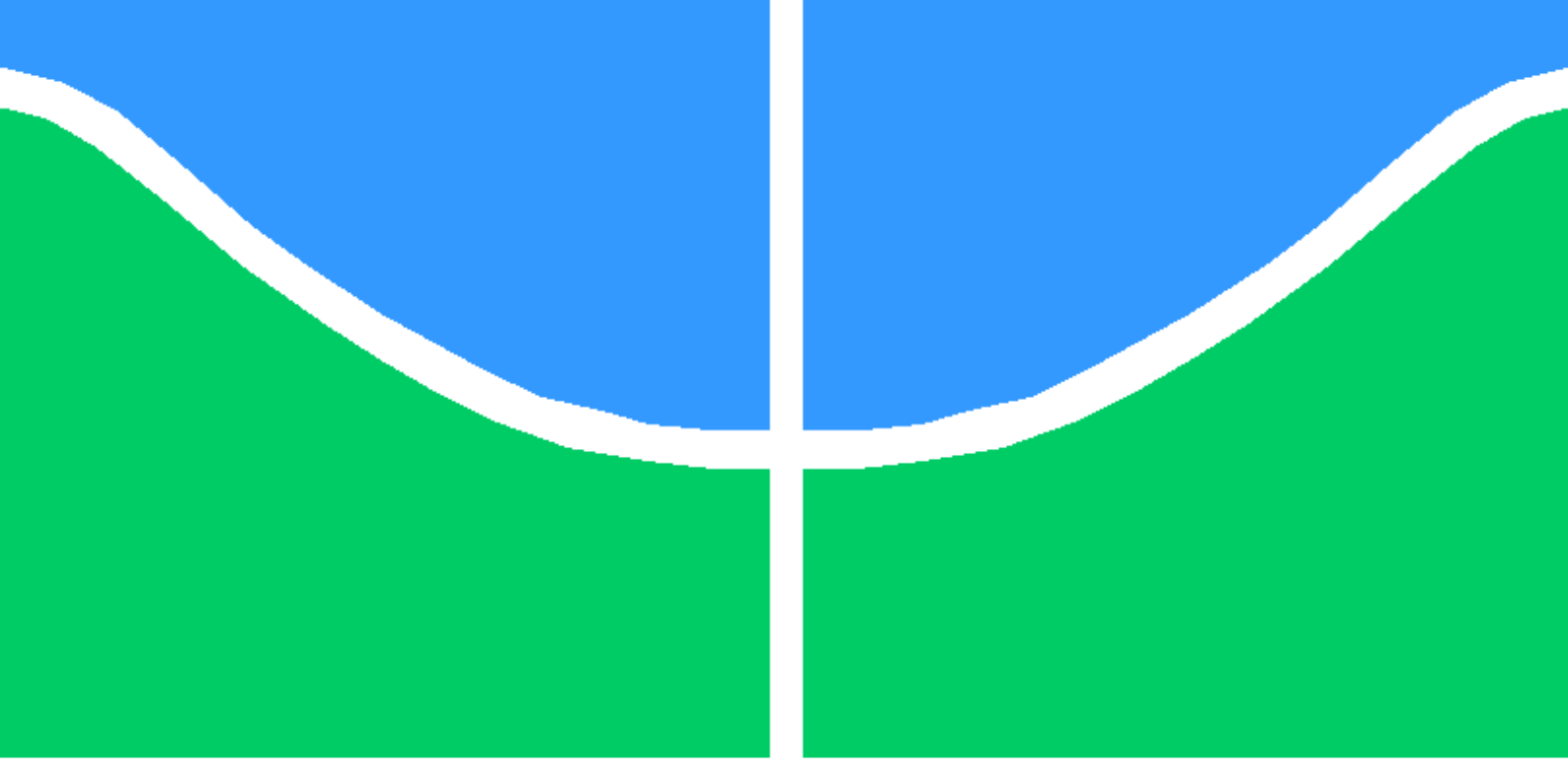

TRABALHO DE GRADUAÇÃO

IMPLEMENTAÇÃO DE BACNET SOBRE ZIGBEE PARA REDE DE AUTOMAÇÃO PREDIAL WIRELESS

Marcos Vinícius Toledo de Brito

Robson Paulo Fernandes da Silva

Brasília, Julho de 2009

ENGENHARIA
MECATRÔNICA
UNIVERSIDADE DE BRASÍLIA

UNIVERSIDADE DE BRASILIA
Faculdade de Tecnologia

TRABALHO DE GRADUAÇÃO

IMPLEMENTAÇÃO DE BACNET SOBRE ZIGBEE PARA REDE DE AUTOMAÇÃO PREDIAL WIRELESS

Marcos Vinícius Toledo de Brito

Robson Paulo Fernandes da Silva

Relatório submetido ao Departamento de Engenharia Mecatrônica como requisito parcial para obtenção do grau de Engenheiro de Controle e Automação

Banca Examinadora

Prof. Dr. Adolfo Bauchspiess, ENE/UnB _____
Orientador

Prof. Dr. Marco Antonio F. do Egito Coelho, ENE/UnB _____
Examinador interno

Eng. Me. Claiton Cesar de Urzêda, Spin Engenharia de Automação _____
Examinador externo

Dedicatórias

Dedico este trabalho ao meu amado Jesus e a meus pais e meu irmão, sem eles eu nada seria.

Robson Paulo Fernandes da Silva

Dedico este trabalho às pessoas que estão sempre ao meu lado. Aos meus pais, irmãs e à minha namorada.

Marcos Vinícius Toledo de Brito

Agradecimentos

A Deus por permitir que eu chegasse até aqui com saúde e determinação. Aos meus pais por terem me dado todo o suporte na vida até começar a caminhar com as próprias pernas, pelo incentivo e pela educação. À minha namorada que tanto me apóia nos momentos mais difíceis com suas palavras de carinho, compreensão e orientação. A todos os amigos conquistados durante a graduação. Aos amigos, Luís, Helger e Paulo, pela contribuição com conselhos, conhecimento e pelo agradável ambiente de trabalho. Ao Robson, amigo e companheiro de trabalho, por ter compartilhado seus conhecimentos, ensinamentos e pela paciência nos momentos difíceis. Ao Prof. Adolfo pelos inúmeros ensinamentos e orientações, pelo estímulo propiciado, pela estrutura e recursos de trabalho e pelo exemplo a ser seguido.

Marcos Vinícius Toledo de Brito

Primeira e infinitamente ao meu Senhor Jesus, e suas infinitas misericórdias, razão pela qual não somos consumidos. Por seu grande amor por mim, pela oportunidade e o sustento à mim durante todo o curso. Tudo Ele fez por mim, e me deu tudo o que tenho e que sou. "Porque dEle, por Ele e para Ele são todas as coisas. Glória, pois, a Ele eternamente. Amém" (Rm 11:36). Te amo Jesus! Aos meus pais, que sempre acreditaram em mim, apoiaram-me e incentivaram-me durante toda a vida. Em especial ao meu amado pai, que me ensinou o amor à engenharia e nunca me negou seus conhecimentos e amor. À minha mãe que me apoiou durante toda a vida e que tem me sustentado em oração. Ao meu irmão pela companhia, pelas discussões, pela outra forma de ver o mundo o que colaborou e muito para minha formação como pessoa. A todos os colegas e amigos da graduação, especialmente Érico Toscano e Farley Braz, amigos desde o colegial que, apesar de terem partido para outras áreas de atuação, sempre me apoiaram e são ótimos amigos. Ao Marcos, amigo e companheiro deste trabalho e muitos outros, pela paciência, excelência e conhecimentos compartilhados neste trabalho. Ao Prof. Adolfo, pelos inúmeros conhecimentos compartilhados conosco durante todo o curso e neste trabalho também, por ter nos apoiado e nos guiado com seus estímulos, conselhos e orientações.

Robson Paulo Fernandes da Silva

RESUMO

Este trabalho trata da implementação de dispositivos sensores e atuadores BACnet-Wireless. O protocolo de comunicação de dados BACnet, *Building Automation and Control Networks* [1], foi embarcado em microcontroladores da fabricante Atmel. Foram utilizados o ATmega168 para os módulos sensores e o ATmega32 para os módulos atuadores. O trabalho inova quanto ao meio físico de comunicação, onde foi utilizado o protocolo ZigBee pelas suas características ideais à utilização em automação predial e residencial. Para a validação dos trabalhos, utilizou-se o software CAS, *Chipkin Automation Systems*.

ABSTRACT

This work approaches the construction of sensors and actuators BACnet-Wireless devices. The data communication protocol BACnet, *Building Automation and Control Networks*, is embbeded in some Atmel microcontrollers. We used the ATmega168 for sensor devices and the ATmega32 for actuator devices. This work is a newness because of the physical layer, where was used the ZigBee communication protocol. To validate the work, we used the software CAS, *Chipkin Automation Systems*.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SÍMBOLOS

Subscritos

0x	Número em hexadecimal

Siglas

ABNT	Associação Brasileira de Normas Técnicas
ADC	Analog Digital Converter
AI	Analog Input (Entrada analógica)
ANSI	American National Standards Institute
AO	Analog Output (Saída analógica)
API	Application Interface
APDU	Application Package Data Unit
ASHRAE	American Society of Heating, Refrigerating and Air-Conditioning Engineers
BACnet	Building Automation and Control Networks
BO	Binary Ouput (Saída binária)
BI	Binary Input (Entrada binária)
CAS	Chipkin Automation Systems
CRC	Cyclic Redundancy
COS	Change of State
COV	Change of Value
DA	Destination Address
DER	Data Expecting Reply
FFD	Full Function Device
GRAV	Grupo de Robótica, Automação e Visão Computacional
I/O	Input / Output
ICI	Interface Control Information
ISM	Industrial, Scientific and Medical
ISO	International Standarts Organization
LARA	Laboratório de Robótica e Automação
LAVSI	Laboratório de Automação, Visão e Sistemas Inteligentes
LED	Light Emitting Diode
MAC	Media Access Control
MSTP	Master Slave Token Passing
NP	Network Priority
NPDU	Network Package Data Unit
NSAP	Network Service Access Point
OSI	Open System Interconnection
PAN	Personal Area Network
PHY	Physical Layer
PTP	Point-to-Point
PWM	Pulse Width Modulation
RDF	Reduced Function Device
SA	Source Address
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
UML	Unified Modeling Language
ZC	ZigBee Coordinator
ZED	ZigBee End Device
ZR	ZigBee Router

Capítulo 1

Introdução

Este capítulo apresenta a contextualização do tema abordado, bem como os objetivos do trabalho e sua principal motivação.

1.1 Contextualização

O mundo atual sofre constantes mudanças, sejam climáticas ou de outra natureza. A velocidade com que as informações trafegam, o avanço tecnológico e a demanda do mercado fazem com que as pessoas se dediquem cada vez mais a passar horas num mesmo ambiente interno.

O ambiente interno, seja residencial ou predial, passou a receber mais atenção quanto à propiciar maior conforto ao usuário. Devido a isto, surgiram variadas linhas de pesquisa que buscavam este objetivo de propor melhoria principalmente ao conforto térmico. Com o foco voltado para o usuário, emergiu o paradigma *Ambient Intelligence*, vindo a alterar a maneira como o ambiente era visto. Este passou a ser tratado como um conjunto de variáveis que poderia ser controlado de forma mais adequada e interagindo com a dinâmica da influência do ambiente exterior e do fluxo de pessoas a que estaria sujeito. Os conceitos dos ambientes inteligentes, dinâmicos e flexíveis.

Um ambiente é dito inteligente quando é capaz de adaptar-se às variações internas e externas sem intervenção humana, tornando-se operacionalmente independente, automático. O LAVSI, Laboratório de Automação, Visão e Sistemas Inteligentes, está envolvido com projetos que visam a implementação desse paradigma. O *Ambient Intelligence* é alcançado com a inserção de redes de sensores ambientes, que fornecem as informações necessárias aos equipamentos de atuação, propiciando desta forma a interação entre os módulos.

O paradigma *Ambient Intelligence* baseia-se em alguns índices já difundidos, um deles trata-se do PMV, *Predicted Mean Vote*. Índice variável de -3 a +3 sendo que o valor 0 indica maior satisfação dos usuários quanto ao conforto térmico. Ambientes inteligentes trazem economia e eficiência energética, pois podem otimizar o funcionamento de equipamentos.

O avanço das tecnologias e a aplicação da automação predial requeriu que houvesse uma padronização dos equipamentos dessa área. A ASHRAE, *American Society of Heating, Refrigerating and Air-Conditioning Engineers*, em 1999, criou o padrão de comunicação de dados BACnet. Definiu para tal o padrão ASHRAE 135-2004 [1], onde constam todas as especificações deste

padrão.

Buscando a padronização, a portabilidade e a interoperabilidade dos módulos, este trabalho busca a implementação de dispositivos sensores e atuadores que comuniquem-se utilizando o BACnet. Além disso, devido à popularização e ao avanço das tecnologias de comunicação wireless, seria interessante a criação de dispositivos que já utilizassem desse advento.

