. [S.l.], 2008.

CTPIM. *TV digital interativa*. 03 2004. 22 p. Disponível em: <http://www.ctpim-.org.br/tv_digital.pdf>.

DELMAR. *Modern Control Technology–Components and Systems*. a: Killian, 2004. 636 p. (s).

ECHELON. *Embedded Controller LC3020*. 2009. 0 p. Disponível em: <http://www-.echelon.com>. Acesso em: 01 dezembro 2009.

ECLIPSE. *Explorer the Eclipse Universe...* 2010. Disponível em: <http://www-.eclipse.org/>. Acesso em: 01 setembro 2010.

EN50090. *Home and Building Electronic Systems (HBES)*. 09 1996. 1 p. Disponível em: <http://www.knx.org>.

FACEBOOK. *Facebook*. 2010. Disponível em: <http://www.facebook.com>. Acesso em: 01 setembro 2010.

FIREFOX. *Firefox*. 2010. Disponível em: <http://www.mozilla.com/pt-BR/firefox/>. Acesso em: 01 setembro 2010.

FRIEDEWALD, M. et al. Perspectives of ambient intelligence in the home environment. *Journal*, 2005.

GECKO. *Gecko*. 2010. Disponível em: <https://developer.mozilla.org/en/Gecko>. Acesso em: 01 setembro 2010.

GEEKNET, I. *http://sourceforge.net/*. 2010. Disponível em: <http://sourceforge.net-/>. Acesso em: 16 janeiro 2010.

GINGA. *TV interativa se faz com Ginga*. 2010. Disponível em: <http://www.ginga-.org.br/>. Acesso em: 15 novembro 2010.

GRANZER, W.; KASTNER, W. *BACnet over KNX*. [S.l.], 2007.

GUIMARãES, A. de A. *Análise da norma ISO 11783 e sua utilização na implementação do barramento do implemento de um monitor de semeadora*. Dissertação (Mestrado) — Escola Politécnica - USP Universidade de São Paulo, 2003.

HALLIDAY; RESNICK. *Fundamentals of Physics*. [S.l.]: Jearl Walker, 2010.

HEATH, S. *ST40 system architecture, volume 1: system*. www.st.com: STMicroelectronics, 2007. 447 p. (SuperH tm (SH) 32-bit RISC series SH-4).

HOLANDA, A. B. de. *Dicionário da Lingua Portuguesa*. [S.l.]: Babylon, 2010.

IE. *Internet Explorer*. 2010. Disponível em: <http://www.microsoft.com/brasil-/windows/internet-explorer/>. Acesso em: 01 setembro 2010.

IE-WIKI. *Internet Explorer*. 2010. Disponível em: <http://en.wikipedia.org/wiki-/Internet_Explorer>. Acesso em: 01 setembro 2010.

IJG. *Independent JPEG Group*. 2010. Disponível em: <http://www.ijg.org/>. Acesso em: 01 setembro 2010.

IPHONE. *IPhone*. 2010. Disponível em: <http://www.apple.com/br/iphone/specs-.html>. Acesso em: 01 setembro 2010.

ISOIEC14543. *Home Electronic Systems (HES) Architecture*. 10 2007. 1 p. Disponível em: <http://www.iso.org>.

JAVA. *Java*. 2010. Disponível em: <http://www.java.com/pt_BR/>. Acesso em: 01 setembro 2010.

KASTNER, e. a. W. Communication systems for building automation and control. *IEEE*, v. 93, p. 1178–1203, 2005.

LEVY, M. The history of the arm architecture - (from inception to ipo). *ARM IQ*, v. 4, n. 1, p. 100, 2005.

LIN, C.-C.; CHEN, M.-S. On controlling digital tv set-top-box by mobile devices via ip network. *IEEE - International Symposium on Multimedia*, v. 1, n. 1, p. 8, 2005.

LIRC. *LIRC*. 2010. Disponível em: <http://lirc.org>. Acesso em: 16 janeiro 2010.

LITZ, L.; GROSS, M. Concepts and realization of an assisted living project by extended home automation. Note. 2007.

LIU, Q.; REN, P. Design and implementation of ms/tp in embedded system. *IEEE - Second IEEE Conference on Industrial Electronics and Applications*, v. 1, n. 1, p. 4, 2007.