1.2 Motivação do problema

A aplicação da automação predial restringiu-se inicialmente a ambientes comerciais e industriais, mas ultimamente vem se ramificando e sendo já bastante aplicada em automação de ambientes residenciais. Sua popularização aliada à difusão das tecnologias wireless, torna o dispositivo bem mais versátil. Além disso, o fato da utilização de comunicação sem fio, traz grandes economias quanto ao gasto com cabeamento e propicia a inserção de novos dispositivos em ambientes já automatizados, seja para *retrofiting* ou simplesmente para ampliação do sistema.

1.3 Objetivos do projeto

Este trabalho visa a implementação de dispositivos sensores e atuadores BACnet com comunicação wireless. Para sua validação, o dispositivo implementado deve ser reconhecido como um device BACnet, ter suas propriedades descritas e seus respectivos valores lidos e/ou sobrescritos, quando for o caso.

Os objetivos principais são:

- Implementação do código BACnet a ser embarcado em um microcontrolador;
- Construção de hardware de módulos que atuem como sensores de temperatura e umidade e controladores capazes de atuar em equipamentos de ar condicionado;
- Configuração dos transceivers de comunicação wireless;
- Validação dos dispositivos através da leitura e escrita de suas propriedades, utilizando para tal um software independente padrão de mercado.

1.4 Apresentação do manuscrito

No capítulo 2 são apresentados alguns conceitos essenciais utilizados no trabalho. Em seguida, no capítulo 3 apresenta-se o aparato experimental usado para a concepção dos módulos. O capítulo 4 encarrega-se de apresentar o que foi desenvolvido ao longo do trabalho. Posteriormente, no capítulo 5 são apresentados os resultados obtidos e, por fim, o capítulo 6 apresenta as conclusões do trabalho, bem como as perspectivas para trabalhos futuros.

Capítulo 2

Fundamentação Teórica

O capítulo presente apresenta os principais conceitos utilizados para a realização do trabalho. São abordados conceitos de transmissão de dados e os padrões ZigBee, BACnet e MS/TP.

2.1 Introdução

O capítulo visa a definição de alguns conceitos utilizados a fim de propiciar uma adequada compreensão do desenvolvimento do projeto, ressalta-se que não é escopo do projeto a abordagem aprofundada de conceitos de transmissão de dados e indica-se como referencia o livro [5], *S. Tanenbaum: Redes de Computadores*. Inicialmente, são abordados conceitos básicos de comunicação de dados, bem como uma breve apresentação do padrão OSI. Em seguida, são apresentadas algumas características do protocolo wireless ZigBee. Posteriormente é mostrado em mais detalhes o protocolo BACnet, suas definições e arquitetura. Ao final, apresenta-se o protocolo de comunicação MS/TP (*Master/Slave Token Passing*).

2.2 Conceitos de Transmissão de Dados

Segundo [5], uma **rede** é formada por um conjunto de módulos processadores, capazes de trocar informações e compartilhar recursos, interligados por um sistema de comunicação que é composto por enlaces físicos (meios de transmissão) e de um conjunto de regras com o fim de organizar a comunicação, denominados **protocolos**.

Alguns parâmetros são importantes quando se deseja comparar diferentes sistemas, entre os quais:

- **Retardo de Acesso**: é o intervalo de tempo decorrido desde que uma mensagem a transmitir é gerada por um nó da rede, até o momento que ele consegue acessar o meio para transmitir a mensagem em questão sem que haja colisão no meio;
- **Retardo de Transmissão**: é o intervalo de tempo decorrido desde o início da transmissão de uma mensagem por uma estação de origem até o momento em que ela chega à estação de destino;

- **Retardo de Transferência**: soma do Retardo de Transmissão com o Retardo de Acesso;
- **Desempenho**: é a capacidade efetiva de transmissão da rede. Medidas características: retardo de transferência, fluxo, velocidade, etc. A escolha adequada da arquitetura - estrutura da conexão, protocolo de comunicação e o meio de transmissão influenciam o desempenho, a velocidade e o retardo de transferência de uma rede;
- **Compatibilidade ou Interoperabilidade**: é a capacidade que um sistema possui de se interligar a dispositivos de diferentes fabricantes;

2.2.1 Topologias de Rede

Topologia se refere à forma como os enlaces físicos e os nós da comunicação estão organizados. Pode-se classificar as linhas de comunicação quanto à ligação física em ponto a ponto e multiponto:

- **Ponto a Ponto**: um enlace (linha) que liga dois únicos dispositivos;
- **Multiponto**: três ou mais dispositivos que podem utilizar o mesmo enlace;

As topologias mais utilizadas em LAN's (Local Area Networks) são linhas multiponto do tipo:

- **Estrela**: nessa topologia, um nó central (chamado de Mestre) interliga cada um dos demais nós (chamados de Escravos), assim, todas as mensagens devem passar pelo Mestre. Em algumas configurações o nó central tem a função de gerência de comunicação e facilidade de processamento de dados, noutras apenas gerencia a comunicação e a operação de diagnósticos. Quanto à confiabilidade, observa-se que falhas em um escravo afetam muito pouco o restante do sistema, porém falhas no mestre podem até parar o sistema. Nestes casos, podem se gerar redundâncias e tolerâncias à falhas, aumentando os custos de montagem e manutenção do sistema como um todo. Em geral, o mestre é um hub ou switch, e essa é a topologia mais comumente encontrada em pequenas e médias redes - Figura 2.1.
- **Anel**: consiste em estações conectadas através de um caminho fechado. As configurações mais usuais são unidirecionais a fim de tornar menos sofisticados os protocolos de comunicação (para evitar problemas de roteamento). Quando uma mensagem é enviada por um nó, ela entra no anel e passa por um ou mais nós até que chegue ao nó de destino ou retorna ao remetente, dependendo do protocolo. No primeiro caso cada nó insere um retardo suficiente para checar o endereço e verificar se a mensagem deve ser retirada ou se deve ser repetida. No segundo caso à medida que os bits vão chegando eles vão sendo repetidos, sendo retirados do anel pela estação que gerou a mensagem. Erros podem provocar a circulação de uma mensagem na rede até ser perdida. Uma falha no canal de comunicação entre quaisquer dois elementos do nó irá parar a rede até que a falha seja solucionada. Essas duas topologias caracterizam uma configuração ponto a ponto - Figura 2.2.
- **Barramento**: nessa topologia todos os nós estão ligados ao mesmo meio de transmissão, caracterizando uma configuração multiponto, o que permite que cada nó possa ouvir todas

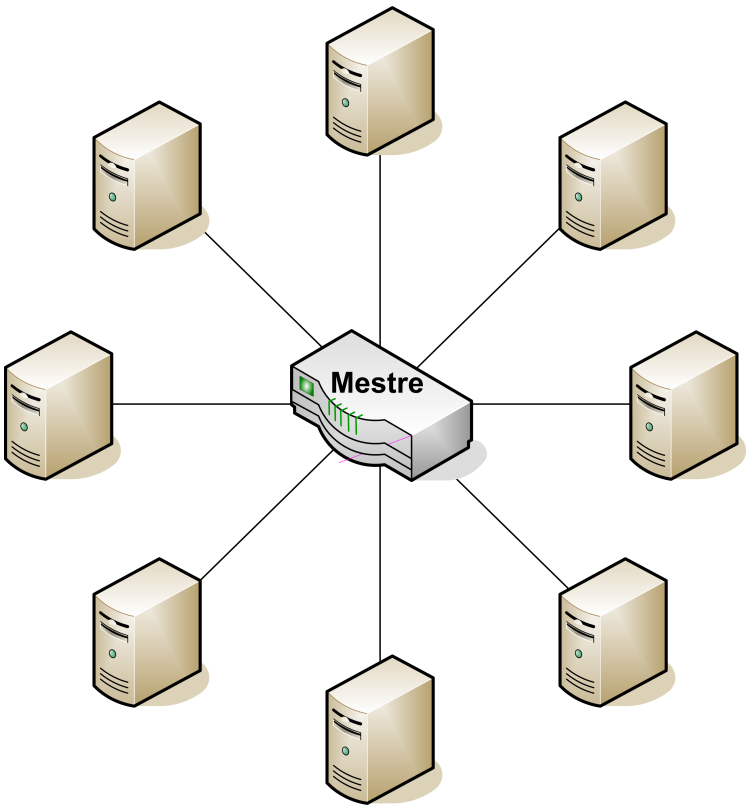

Figura 2.1: Topologia de Rede: Estrela

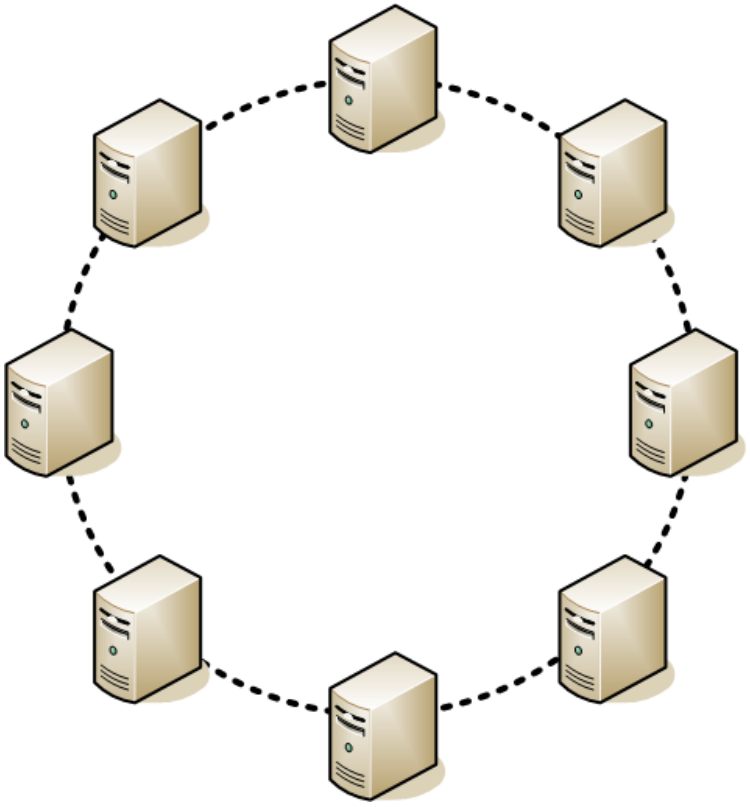

Figura 2.2: Topologia de Rede: Anel

as informações transmitidas. Existem vários métodos de acesso ao barramento, que pode ser centralizado ou não. Quando o controle é centralizado o direito de acesso é determinado por uma das estações, o que traz a mesma limitação da topologia em estrela, com a vantagem que o nó redundante pode ser um outro no da rede. No controle descentralizado a responsabilidade de acesso é distribuída entre todos os nós. As falhas geralmente não provocam a parada total do sistema e mecanismos de prevenção devem ser adicionados para desconectar nós que falhem em modo de transmissão e não parem mais de transmitir. O desempenho será determinado pelo meio de transmissão, pelo número de nós conectados, pelo mecanismo de controle de acesso e pelo tipo de tráfego, entre outros fatores - Figura 2.3.

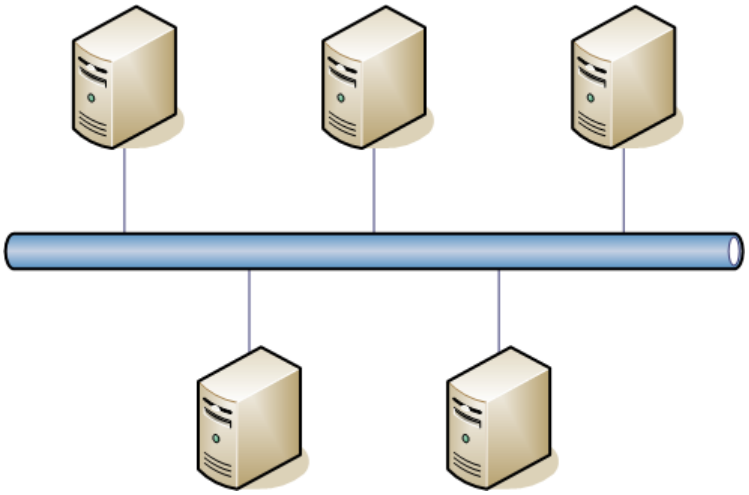

Figura 2.3: Topologia de Rede: Barramento

Quanto ao tipo de comunicação entre as máquinas, podemos ter basicamente duas configurações:

- **Cliente/Servidor**: esse tipo de rede usa uma estrutura rígida para gerenciar e manter os recursos. Alguns computadores são preparados para armazenar e distribuir dados e recursos, sendo denominados servidores, enquanto outros computadores, geralmente mais numerosos, são preparados para acessar e usar os dados e recursos gerenciados pelos servidores - essas máquinas são chamadas clientes. Quando a rede possui mais de um mestre (caso mais raro) ela é denominada multimestre;

- **Peer-to-Peer**: é basicamente uma rede de pares. Toda máquina conectada a rede desse tipo possui os mesmos direitos de acesso de qualquer outra máquina da rede, não existindo um local centralizado para os aplicativos nem um controle central, de modo que a administração deve ser realizada em cada máquina;

2.3 Termos Adaptados de Padrões Internacionais

2.3.1 Padrão ISO OSI

Os termos a seguir são definidos por padrões internacionais (ISO - *Internatinal Stardards Organization*) para sistemas baseados no modelo OSI (*Open Systems Interconections*). Os termos são citados aqui e em seguida uma referência para o padrão associado.

- **Aplicação**: um conjunto de requerimentos para o processamento de informações de um usuário (ISO 8649);
- **Entidade de Aplicação**: os aspectos de um aplicativo pertinente ao OSI (ISO 7498);
- **APDU (Application Protocol Data Unit)**: uma unidade de informação especificada um protocolo de aplicação que consistem em informações que são trocadas por entidades de aplicação para coordenar seus conjuntos de operação (ISO 9545);
- **Sistemas Reais**: um conjunto de um ou mais computadores, o software associado, periféricos, terminais, operadores humanos, processos físicos, etc, que formam um conjunto capaz de executar o processamento de informações e a transferência desses dados (ISO 7498);
- **Usuário de Serviço**: uma entidade num sistema aberto que faz uso de um serviço através de pontos de acesso (ISO TR 8509);
- **Provedor de Serviço**: nome dado a toda e qualquer entidade que disponibilizar algum serviço para usuários de serviço com qualquer nível de acesso (ISO TR 8509);
- **Request(Requisição)**: uma representação de uma interação em que um usuário invoca algum procedimento (ISO TR 8509);

A fim de facilitar o projeto de protocolos de redes, geralmente elas são divididas em camadas ou níveis. O objetivo de cada camada é oferecer serviços para as camadas superiores, acessados através de uma interface padronizada, encapsulando os detalhes de implementação. Cada camada é projetada para resolver funções afins do problema da comunicação.

O modelo de referência OSI é baseado numa proposta desenvolvida pela ISO sendo denominado conforme o título da sub-seção. Seu objetivo é criar uma arquitetura padrão ou modelo de desenvolvimento de protocolos de forma a facilitar o desenvolvimento. Trata-se de um modelo genérico, pois não especifica os serviços e os protocolos que devem ser usados em cada camada, mas é freqüentemente usado como uma arquitetura de implementação de rede. Raramente é implementado na sua totalidade [6]. Na prática, algumas camadas são unidas ou simplesmente suprimidas.

Nesse modelo, a rede é dividida em 7 camadas como pode ser visto na figura a seguir. Um detalhe importante é o retardo ocorrido entre a troca de mensagens entre todas as camadas do modelo. O problema é discutido por diversos autores e Kopetz afirma que:

"(...) as características de uma arquitetura em tempo real sugerem que a arquitetura OSI não é adequada para redes de tempo real com restrições de tempo e field bus." [6].

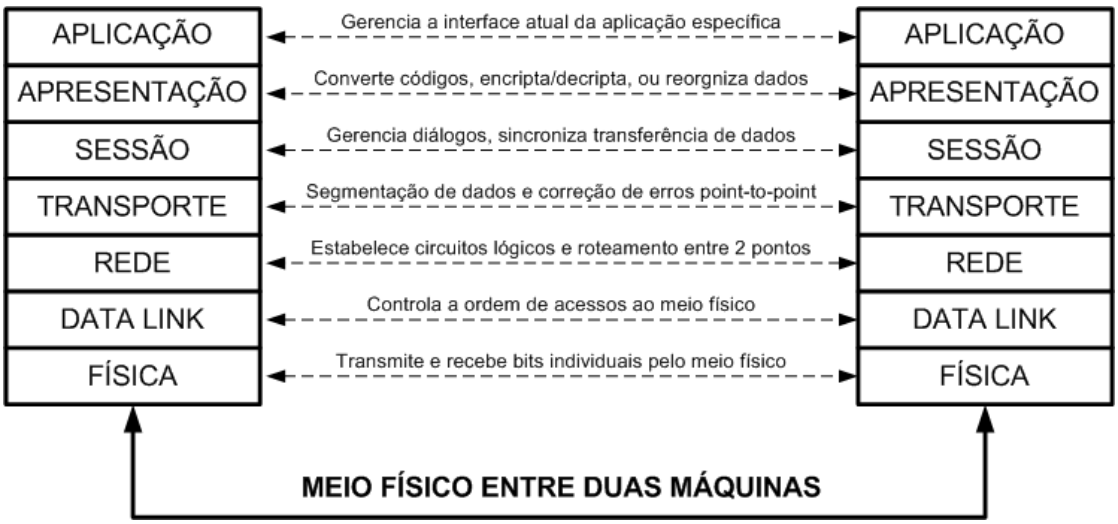

Figura 2.4: Modelo OSI

2.4 O protocolo BACnet - *Building Automation and Control Networks*

Essa seção descreve o protocolo BACnet, apresentando inicialmente seu histórico seguido de suas características principais e de sua arquitetura. Recomenda-se, caso o leitor necessite de informações mais detalhadas não contidas no presente texto, que consulte o documento [1] definidor do protocolo.

2.4.1 Histórico

O BACnet é um protocolo completo de redes de automação predial criado pela ASHRAE e padronizado pela primeira vez em 1995. Foi desenvolvido especificamente para intermediar toda e qualquer comunicação necessária em sistemas de automação e controle em prédios para diversas aplicações como aquecimento, ventilação, ar-condicionado, iluminação, controle de acesso e sistemas de detecção e alarme de incêndio. O protocolo BACnet provê mecanismos pelos quais equipamentos e computadores de funções arbitrárias podem trocar informações, não interessando sua função no prédio ou seu fabricante. Como resultado, o BACnet pode ser utilizado por todo tipo de equipamento, como computadores, palmtops, laptops, gerenciadores complexos de rede, controladores digitais ou simples sensores.

A maior motivação para o surgimento do grupo de estudos para este protocolo e sua posterior padronização foi o acentuado desejo de proprietários de prédios, operadores, fabricantes e prestadores de serviços de interoperabilidade, que, como dito anteriormente, é a habilidade de integrar equipamentos de diversos fabricantes em um sistema coerente de automação e controle e torná-lo perfeitamente competitivo no mercado. Para alcançar esse desafio, o grupo responsável solicitou e ouviu dezenas de fabricantes e usuários individuais; revisou todos os importantes padrões internacionais de comunicação e dispensou horas de debates discutindo os prós e os contras do protocolo. Posteriormente tornou-se um padrão ANSI, em 1995, mas veio a se difundir amplamente somente

quando tornou-se um padrão ISO (ISO 16484-5).

Desde que foi padronizado, o padrão foi traduzido para diversas línguas e vem sendo largamente utilizados por grandes fabricantes e prestadores de serviços na área de automação predial como a Honeywell, Rockwell e Johnson Controles. Um grupo de trabalho é mantido permanentemente por essas empresas para garantir as constantes atualizações do protocolo. Algumas das características que já foram inseridas no protocolo desde sua padronização foram: incremento nas capacidades de interconectar sistemas em diferentes lugares utilizando protocolos de internet; novos objetos e serviços foram adicionados para dar maior suporte a sistemas de detecção e alarmes de incêndio e outras aplicações de bem-estar e segurança; capacidades de realizar backup's e restaurar equipamentos; padronização de formas de se criar gráficos de tendências; novas ferramentas para tornar mais fácil o desenvolvimento; mecanismos para tornar interoperáveis extensões criadas por cada fabricante, novos meios físicos de transmissão, como o ZigBee; e muitos outros.

Todos os protocolos de comunicação são, no final das contas, uma coleção de soluções consagradas para problemas de troca de informações e todos estão sujeitos a mudanças com o tempo e o avanço da tecnologia. O BACnet não é exceção, mas desde sua concepção, ele foi preparado para incorporar grande parte dessas mudanças e por isso tornou-se um dos protocolos mais utilizados e promissores.

2.4.2 Principais Características

O protocolo apresenta como principais características:

1. Todos os equipamentos (exceto os escravos em uma rede MS/TP) são agentes ativos, porém alguns desses agentes podem ter privilégios e responsabilidades maiores que outros;

2. Cada dispositivo na rede é modelado como uma coleção de objetos. Cada objeto é caracterizado por um conjunto de atributos (ou propriedades) e métodos (ou funcionalidades), onde o padrão ISO define alguns e cada fabricante é livre para descrever suas propriedades próprias;

3. A comunicação deve ser realizada por leituras e escritas de objetos particulares e por uma mútua execução de serviços disponibilizados por cada dispositivo, e esses serviços também são padronizados mas outros podem ser desenvolvidos por cada fabricante;

4. Devido ao padrão aderir ao conceito ISO de uma arquitetura de comunicação em camadas, as mesmas mensagens podem ser trocadas utilizando variados tipos de acesso à rede e meios físicos. Isso significa que uma rede BACnet pode ser configurada para uma grande variedade de velocidades e fluxos de rede. Vários tipos de rede BACnet podem ser interconectadas, e essa flexibilidade permite que novas tecnologias possam ser adotadas pelo protocolo futuramente, o que significa uma grande vantagem perante outros protocolos, pois uma vez implementados, mudar toda uma linha de produção ou uma instalação devido a uma nova tecnologia é um grande desperdício.

Dentre os termos já comumente utilizados em aplicações de comunicação de dados, há aqueles específicos do BACnet. Definidos por [1]:

- **Controle de Acesso**: um método para regular ou restringir recursos de uma rede;
- **Alarme**: 1) um anuncio, visual ou sonoro, que alerta o operador de uma condição anormal que pode requerer ações de correção. 2) uma condição anormal detectada por um dispositivo ou um controlador que apresenta uma regra ou uma lógica específica para inspecionar aquela condição;
- **Dispositivo BACnet**: um equipamento/dispositivo, real ou virtual, que suporta comunicação digital usando o protocolo BACnet;
- **Bridge (Ponte)**: um equipamento que conecta dois ou mais segmentos das camadas físicas e data link, do modelo OSI. Esse equipamento também pode filtrar mensagens baseado no endereço da camada MAC;
- **Broadcast**: uma mensagem transmitida como uma única unidade, que se endereça a dois ou mais destinatários;
- **COS (*Change of State* - Mudança de Estado)**: um evento disparado quando um valor binário medido, calculado ou discretamente enumerado muda de estado;
- **COV (*Change of Value* - Mudança de Valor)**: um evento disparado quando um valor analógico medido ou calculado muda de valor de uma quantidade pré-definida;
- **Cliente**: um sistema ou equipamento que faz uso de um outro equipamento com algum propósito particular através de uma instancia de serviço. Um **cliente** faz requisições de serviços de um **servidor**;
- **Contexto**: um conjunto de dados e/ou informações que descrevem completamente um ambiente de comunicação particular em um dado ponto no tempo;
- **Controlador**: um equipamento/dispositivo para a regulação ou gerenciamento de um sistema ou componente;
- **Rede de Conexão Direta**: uma rede que é acessível de um roteador sem que alguma mensagem seja retransmitida por nenhum outro roteador. Uma conexão PTP (*Point to Point*) é para uma rede de conexão direta se a conexão PTP é atualmente ativa e nenhum roteador intervém na conexão;
- ***Download***: um tipo específico de transferência de arquivos que transmite um programa executável ou algum tipo de base de dados a um equipamento remoto onde ela pode ser executada de alguma forma;
- ***Gateway***: um equipamento que conecta duas ou mais redes diferentes, permitindo a troca de informações entre elas;
- **Meio**: um entidade de transmissão física. Por exemplo, par trançado, fibra ótica, cabo coaxial, etc.;
- **Nó**: um equipamento endereçável conectado a um dado meio de comunicação;

- ***Object Profile* - (Perfil de Objeto)**: um meio de definir objetos. Um perfil de objeto define o conjunto de propriedades, comportamentos, e/ou requisitos para objetos proprietários ou para extensões de um objeto padrão;
- **Roteador**: um equipamento que conecta duas ou mais redes na camada de rede do modelo OSI;
- **Servidor**: um sistema ou equipamento que responde a *requests* de serviços com algum propósito particular. Um **servidor** provê serviços a um **cliente**;
- ***Upload***: processo inverso de um *download*. Faz a leitura de um *device*.

2.4.3 Arquitetura do protocolo BACnet

O modelo OSI endereça uma comunicação de computador a computador com uma perspectiva muito genérica, como visto anteriormente. O custo de implementação de um protocolo totalmente conforme o modelo OSI é alto para a maioria das aplicações e geralmente não é necessário. No entanto, ele é um excelente guia de referência para a grande maioria dos protocolos não só de automação. O BACnet foi construído numa arquitetura compacta do modelo OSI, isto é, BACnet utiliza somente algumas das 7 camadas do modelo OSI. Os efeitos das outras camadas para as aplicações a que se destina o BACnet são completamente nulos.