LOYTEC. Embedded controller lc3020. *Datasheet*, January 2009, p. 2, 2009. Disponível em: <www.loytec.com>.

LYNX. *Lynx*. 2010. Disponível em: <http://lynx.isc.org/>. Acesso em: 01 setembro 2010.

MCLOUGHLIN, I. Secure embedded systems: the threat of reverse engineering. *IEEE International Conference on Parallel and Distributed Systems*, p. 729–736, 2008.

METASYS. *Metasys (R) System Extended Architecture*. [S.l.], 2010. Disponível em: <http://cgproducts.johnsoncontrols.com/MET_PDF/1201526.PDF>. Acesso em: 01 setembro 2010.

MüHLHäUSER, M.; GUREVYCH, I. Introduction to ubiquitous computing. *Human Computer Interaction*, 2009.

MHP. *MHP*. 2010. Disponível em: <http://www.mhp.org/>. Acesso em: 15 novembro 2010.

MICROSOFT. *O comprimento máximo da URL é de 2.083 caracteres no Internet Explorer*. 2010. 0 p. Disponível em: <http://support.microsoft.com/kb/208427>. Acesso em: 01 dezembro 2010.

MINASSIAN, A. A. *Audiência Pública - Impacto da Digitalização dos Serviços de Radiodifusão nos Procedimentos de Outorga de Rádio e Televisão*. 10 2009. Disponível em: <http://www2.camara.gov.br/atividade-legislativa/comissoes-/comissoes-permanentes/cctci/Eventos/apresentacoes/apresentacoes-2009/ap-20-10-2009-digitalizacao-da-radiodifusao/ANATEL-Ara-Apkar-Minassian.pdf>.

MOSAIC. *Mosaic*. 2010. Disponível em: <http://www.ncsa.illinois.edu/Projects-/mosaic.html>. Acesso em: 01 setembro 2010.

OLIVEIRA, F. S. de et al. Uma análise de metadados de tvd para suporte a informações de serviço no middleware ginga. *CEFETCE-Centro Federal de Educação Tecnológica do Ceará*, v. 1, n. 1, p. 4, 2008.

OLIVEIRA, M. et al. Diga ginga- digital automation in monitoring and control using ginga technology. *CEFETCE-Centro Federal de Educação Tecnológica do Ceará*, v. 1, n. 1, p. 7, 2008.

OPENGL. *OpenGL*. 2011. Disponível em: <http://www.opengl.org/>. Acesso em: 01 junho 2011.

OPERAMINI. *Opera para telefones*. 2010. Disponível em: <http://www.opera.com-/mobile/>. Acesso em: 01 setembro 2010.

OREBAUGH, A. *Wireshark Ethereal Network Protocol Analyzer Toolkit, Jay Beale's Open Source Security Series*. [S.l.]: Syngress Publishing, 2007.

PARSONS, B. Bacnet 20 years on. *Canadian Consulting Engineer*, v. 1, n. 1, p. 31 38, 2009.

PATTERSON, D. A.; HENNESSY, J. L. *Organização e Projeto de Computadores*. [S.l.]: Elsevier, 2008. 512 p.

PIRES, F.; GONÇALVES, M. N. *Desenvolvimento de um sistema de controle de acesso utilizando sistema operacional Linux*. Dissertação (Mestrado) — UnB - Universidade de Brasília, 2006.

POSITIVO. *DigiTV Positivo*. [S.l.], 2010. Disponível em: <http://digitvpositivo.com-.br>. Acesso em: 10 setembro 2010.

PPM. *ppm*. 2010. Disponível em: <http://netpbm.sourceforge.net/doc/ppm.html>. Acesso em: 01 setembro 2010.

PROVIEW. *XPS-1000*. [S.l.], 2010. Disponível em: <http://proview.com>. Acesso em: 10 setembro 2010.

RFC1945. *Hypertext Transfer Protocol – HTTP/1.0*. 02 1996. 1 p. Disponível em: <www.w3.org/Protocols/HTTP/1.0/spec.html>.

RFC3986. *Uniform Resource Identifier (URI): Generic Syntax*. 01 2005. 1 p. Disponível em: <http://tools.ietf.org/search/rfc3986>.

RS485. *Standard RS-485 Electrical Characteristics of Generators and Receivers for Use in Balanced Multipoint Systems*. 1985. 1 p. Disponível em: <http://www.eia.org>.