O BACnet é baseado em 4 camadas que correspondem às camadas física, de enlace, de rede e de aplicação no modelo OSI, como pode ser visto na figura 2.5. A primeira padronização Bacnet previa 5 opções de camadas de enlace e de rede. Recentemente, foi adicionado ao padrão a opção de BACnet sobre redes ZigBee. [7]

A camada física, *physical layer*, provê meios de conexão de equipamentos e transmissão elétrica de dados e sinais. Claramente, essa é uma camada necessária ao protocolo.

A camada de enlace, *data link layer*, organiza os dados em frames ou pacotes, regulamenta o acesso ao meio, faz o endereçamento e gerencia alguns erros de transmissão e de fluxo.

As funções previstas na camada de rede incluem a tradução de endereços globais para endereços locais, e vice-versa, o roteamento de mensagens através de diferentes redes, acomodação do tamanho dos pacotes que são permitidos por cada meio de transmissão, sequenciamento, controle de fluxo, controle de erros e multiplexação de mensagens. O BACnet foi desenvolvido para que houvesse um único caminho lógico entre dois equipamentos, evitando assim a necessidade de algoritmos de roteamento ótimos pela rede, o que facilita muito sua implementação. Uma rede é feita de um ou mais dispositivos conectados por repetidores, bridges, gateways, mas com um espaço único de endereçamento local. Para os casos de uma rede simples, muitas das funções da camada de rede são desnecessárias ou simples repetições das funções da camada de enlace, em outros casos no entanto, essas funções se fazem realmente necessárias, como por exemplo o caso em que numa rede BACnet existam dois segmentos de rede que utilizam diferentes camadas físicas, uma rede RS-485 e uma rede sobre ethernet, por exemplo. Nesses casos é necessário diferenciar endereços globais (em toda a rede BACnet) e endereços locais (para as redes de mesmo meio físico). BACnet faz isso

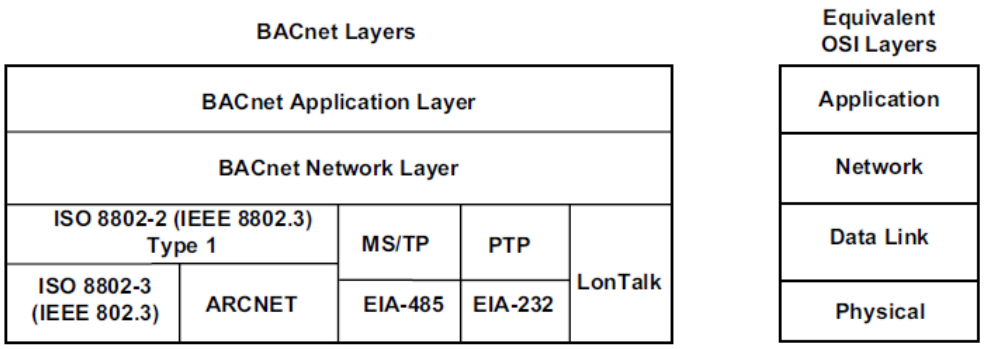

Figura 2.5: Camadas do protocolo BACnet [1].

através da inserção de cabeçalhos da camada de rede nos frames de dados que contém informações de endereçamento e controle de fluxo.

As funcionalidades das camadas de apresentação, sessão e transportes são implementadas na camada de aplicação do BACnet ou são puramente eliminadas por não serem necessárias em alguns casos. A camada de aplicação provê serviços de comunicação requeridos pelas aplicações para executar suas funções.

2.4.3.1 A Camada de Aplicação

A camada de aplicação é responsável pela interação real com o programa aplicativo do equipamento e, como era de se esperar, mantém uma interação ponto a ponto com a camada de aplicação remota. Um diagrama desse modelo pode ser visto na figura 2.6.

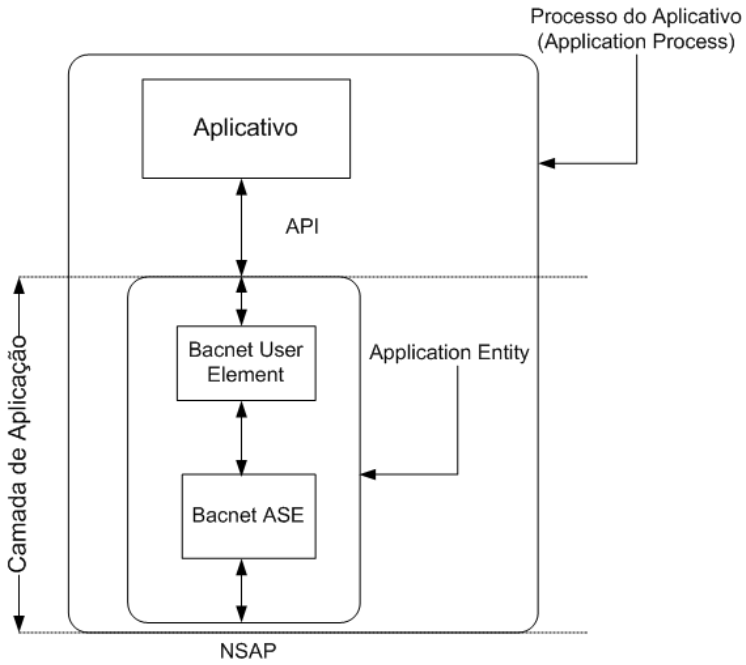

Figura 2.6: Camada de Aplicação Protocolo BACnet [1].

Algumas definições:

- Application Process: funcionalidade que processa a informação necessária para uma aplicação particular;
- Application Entity: relaciona a função de comunicação do protocolo;
- NSAP: Network Service Access Point;
- API: Application Interface, responsável pela interação com o programa, não é definido pelo padrão [1];

O funcionamento da camada de aplicação é basicamente resumido por:

- Receber uma requisição de serviço via API (vinda do aplicativo em si) e gerar uma APDU(*Application Data Unit*) que será repassada à camada de rede para sua execução; ou
- Receber uma APDU da camada de rede , processá-la e repassar as informações necessárias via API ao aplicativo;

Como dito anteriormente, o BACnet é um protocolo com seu funcionamento orientado a serviços. Os serviços primitivos definidos pelo padrão para a camada de aplicação são:

- Request (Requisição);
- Indication (Indicação);
- Response (Resposta);
- Confirm (Confirmação);

A sintaxe utilizada pelo padrão é, ver também a figura 2.7:

- **CONF_SERV:** requisição que exige uma confirmação;
- **UNCONF_SERV:** requisição que não exige uma confirmação;
- **SEGMENT_ACK:** confirmação de um segmento. Pode ser uma confirmação simples (SegmentACK) ou complexa (ComplexACK);
- **ERROR:** indica um erro na requisição;
- **REJECT:** significa que o dispositivo rejeitou o pacote enviado;
- **ABORT:** pede uma finalização forçada da comunicação.

Outro tipo de informação que é trocada com a API são as ICI's (*Interface Control Information's*) que contém as seguintes informações:

- *Destination_Address* (DA): Endereço do device que receberá a primitiva de serviço;
- *Source_Address* (SA): Endereço do qual partiu a primitiva de serviço;

Service Primitive	DA	SA	NP	DER
CONF_SERV.request	Yes	No	Yes	Yes
CONF_SERV.indication	Yes	Yes	Yes	Yes
CONF_SERV.response	Yes	No	Yes	Yes
CONF_SERV.confirm	Yes	Yes	Yes	No
UNCONF_SERV.request	Yes	No	Yes	No
UNCONF_SERV.indication	Yes	Yes	Yes	No
REJECT.request	Yes	No	Yes	No
REJECT.indication	Yes	Yes	Yes	No
SEGMENT_ACK.request	Yes	No	Yes	No
SEGMENT_ACK.indication	Yes	Yes	Yes	No
ABORT.request	Yes	No	Yes	No
ABORT.indication	Yes	Yes	Yes	No

Figura 2.7: Serviços Primitivos [1].

- *Network_ Priority* (NP): Parâmetro de prioridade da rede de quatro níveis;
- *Data_ Expecting_ Reply* (DER): Parâmetro booleano que indica se é esperada uma resposta para o serviço requisitado.

Esses dados são importantes para o controle das informações pelo aplicativo em alguns casos. Na figura 2.8 segue um modelo da pilha BACnet e do fluxo de dados durante uma comunicação.

2.4.3.2 A Camada de Rede

O propósito da camada de rede é prover meios para que as mensagens sejam adequadamende roteadas de uma rede BACnet a outra, encapsulando tais responsabilidades da camada de aplicação e sendo independente da tecnologia utilizada para link dos dados. O datagrama tratado pela camada de rede, chamado NPDU (*Network Package Data Unit*), tem sua estrutura mostrada na figura 2.9.

Cada um dos campos do pacote NPDU é definido como:

- ***Version* - 1 octeto**: versão do protocolo BACnet utilizada;
- ***Control* - 1 octeto**: indica os campos presentes e a prioridade de mensagem;
- **DNET - 2 octetos**: número da rede de destino final;
- **DLEN - 1 octeto**: tamanho do endereço MAC do destino final;
- **DADR - variável**: endereço MAC do destino final;
- **SNET - 2 octetos**: número da rede do destinatário original;
- **SLEN - 1 octeto**: tamanho do endereço da MAC do destinatário original;
- **SADR - variável**: endereço MAC do destinatário original;

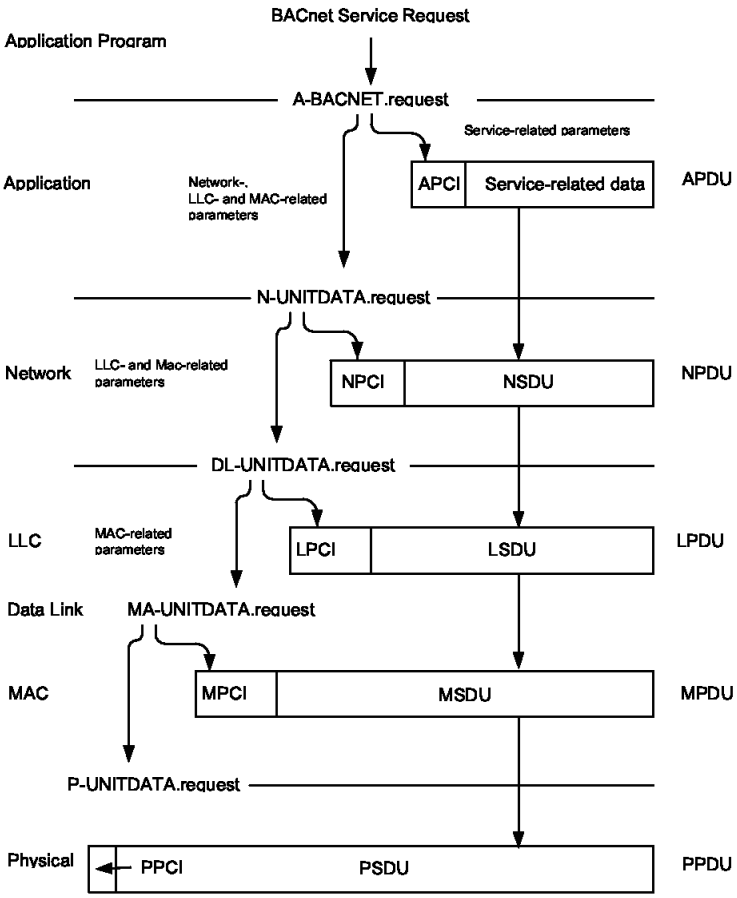

Figura 2.8: Data Flow BACnet,fonte [1]

Version	1 octet
Control	1 octet
DNET	2 octets
DLEN	1 octet
DADR	variable
SNET	2 octets
SLEN	1 octet
SADR	variable
Hop Count	1 octet
Message Type	1 octet
Vendor ID	2 octets
APDU	N octets

Figura 2.9: Estrutura do pacote NPDU [1].

- ***Hop Count* - 1 octeto**: contador utilizado para evitar que uma mensagem enviada a uma rede remota fique em looping.
- ***Message Type* - 1 octeto**: tipo da mensagem trocada entre as camadas de rede para auxiliar no roteamento;
- **Vendor ID - 2 octetos**: identificação do fabricante;

- **APDU - variável**: pacote da camada de aplicação.

Os bits de controle são definidos da seguinte maneira:

- **Bit 7:**

  1: O pacote contém uma mensagem da camada de rede (Message Type presente);

  0: O pacote contém uma BACnet APDU (Message Type ausente);

- **Bit 6:** reservado (deve ser 0);

- **Bit 5:**

  0: DNET, DLEN, DADR e Hop Count ausente;

  1: DNET, DLEN e Hop Count presente;

  Se DLEN = 0: broadcast MAC DADR e DADR está ausente;

  Se DLEN > 0: DADR presente;

- **Bit 4:** reservado (deve ser 0);

- **Bit 3:**

  0: SNET, SLEN e SADR ausente;

  1: SNET, SLEN e SADR presente;

  SLEN = 0 é inválido;

- **Bit 2:**

  Indica se o destinatário está esperando uma confirmação;

- **Bits 1, 0:** prioridade da mensagem na rede:

  B'11': Life safety message;

  B'10': Critical equipment message;

  B'01': Urgent message;

  B'00': Normal message.

2.4.4 Objetos, Propriedades e Serviços

Como dito anteriormente, o protocolo BACnet é totalmente orientado a objetos, e esses objetos definem o comportamento externo de equipamentos e sistemas, ou seja, nenhuma funcionalidade interna de um dado dispositivo é definida. Isso significa que esses objetos não dependem do fabricante ou do equipamento em que estão instalados e que qualquer um que seja capaz de ler objetos BACnet será capaz de ler esses objetos não importando onde ele foi definido. O padrão define 18 tipos de objetos, tabela 2.1 e permite que cada fabricante defina seus próprios objetos, o que não necessariamente os tornaria padrão.

Tabela 2.1: Tipos de objetos BACnet.

Valor	**Descrição**
0	Analog input
1	Analog output
2	Analog value
3	Binary input
4	Binary output
5	Binary value
6	Calendar
7	Command
8	Device
9	Event enrollment
10	File
11	Group
12	Loop
13	Multistate input
14	Multistate output
15	Notification class
16	Program
17	Schedule

Cada objeto contém um grupo de propriedades obrigatórias e um grupo de propriedades optativas, isso é, existem propriedades que devem ser definidas para a existência de um objeto BACnet e outras que podem ser utilizadas ou não. Essas propriedades podem ser lidas ou escritas por dispositivos externos. Foram definidas, ao todo 123 propriedades para os 18 tipos de objetos. Da mesma forma, cada fabricante pode definir suas próprias propriedades, mas isso não garante a interoperabilidade do seu equipamento.

Toda a comunicação entre dispositivos numa rede BACnet é baseada em requisições e respostas de serviços, que são mensagens utilizadas para ler, escrever ou monitorar variáveis, equipamentos, sistemas, tendências e etc. Um equipamento pode disparar um serviços (request) ou reagir a uma requisição (response). Como definido anteriormente, quem faz uma requisição de serviço é chamado de cliente e quem tem os dados a responder e efetivamente os responde é chamado de servidor. Isso caracteriza a arquitetura do protocolo como cliente/servidor. Alguns equipamentos podem ser clientes e servidores, isto é, um dado dispositivo pode ter dados a serem lidos/escritos e/ou pode precisar ler/escrever dados em outros equipamentos. Por exemplo, um atuador inteligente pode ter alguns dados como percentagem de abertura de uma válvula que podem ser lidos/escritos e pode precisar ler os dados de um sensor de temperatura na linha de água gelada, por exemplo.

São 38 serviços disponibilizados em 5 categorias:

- Acesso a objetos;

- Gerenciamento de Equipamentos;
- Alarmes e Eventos;
- Transferência de Arquivos;
- Terminal Virtual;

2.4.4.1 Descrição de Alguns Objetos

São alguns dos objetos mais comuns do BACnet:

- **Analog Input**: Este define um objeto padronizado cujas propriedades representam as características de uma entrada analógica visíveis externamente. Suas principais propriedades obrigatórias são:

  Object_ Identifier: trata-se de um código numérico identificador do objeto e deve ser único na rede para o objeto que ele identifica.

  Object_ Name: identifica o nome do objeto, é do tipo string e seu tamanho mínimo é de um único caracter.

  Object_ Type: define o tipo do objeto, neste caso deve ser do tipo ANALOG_INPUT.

  Present_ Value: indica o valor atual, em unidades de engenharia, do valor medido.

  Status_ Flags: representa os quatro valores booleanos que indicam o status da entrada analógica, podendo ser: IN_ALARM, FAULT, OVERRIDDEN, OUT_OF_SERVICE.

  Event_ State: indica se o objeto tem um evento ativo associado a ele.

  Out_ Of_ Service: trata-se de um valor booleano que indica se o objeto está ou não disponível.

  Units: indica a unidade de medida do objeto.

- **Analog Value**: trata-se de um objeto padronizado indicador do parâmetro valor da variável analógica, o qual reside na memória do device BACnet. Apresenta as mesmas propriedades obrigatórias do Analog Value.

- **Analog Output**: é um objeto que padroniza a saída analógica. Tem as mesmas propriedades obrigatórias já mostradas para o Analog Input, diferindo deste somente por seu valor presente poder ser escrito além de lido.

- **Binary Input**: trata-se de um objeto que padroniza uma entrada digital, cujo valor apresenta somente dois estados distintos. Neste caso, ativo e inativo. Difere da entrada analógica por não apresentar a propriedade de unidades e em seu lugar apresenta a propriedade:

  Polarity: indica o relacionamento entre o valor físico e o valor lógico representado pela propriedade Present_Value.

- **Binary Output**: objeto que formaliza a descrição de uma saída digital. Diferencia da entrada digital por sua propriedade do valor presente poder ser escrito além de lido.

- **Binary Value**: análogo ao objeto de valor analógico, mas agora para valores digitais.

- **Device**: objeto que formaliza as propriedades de um dispositivo BACnet, deve ser sempre único ao longo da rede. Além das propriedades comuns ao objeto entrada analógica, este apresenta também as seguintes propriedades obrigatórias:

  System_ Status: propriedade que reflete o presente estado lógico e físico do device. Os valores possíveis são: OPERATIONAL, OPERATIONAL_READ_ONLY, DOWNLOAD_REQUIRED, DOWNLOAD_IN_PROGRESS, NON_OPERATIONAL.

  Vendor_ Name: string que identifica o fabricante do device.

  Vendor_ Identifier: código identificador do fabricante do device, atribuído pela ASHRAE.

  Model_ Name: utilizado pelo fabricante para identificar o modelo do device.

  Firmware_ Revision: identificador da versão do firmware intalado no device.

  Application_ Software_ Version: identificador da versão do software aplicativo intalado na máquina.

  Protocol_ Version: identificador da versão do protocolo BACnet que é suportada pelo device.

  Protocol_ Revision: identificador da última revisão do padrão BACnet.

  Protocol_ Services_ Supported: indica quais versões dos serviços padronizados do protocolo são suportadas por esta implementação do BACnet.

  Protocol_ Object_ Types_ Supported: indica quais objetos do padrão são suportados pela versão implementada.

  Object_ List: trata-se de um array de identificadores de objetos BACnet, um identificador para cada objeto acessível pelo device.

  Max_ APDU_ Length_ Accepted: indica o numero máximo de octetos que podem estar contidos num único APDU. Este valor deve ser maior ou igual a 50.

  Segmentation_ Supported: indica se o device suporta segmentação de mensagens e quando suporta, na transmissão, na recepção ou em ambos.

  APDU_ Timeout: indicador de qual o intervalo de tempo, em milisegundos, entre as retransmissões de uma requisição de APDU.ack para cada ack não recebido. Seu valor default é 3000 milisegundos.

  Number_ Of_ APDU_ Retries: indica o máximo número de vezes que um APDU deverá ser retransmitido. Seu valor default é 3.

  Device_ Address_ Binding: trata-se de uma lista de BACnetAddressBinding cada um consistindo num identificador de objeto de um device e o endereço do device.

  Database_ Revision: número identificador do database do device. É incrementado quando um objeto é criado, é deletado ou tem seu nome ou alguma propriedade modificada.

2.4.4.2 Descrição de Alguns Serviços

Como dito anteriormente, um dispositivo para ser implementador do padrão BACnet, não necessita implementar todas as suas funcionalidades. O protocolo dispõe de uma extensa lista de objetos, propriedades e serviços. Destaca-se como os mais importantes para este trabalho os serviços: Who-Is, I-Am, WriteProperty e ReadProperty.

- **Who-Is**: é usado para determinar o identificador de um objeto, seu endereço de rede, ou ambos. É um serviço que não requer confirmações, podendo ser utilizado para determinar tais características de todos os devices conectados na rede. Seus argumentos são opcionais e, caso estejam presentes, apenas os devices pertencentes aos limites formados pelos dois parâmetros poderão responder à requisição Who-Is. Sua definição a seguir:

```
/*
** deviceInstanceRangeLowLimit - Opcional - [0..4.194.303]
** deviceInstanceRangeHighLimit -  Opcional - [0..4.194.303]
*/
void Who-Is(
unsigned deviceInstanceRangeLowLimit,
unsigned deviceInstanceRangeHighLimit);
void Who-Is(void);
```

- **I-Am**: trata-se de um serviço de resposta à requisição do serviço Who-Is, é também um serviço que não requer confirmação. Não necessariamente deve responder a uma requisição de Who-Is, em particular, um device pode executar este seviço em broadcast na rede quando for ligado. Sua definição a seguir:

```
/*
** iAmDeviceIdentifier - Obrigatório
** maxAPDULengthAccepted - Obrigatório
** segmentationSupported - Obrigatório
** vendorIdentifier - Obrigatório
*/
void I-Am(
BACnetObjectIdentifier iAmDeviceIdentifier,
unsigned maxAPDULenghtAccepted,
BACnetSegmentation segmentationSupported,
unsigned16 vendorIdentifier);
```

- **WriteProperty**: é um serviço utilizado por um cliente BACnet para modificar o valor de uma propriedade específica de um objeto BACnet. Este serviço potencialmente permite o acesso a escrita a qualquer propriedade de qualquer objeto. Sua estrutura a seguir:

```
/*
```

```
**  Argumentos:
**  objectIdentifier - Obrigatório
**   propertyIdentifier - Obrigatório
**   propertyArrayIndex - Opcional
**   propertyValue - Obrigatório
**   priority - Opcional - [1..16]
**  Retorno:
** Result(+) - indicativo de que a escrita foi feita corretamente
** Result(-) - ErrorType (ErrorClass e ErrorCode)
*/
Result WriteProperty(
BACnetObjectIdentifier objectIdentifier,
BACnetPropertyIdentifier propertyIdentifier,
unsigned propertyArrayIndex,
BACnetObjectValue propertyValue,
integer priority);
```

- **ReadProperty**: este serviço é utilizado por um cliente BACnet para requisitar o valor de uma propriedade de um objeto BACnet. Este serviço permite acesso a leitura a qualquer propriedade de qualquer objeto. Sua estrutura a seguir:

```
/*
**  Argumentos:
**  objectIdentifier - Obrigatório
**   propertyIdentifier - Obrigatório
**   propertyArrayIndex - Opcional
**  Retorno:
** Result(+) - indicativo de que a escrita foi feita corretamente
**    objectIdentifier - Resposta Obrigatória
** propertyIdentifier - Resposta Obrigatória
** propertyArrayIndex - Resposta Opcional
** propertyValue - Resposta Obrigatória
** Result(-) - ErrorType (ErrorClass e ErrorCode)
*/
Result ReadProperty(
BACnetObjectIdentifier objectIdentifier,
BACnetPropertyIdentifier propertyIdentifier,
unsigned propertyArrayIndex);
```

2.5 O Padrão ZigBee

2.5.1 Introdução

As tecnologias de transmissão sem fio não são recentes, pelo contrário. Porém, devido aos altos custos de produção e baixas taxas de transmissão no passado, seu uso foi pouco difundido. Anteriormente, era vista como solução somente para comunicações em longas distancias, via satélite, por exemplo, mas atualmente vem sendo amplamente utilizada em ambientes e distâncias cada vez menores. Parte desse ganho de importância na vida da sociedade vem do maciço investimento das grandes empresas em desenvolver tecnologias para a transmissão sem fio em redes de computadores.

O protocolo ZigBee foi desenvolvido pelo consórcio *ZigBee Alliance*, em conjunto com o IEEE, objetivando a criação de um padrão para comunicação em redes de sensores *wireless*. Alguns dos membros deste consórcio podem ser vistos na figura 2.10.

Figura 2.10: Alguns membros da ZigBee Alliance, fonte [3]

Na presente seção, serão apresentadas as principais características do padrão ZigBee, bem como sua arquitetura em camadas e, por fim, é feita uma comparação entre este padrão, o Bluetooth e o WLAN. Recomenta-se para maiores detalhes não abordados no presente texto, que seja consultado o documento especificador do protocolo [8].

2.5.2 Características Principais

A *ZigBee Alliance* e o IEEE, criaram o protocolo atendendo aos padrões definidos no padrão IEEE 802.15.4, que trata de uma norma que especifica a camada física e a de enlace para as LR-WPAN, *Low-Rate Wireless Personal Area Networks*. O IEEE criou um padrão com a intenção de fornecer as bases fundamentais para o desenvolvimento de redes de baixo custo e baixas taxas de transferência com ênfase na comunicação entre os equipamentos, contrastando com os demais padrões que têm uma abordagem voltada ao usuário final, não importando muito para o quesito consumo.