SERVLET. *Java Servlet Technology*. 2010. Disponível em: <http://www.oracle.com-/technetwork/java/index-jsp-135475.html>. Acesso em: 01 setembro 2010.

STALLMAN, R. M. *Using the GNU Compiler Collection*. [S.l.]: Free Software Foundation, 2002.

ÖSTERLIND, F. et al. *Integrating Building Automation Systems and Wireless Sensor Networks*. [S.l.], 2007. Disponível em: <http://www.sics.se/~fros-/osterlind07integrating.pdf>.

STLINUX. *www.stlinux.com*. 2009. Disponível em: <www.stlinux.com>. Acesso em: 01 dezembro 2009.

STMICROELECTRONICS. *ST40RA 32-bit Embedded SuperH Device*. www.st.com: STMicroelectronics, 2005. 92 p.

STMICROELECTRONICS. *www.st.com*. 2009. Disponível em: <www.st.com>. Acesso em: 01 dezembro 2009.

STRUTS. *How does Struts work?* 2010. Disponível em: <http://struts.apache.org/1-.x/faqs/works.html>. Acesso em: 10 abril 2010.

STRUTS. *http://struts.apache.org/1.x/index.html*. 2010. Disponível em: <http:/-/struts.apache.org/1.x/index.html>. Acesso em: 10 abril 2010.

STRUTS. *http://struts.apache.org/2.0.14/docs/comparing-struts-1-and-2.html*. 2010. Disponível em: <http://struts.apache.org/2.0.14/docs/comparing-struts-1-and-2-.html>. Acesso em: 10 abril 2010.

STRUTS. *http://struts.apache.org/2.x/index.html*. 2010. Disponível em: <http:/-/struts.apache.org/2.x/index.html>. Acesso em: 10 abril 2010.

SYSINTERNALS. *Windows Sysinternals*. 2010. Disponível em: <http://technet-.microsoft.com/en-us/sysinternals/default.aspx>. Acesso em: 01 setembro 2010.

TANENBAUM, A. S. *Computer Networks*. [S.l.]: Publisher, 2003. ISBN 8535211853.

THOM, D.; PURNHAGEN, H.; SUBGROUP the M. A. *MPEG Audio FAQ Version 9 MPEG-1 and MPEG-2 BC*. 10 1998. 1 p. Disponível em: <http://mpeg.chiariglione-.org/faq/mp1-aud/mp1-aud.htm>.

TZU, S. *A arte da guerra*. [S.l.]: Projeto Cultura Brasileira, 2010. 45 p.

UM0339. *SuperH(TM) (SH) 32-bit RISC series, SH-4, ST40 system architecture, volume 1: system*. [S.l.], 2007.

UMBERGER, e. a. M. The integration of home-automation and iptv system and services. *Computer Standards and Interfaces - Elsevier*, CSI-02616, p. 10, 2008.

USB. *USB*. 1 2010. 11 p. Disponível em: <http://www.bacnet.org/Addenda/Add-135-2008k.pdf>.

VELOCITY. *Velocity User Guide*. 2010. Disponível em: <http://velocity-.apache.org/engine/releases/velocity-1.5/user-guide.html>. Acesso em: 10 abril 2010.

VUSB. *http://vusb.net/*. 2010. Disponível em: <http://vusb.net/>. Acesso em: 16 janeiro 2010.

WEISER, M. The computer for the 21st century. Scientific American Ubicomp Paper after Sci Am editing. 09 1991.

WHATBROWSER. *O que é um navegador?* 2010. Disponível em: <http://www-.whatbrowser.org/pt-br/>. Acesso em: 01 setembro 2010.

WIRESHARK. *Wireshark*. 2010. Disponível em: <http://www.wireshark.org/>. Acesso em: 01 setembro 2010.

WPAUTOMACAO. *WP Automação*. 2010. Disponível em: <http://www-.wpautomacao.com.br>. Acesso em: 01 setembro 2010.

YOUTUBE. *Sobre o YouTube*. 2010. Disponível em: <http://www.youtube.com/t-/about>. Acesso em: 01 setembro 2010.

ZIGBEE. *ZigBee wireless technology (IEEE 802.15.4)*. 2010. Disponível em: <http://www.zigbee.org>. Acesso em: 01 setembro 2010.