A idéia é que o ZigBee seja utilizado em equipamentos de baixo custo e que não requerem infra-estrutura elaborada para não elevar o consumo. O ZigBee passou então a ser uma solução excelente na automação predial, onde são desejados dispositivos capazes de se comunicar com baixo custo e baixo consumo.

O ZigBee permite comunicações robustas e opera na freqüência ISM, *Industrial, Scientific and Medical*, no Brasil opera com 16 canais a $2,4GHz$ e não requerem licenciamento específico do órgão regulador. Redes ZigBee oferecem uma certa imunidade a interferências, além de ser capaz de se conectar a mais de 65.000 módulos com taxas de transferência variando de $20Kbps$ e $250Kbps$.

Os módulos RF padrão ZigBee foram criados, como na especificação, para economizar o máximo possível de energia. Sendo assim, pode-se criar sensores que funcionem com baterias ou pilhas, com meses ou até anos de autonomia. Isso se deve ao fato que os módulos ZigBee, quando não estão em modo de transmissão, podem entrar num modulo de dormência, *sleep*, consumindo o mínimo possível de energia. Para se ter uma idéia, quando o módulo ZigBee está em modo ativo, este consome cerca de $50mA$, enquanto que a operação em modo *sleep* consome apenas $50\mu A$ (1000 vezes menor).

O padrão [8] define 5 tipos de dispositivos:

- **FFD - *Full Function Devices***: São dispositivos mais complexos que podem funcionar como coordenadores, roteadores ou até mesmo como dispositivos finais (*end devices*). Esses dispositivos podem se comunicar com qualquer nó da rede. Por causa dessas funções mais complexas, requerem um hardware mais completo e potente para a implementação da pilha de protocolos. Geralmente são implementados em microcontroladores de memória maior, para as tabelas de roteamento etc. Por causa disso, consomem mais energia;
- **RDF - *Reduced Function Device***: São dispositivos com os recursos mínimos de funcionamento, numa rede ZigBee só podem se comunicar com FFD's e assumem invariavelmente o papel de end devices. Um exemplo, na prática, podem ser interruptores, sensores de temperatura, dimerizadores, etc;
- **ZED - *ZigBee End Device***: é onde se hospedam os sensores e atuadores, por exemplo. Pode ser implementado em um FFD ou num RFD e é o dispositivo que menos consome energia. É na pratica o nó comum, aquele que só recebe e envia informações quando solicitado;
- **ZC - *ZigBee Coordinator***: Só pode ser implementado em um FFD. É responsável pela inicialização da rede, distribuição de endereços, manutenção da rede, reconhecimento dos nós, podendo até servir como gateway entre várias outras redes ou estrutura de redes ZigBee;
- **ZR - *ZigBee Router***: Possui as mesmas características de um nó comum na rede, porém com poderes que lhe permite fazer o roteamento entre nós. Por sua complexidade, também só podem ser implementados em um FFD. Na prática, são utilizados para se aumentar o range de alcance da rede num prédio, ou para amplificar o sinal;

O ZigBee apresenta flexibilidade em sua configuração de topologia de rede, podendo ser ponto-a-ponto, ponto-a-multiponto, par-a-par e mista. Na figura 2.11 é exemplificada uma rede mista composta de diferentes classes de dispositivos ZigBee.

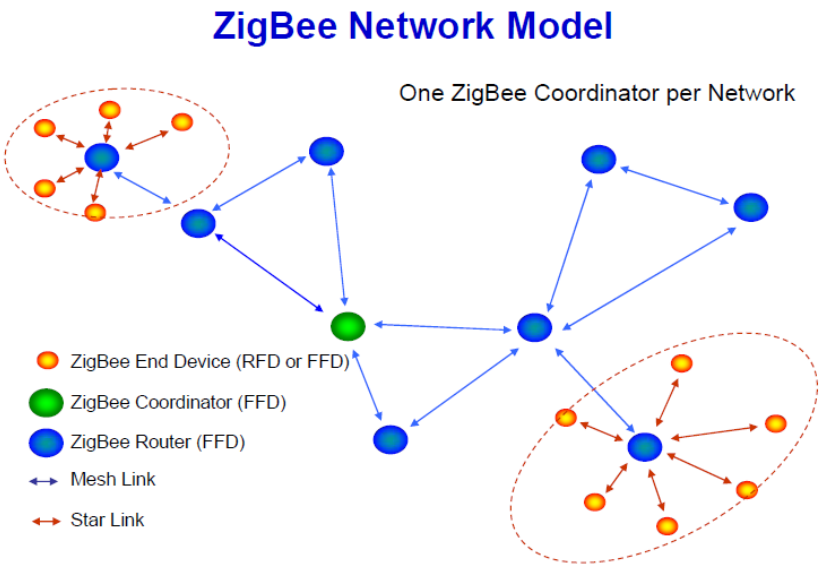

Figura 2.11: Exemplo de Rede Mista ZigBee [3]

Todo e qualquer dispositivo operando numa rede zigbee deve ter um endereço único de 64bits. Esse endereço pode ser utilizado para comunicação direta dentro da PAN, ou um endereço mais curto pode ser definido pelo coordenador da PAN quando o dispositivo já foi identificado e está em uso. Cada PAN independente utiliza um único identificador. Esse identificador permite a comunicação entre os dispositivos dessa rede usando endereços curtos, análogo a apelidos, e habilita a transmissão de dados entre equipamentos de PAN's diferentes.

A rede pode operar em dois modos:

- *Beaconing*: Nesse modo, os roteadores da rede transmitem de tempos em tempos um sinal de *beaconing* (sinalização) para os outros roteadores da mesma rede para confirmar sua presença. Nesse modo, a maioria dos dispositivos finais opera no modo *sleep*. Neste caso, geralmente são utilizados os chamados coordenadores de *sleep*, responsáveis por armazenar em buffer solicitações que chegam enquanto o sensor encontra-se em modo *sleep* e, ao entrar em modo ativo, o coordenador envia o buffer contendo as solicitações acumuladas.
- *Non-Beaconing*: Nesse modo todos os dispositivos da rede permanecem com seus receptores sempre ativos, o que os leva a consumir mais energia;

Algumas outras vantagens do ZigBee que merecem ser citadas são: possibilita a configuração automática da rede, endereçamento dinâmico dos módulos escravos e possibilidade da utilização de criptografia AES de 128 bits.

2.5.3 Arquitetura do Protocolo

Como afirmado anteriormente, a norma IEEE 802.15.4 é responsável por definir a camada física e a camada de acesso ao meio. Para um módulo trabalhar no protocolo ZigBee, ele deve obrigatóriamente ter um microcontrolador que implementa a camada MAC e também um rádio

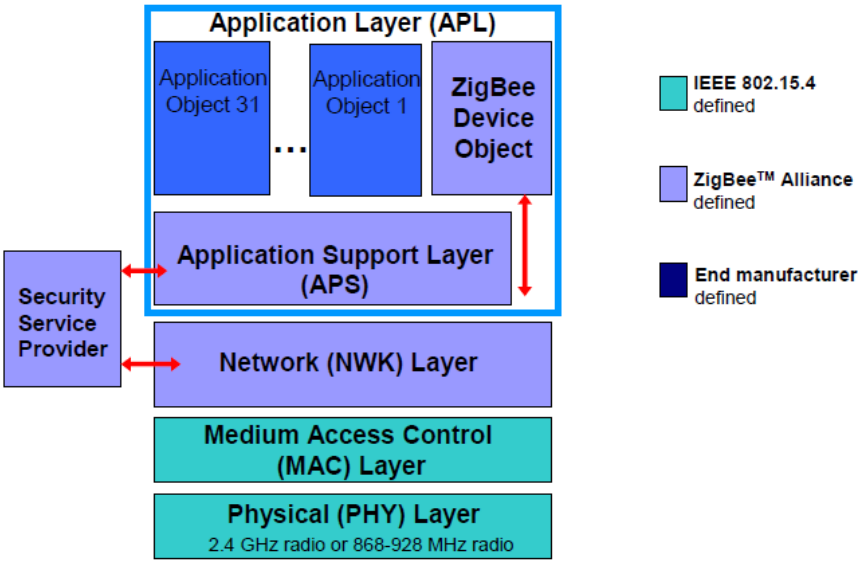

Figura 2.12: Arquitetura do ZigBee [3]

transmissor/receptor IEEE 802.15.4. O documento [8] padroniza somente o framework da camada de aplicação e o funcionamento da camada de rede. A figura 2.12 retrata bem a arquitetura do protocolo.

A camada física (PHY) provê, em ultima análise, os serviços de transmissão de dados. Assim, PHY gere o transceptor físico de RF e faz a seleção do canal de transmissão e as funções de gerenciamento de sinal e energia, sendo projetada para fornecer um alto grau de integração entre os módulos, além da utilização de equipamentos simples e baratos.

A camada MAC permite a transmissão dos frames MAC através do canal físico. Por trás dos serviços de dados, também oferece uma interface de gerenciamento e ela mesmo gerencia o canal de acesso à camada física. Foi projetada para prover alto desempenho com baixo consumo e simplicidade, diminuindo os custos em memória de programa. Esta camada permite a utilização de topologias múltiplas de rede e o controle de um grande número de dispositivos sem a necessidade de operações complexas. Além disso, permite a utilização do padrão AES como algoritmo de criptografia (entretanto, seu controle é feito pelas camadas superiores). [2]

Entre as outras camadas implementadas pelo padrão, merece destaque a camada de rede/segurança projetada para operar com grandes quantidades de nós de baixa potência e latências relativamente baixas [9]. A pilha de protocolos do padrão permite o balanceamento dos custos em unidades específicas, reduzindo o consumo de energia e fornecendo soluções de melhor custo-desempenho para as aplicações. [2]

O frame é a unidade básica de transporte de dados numa rede ZigBee, dos quais existem quatro tipos fundamentais (dados, reconhecimento, sinalização e MAC), o que proporciona um fluxo razoável com simplicidade e robustez. Adicionalmente, como foi mencionado, o coordenador pode enviar superframes para coordenar a rede (beacons). Esses superframes são geralmente maiores que os frames normais, e quando os limites normais de tamanho são atingidos eles são resolvidos por CSMA/CA. Toda transmissão deve terminar antes de chegar o próximo beacon.

2.5.4 Comparação entre alguns protocolos *wireless*

O protocolo ZigBee apresentou-se interessante devido às suas características propícias à sua utilização em automação predial. No entando, buscando visualizar melhor tais parâmetros, é interessante fazer a comparação do ZigBee com outros protocolos *wireless* concorrentes a ele. Como pode ser visto na tabela 2.2, o ZigBee apresenta um consumo bem reduzido em relação ao seu alcance e taxa de transferência.

	ZigBee (WPAN)	**Bluetooth (WLAN/WPAN)**	**Wi-Fi (WLAN)**
Padrão	802.15.4	802.15.1	802.11
Transferência	250kbps	1Mbps	54Mbps
Consumo(Tx)	35mA	40mA	400+mA
Consumo(Standby)	$3\mu A$	$200\mu A$	$20\mu A$
Memória	32-60KB	100+KB	100+KB
Alcance Indoor	30m	5m	50m
Alcance Outdoor	100m	10m	350m
Topologia	Mesh	Ponto-a-multiponto	Ponto-a-multiponto

Tabela 2.2: Tabela comparativa entre tecnologias wireless.

2.6 O padrão de comunicação MS/TP

2.6.1 Uma breve descrição do MS/TP

O BACnet é baseado numa arquitetura de quatro camadas, como apresentado anteriormente na figura 2.5, que correspondem às camadas de aplicação, de rede, de link de dados e camada física. O BACnet adota o protocolo MS/TP como um dos seus protocolos de rede local, onde é utilizado para fazer a conexão entre os dispositivos de campo.

O protocolo MS/TP, *Master-Slave/Token-Passing*, foi desenvolvido especificamente para automação predial e sistemas de controle a serem implementados usando um único microprocessador com a UART, *Universal Asynchronous Receiver/Transmitter*. As redes MS/TP podem ser configuradas como uma rede *master/slave*, uma rede *peer-to-peer token passing*, ou uma rede mista composta de ambas. Existem dois diferentes gêneros de nós de não-reciprocidade em redes MS/TP, os nós mestres e os escravos. O token é transmitido com a finalidade de regular o acesso ao meio, circulando de um nó mestre a outros de acordo com o endereço lógico da rede. Um nó mestre pode transmitir mensagens aos demais nós mestres ou escravos somente quando é mantenedor do token, caso contrário, estará em modo de recepção e monitoramento. Um nó mestre que tenha o token, pode transmitir mensagens de tamanho limitado configurado pela rede através do parâmetro $N_{max\_info\_frames}$.

Um nó mestre transmite um frame *Poll_For_Master* objetivando encontrar um outro nó mestre que queira conectar-se ao anel após receber o token 50 vezes. Os nós escravos nunca se tornam mantenedores do token, estes somente retornam uma mensagem de resposta quando recebem um

pedido de algum mestre [10].

2.7 Análise

O protocolo BACnet possui diversas funcionalidades, afinal foi definido para atender a todo tipo de serviço imaginável num prédio. No entanto, na maioria das vezes não é necessário implementá-lo por completo em um dispositivo, é possível implementar somente a parte que interessar com os objetos necessários. Essa é a grande vantagem do protocolo orientado a objetos. Cada dispositivo é visto como uma caixa preta com vários objetos dentro. Essa caixa pode ser enorme e caber vários objetos e funcionalidades, ou pode ter o tamanho ideal para um só objeto. E se for necessária a ampliação dessa caixa isso é facilmente implementável.

Devido a essa flexibilidade nas funcionalidades do protocolo e em sua implementação, que não depende de um chip específico como o LonWorks, o potocolo BACnet se mostrou o mais adequado para solucionar o problema proposto. Não só pelas funcionalidades técnicas já apresentadas, mas também por sua enorme aceitação no mercado, que vem crescendo vertiginosamente mais a cada dia que passa. Por ser um protocolo aberto e de fácil integração, a grande maioria dos fabricantes de equipamentos e serviços em automação predial no mundo, não só de controladores, softwares e sensores, como também painéis de incêndio, medidores de energia, de água etc, a grande maioria desses fabricantes tem investido e muito nessa área, tanto em pessoal quanto em equipamentos.

Portanto, a utilização desse protocolo no projeto de automação visando a economia de energia, provavelmente em casos de retroffiting, como é o caso desse pojeto, demonstra-se perfeitamente viável e economicamente plausível, já que os equipamentos e softwares produzidos sob essa linha de pensamento podem ser facilmente comercializados.

Portanto, a melhor solução encontrada e apresentada é a implementação do protocolo BACnet sobre redes ZigBee. Isso não é um fato novo, existe um grande grupo de estudo da ASHRAE que padronizou no início deste ano essa implementação, justamente por esses motivos. Pode-se afirmar com alguma certeza, que redes ZigBee nasceram para redes BACnet e vice-versa.

Capítulo 3

Aparato Experimental

Este capítulo apresenta os componentes utilizados no decorrer dos trabalhos, abordando componentes de hardware, além dos softwares de desenvolvimento.

3.1 O Transceiver XBee

Módulo de comunicação utilizado, implementador do padrão ZigBee. É fabricado pela Digi International, empresa especializada em tecnologias *wireless* e integrante da *ZigBee Alliance*. Para informações mais detalhadas não encontradas no presente texto, recomenda-se consultar o manual completo do XBee [4].

3.1.1 Características Principais

O XBee, figura 3.1, é um dispositivo de baixo custo e de utilização relativamente simples. Apresenta dimensões reduzidas e baixo consumo. Características que atendem bem ao requerido pela implementação de dispositivos móveis voltados a automação predial. Estes dispositivos podem ser interfaceados com outros, como PC's e microcontroladores, através da sua comunicação serial. Ele pode, ainda, conectar-se a outros dispositivos de fabricantes diversos, desde que todos implementem o IEEE 802.15.4. Sua dimensão e pinagem são mostrados na figura 3.2. As especificações dos pinos são descritas na figura 3.3.

Figura 3.1: Módulo XBee [4]

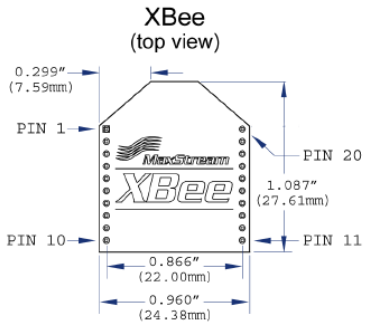

Figura 3.2: Dimensões e pinagem do XBee [4]

Pin #	Name	Direction	Description
1	VCC	-	Power supply
2	DOUT	Output	UART Data Out
3	DIN / CONFIG	Input	UART Data In
4	DO8*	Output	Digital Output 8
5	$\overline{\text{RESET}}$	Input	Module Reset (reset pulse must be at least 200 ns)
6	PWM0 / RSSI	Output	PWM Output 0 / RX Signal Strength Indicator
7	PWM1	Output	PWM Output 1
8	[reserved]	-	Do not connect
9	$\overline{\text{DTR}}$ / SLEEP_RQ / DI8	Input	Pin Sleep Control Line or Digital Input 8
10	GND	-	Ground
11	AD4 / DIO4	Either	Analog Input 4 or Digital I/O 4
12	$\overline{\text{CTS}}$ / DIO7	Either	Clear-to-Send Flow Control or Digital I/O 7
13	ON / $\overline{\text{SLEEP}}$	Output	Module Status Indicator
14	VREF	Input	Voltage Reference for A/D Inputs
15	Associate / AD5 / DIO5	Either	Associated Indicator, Analog Input 5 or Digital I/O 5
16	$\overline{\text{RTS}}$ / AD6 / DIO6	Either	Request-to-Send Flow Control, Analog Input 6 or Digital I/O 6
17	AD3 / DIO3	Either	Analog Input 3 or Digital I/O 3
18	AD2 / DIO2	Either	Analog Input 2 or Digital I/O 2
19	AD1 / DIO1	Either	Analog Input 1 or Digital I/O 1
20	AD0 / DIO0	Either	Analog Input 0 or Digital I/O 0

Figura 3.3: Descrição dos pinos do XBee [4]

Interessante observar que seu alcance em ambientes internos é de até 30 metros e seu maior consumo de corrente é de 50mA, quando está em modo receive e alimentado a 3,3V. O XBee apresenta como principais características de desempenho as seguintes [11]:

- *Alcance Indoor*: até 30m;
- *Alcance Outdoor*: até 100m;
- *Potência Máxima de Transmissão*: 1mW (0 dBm);
- *Taxa de Dados Interface Serial*: até 115,2 Kbps;
- *Taxa de Dados de RF*: 250 Kbps;
- *Sensibilidade do Receptor*: -92 dBm;
- *Tensão de Alimentação*: 2,8 - 3,4 V;
- *Corrente de Transmissão*: 45mA (@ 3,3V);

- *Corrente de Recepção*: 50mA (@ 3,3V);
- *Corrente em modo sleep*: $< 10\mu A$.

Os módulos XBee utilizados implementam somente o padrão IEEE 802.15.4, como dito anteriormente, definidor das camadas MAC e PHY da pilha ZigBee. Atualmente a empresa Digi já fabrica módulos XBee's mais modernos e que implementam mais camadas do protocolo, além das MAC e PHY.

3.1.1.1 Modos de Operação do XBee [2]

- *Modo Transparente*: modo padrão de operação do módulo, neste modo o dispositivo trabalha simplesmente como transmissor de dados seriais; todo o dado inserido no pino de entrada serial será transmitido pela antena, e todo o dado recebido pela antena é enviado pelo pino de saída da serial;
- *Modo de Comando*: permite acesso às variáveis de configuração do módulo. Inserindo a sequência de caracteres ("+++") no terminal, o XBee entra neste modo onde seus parâmetros podem ser monitorados ou reconfigurados por meio de comandos como: ATMY, ATDL, ATBD, ATID, dentre muitos outros.
- *Modo Sleep*: modo de grande importância em sistemas de automação predial *wireless*, devido o consumo do dispositivo cair cerca de 1000 vezes, o que propicia a implementação de dispositivos totalmente independentes de fios, até mesmo para sua alimentação, que pode ser com pequenas baterias. O módulo pode ser acordado via hardware, por meio do pino 9, ou via configuração prévia pelo modo de comando. Até mesmo ser subordinado a um outro módulo XBEE chamado Coordenador de Sleep que lhe enviará os dados a ele endereçados quando sair do modo de dormência.
- *Modo API*: *Application Programming Interface*, é uma outra opção alternativa à operação em modo transparente. Diferenciam-se somente em relação ao modo como os dados são transmitidos, nesse caso passando a ser frames especiais que possuem dados adicionais facilitando o roteamento dos pacotes.

3.1.2 Software de Configuração

A fabricante Digi disponibiliza em seu site o software configurador do módulo XBee. O software X-CTU apresenta uma interface (mostrada na figura 3.4) que possibilita a configuração do módulo pelo seu terminal, bem como a atualização do seu *firmware*. Para mais detalhes sobre o software, consulte [4] ou o site *http://www.digi.com/support/*.

3.2 Hardware de Interfaceamento - CON-USBBEE

A empresa Rogercom Com. e Serv. de Informática Ltda, *www.rogercom.com*, trabalha no desenvolvimento de dispositivos para trabalhar com o protocolo ZigBee. Entre tais equipamentos

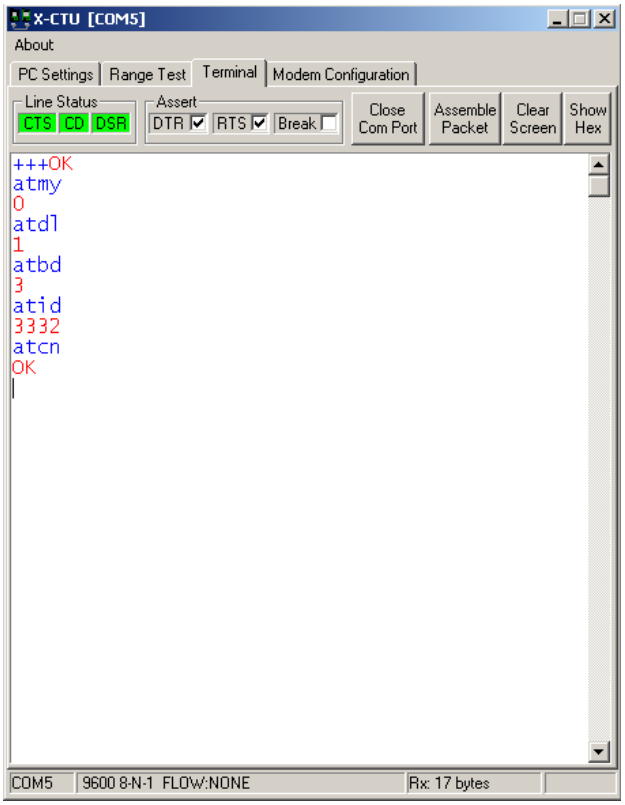

Figura 3.4: Software de configuração dos módulos XBEE's - XCTU

encontra-se a placa CON-USBBEE, figura 3.5, que trata-se de uma interface USB do módulo XBee com o PC, criando uma porta serial virtual. Este dispositivo facilitou, e muito, a configuração dos módulos XBee's além de ser usado pelo PC que monitora, supervisiona os devices BACnet-ZigBee. Esta placa também dispõe de LED's indicadores da intensidade do sinal, figura 3.6.

Figura 3.5: Placa CON-USBBee

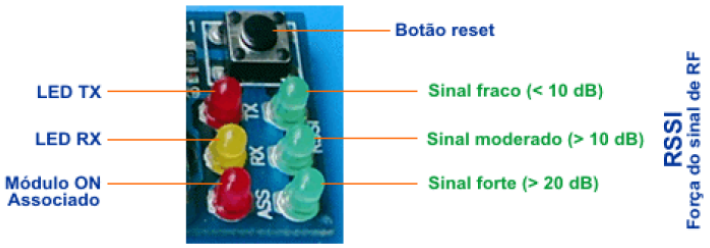

Figura 3.6: LED's indicadores da CON-USBBee

3.3 Microcontroladores ATMega

3.3.1 Características

Os microcontroladores utilizados tanto no módulo sensor como no atuador são da linha ATmega, fabricados pela empresa Atmel. Foram escolhidos pelo fato de apresentarem baixo consumo, baixo custo, flexibilidade de desenvolvimento, extensa biblioteca de desenvolvimento e simplicidade de configuração. Utilizou-se os microcontroladores ATmega168, para o módulo sensor, e o ATmega32, para módulo atuador. A diferença principal entre ambos a ser destacada é a capacidade da memória de programa, 16KB e 32KB respectivamente. Em necessário mais informações acerca dos microcontroladores, consultar [12] ou [13].

Tratam-se de microcontroladores de arquitetura 8-bits de tecnologia CMOS e arquitetura RISC, podendo executar uma instrução por ciclo de clock. Apresentam biblioteca de desenvolvimento que auxilia o projeto em alto nível de linguagem. Além disso, apresentam vários periféricos integrados dentro de seu encapsulamento, o ATmega168 apresenta 6 canais de conversores AD onde 4 são com 10 bits de resolução e 2 são com 8 bits; porta serial programável USART; oscilador interno de 1MHz; encapsulamento 28-pinos PDIP (figura 3.7). O ATmega32 apresenta: oscilador interno de 1MHz; porta serial programável USART; 4 canais de PWM; 8 canais de conversores AD de 10 bits; encapsulamento 40-pinos PDIP (figura 3.8).

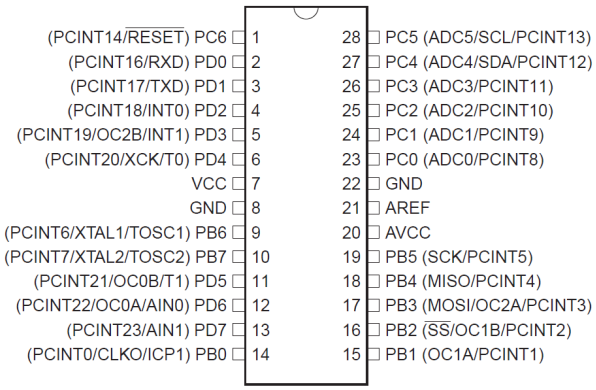

Figura 3.7: Pinagem do ATmega168

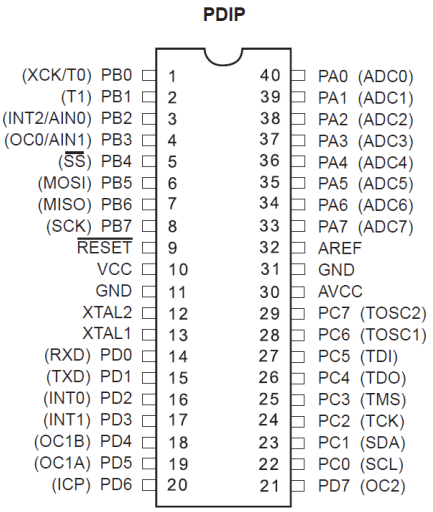

Figura 3.8: Pinagem do Atmega32

3.3.2 Software de Desenvolvimento

Para auxiliar no desenvolvimento, baseou-se nas orientações contidas no documento de notas técnicas [14]. O CD de desenvolvimento pode ser encontrado em: *http://www.ene.unb.br/ gaborges/recursos/embarcados/avr/AVR.zip*. Nele encontram-se alguns programas exemplos, documentos sobre os microcontroladores e também programas-ferramenta. Um deles trata-se do WinAVR, que permite programação em linguagem C ou *Assembly*. Os executáveis mais utilizados foram o avrdude.exe e o avrdude-gui.exe, que permitem fazer o *upload* do programa para a memória do microcontrolador, bem como configuração dos seus registradores, como os *fuses* que configuram o clock de processamento. Como ambiente de desenvolvimento utilizou-se o IDE NetBeans devido ser gratuito e já ser um software renomado e em constante aprimoramento, além de propiciar melhor ambiente para programação modular e também apresentar uma versão apropriada à linguagem de programação utilizada. O ambiente de desenvolvimento utilizado pode ser visto na figura 3.9. Para maiores informações sobre o NetBeans recomenda-se acesso ao site *http://www.netbeans.org/*.

3.4 O Software CAS BACnet Explorer

Devido à sua simplicidade de operação e ao seu atendimento dos requisitos necessários, utilizou-se o software CAS como ferramenta de supervisão de funcionamento dos módulos. Nele são permitidas configurações como: Habilitação de rede BACnet MS/TP, BACnet IP ou BACnet Ethernet. Após a seleção do tipo de rede BACnet a ser monitorada, configura-se quais os objetos BACnet a serem procurados no device. Terminada a configuração e conectado o módulo receptor, o CAS faz uma varredura na rede procurando por dispositivos BACnet listando, em seguida, os dispositivos e seus objetos encontrados (*Read Properties*), podendo ser configurado para uma varredura constantemente atualizada, se necessário.

O CAS explorer, em caso de monitoramento de objetos de saída como uma analógica ou digital, permite que seu valor seja lido (*Read Property*) e alterado enviando assim uma solicitação de escrita

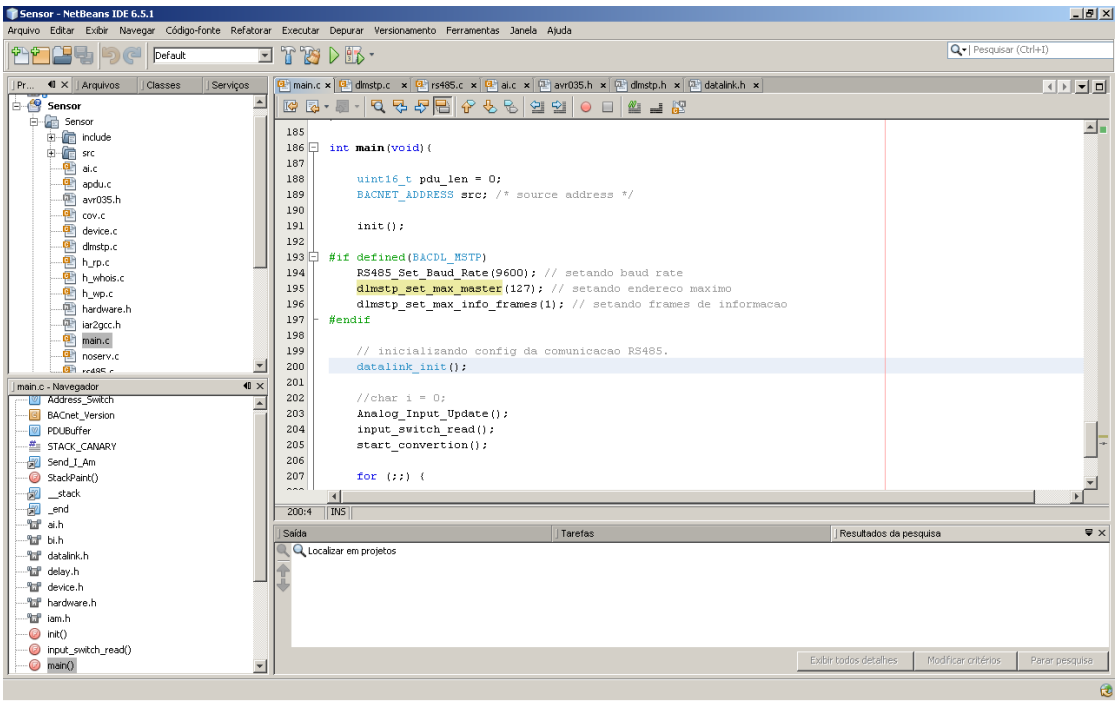

Figura 3.9: Ambiente de desenvolvimento - IDE Netbeans

de propriedade, ou seja, *Write Property*. O software possibilita também a criação de uma lista de valores a serem monitorados para serem constantemente atualizados. A figura 3.10 apresenta uma parte da tabela de propriedades que podem ser lidas pelo software.

Figura 3.10: CAS BACnet Explorer - Tabela de Propriedades

Em 3.11 encontram-se as configurações relativas à rede, ao padrão que será utilizado. Selecio-

nando uma das opções, tem-se também as configurações de comunicação. Caso a opção escolhida seja o MSTP, teria-se as opções mostradas em 3.12: Porta COM, Baud Rate, Endereço MAC.

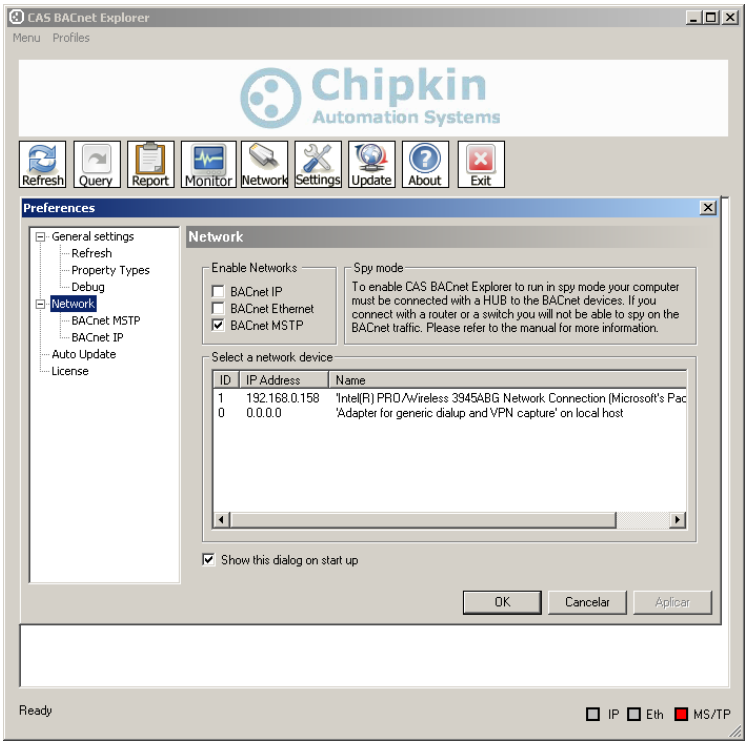

Figura 3.11: CAS BACnet Explorer - Configuração de Rede

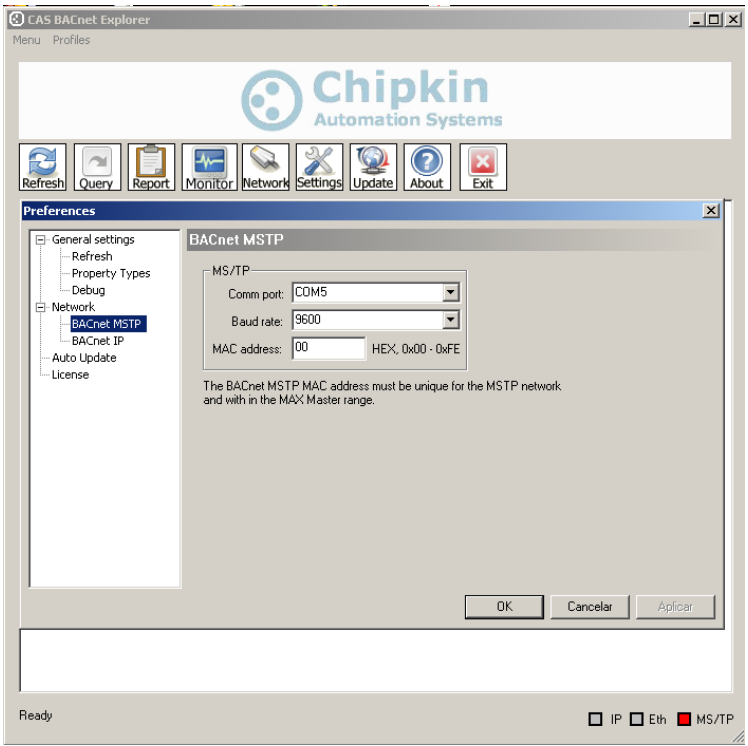

Figura 3.12: CAS BACnet Explorer - Configuração de Rede

Capítulo 4

Dispositivos BACnet-ZigBee

Neste capítulo serão tratados os módulos desenvolvidos durante os trabalhos. Apresenta-se o módulo sensor e o módulo atuador BACnet-ZigBee, bem como suas principais características físicas e virtuais, além de comentários sobre suas aplicações.

4.1 Dispositivo Sensor

4.1.1 Características Principais

A um dos dispositivos desenvolvidos, chamou-se de Módulo ou Dispositivo Sensor BACnet-ZigBee. Trata-se de um equipamento construído com o intuito de ser aplicado em projetos de automação de ambientes prediais. Aplicação voltada a uma grande variedade de projetos, sejam de custo elevado ou reduzido. Seu maior diferencial é a independência de fiação, o que traz inúmeros benefícios como: a utilização em projetos já implementados e estruturados fisicamente, que demandariam um custo considerável de mão-de-obra para inclusão de novos equipamentos, expansão e modernização do projeto; manutenção facilitada, já que é possível o deslocamento do dispositivo para outro ambiente mais adequado; flexibilidade na disposição dos devices pois permite alteração do local físico se necessário. Os dispositivos desenvolvidos são alimentados por baterias de 9V, mas podem também ser alimentados com fontes de até 12V.

4.1.1.1 Hardware

O dispositivo sensor protótipo foi implementado em placa de conexões universais, possibilitando assim a inserção de mais componentes eletrônicos em estudos futuros. O dispositivo é um device BACnet com, uma entrada analógica (AI) que é o sinal da saída do sensor de temperatura LM35. Podendo ser configurado para ter outra AI, para sensor de umidade, por exemplo.

Para versão didática foram implementados 2 módulos sensores, sendo um na placa universal e 1 na placa impressa (figura 4.4). O circuito implementado no módulo sensor (figura 4.1) consiste, basicamente, na conexão requerida para possibilitar gravação BSD no módulo, conexão entre o AVR e o XBEE para comunicação serial com RTS, LED's indicativos de Tx (Verde) e Rx (Vermelho) e

a conexão da saída do LM35DZ ao conversor ADC, passando por um amplificador não-inversor de ganho 4,9. Lembrando que a alimentação se dá pela bateria de 9V e sendo convertida a 3,3V pelo LM1117.

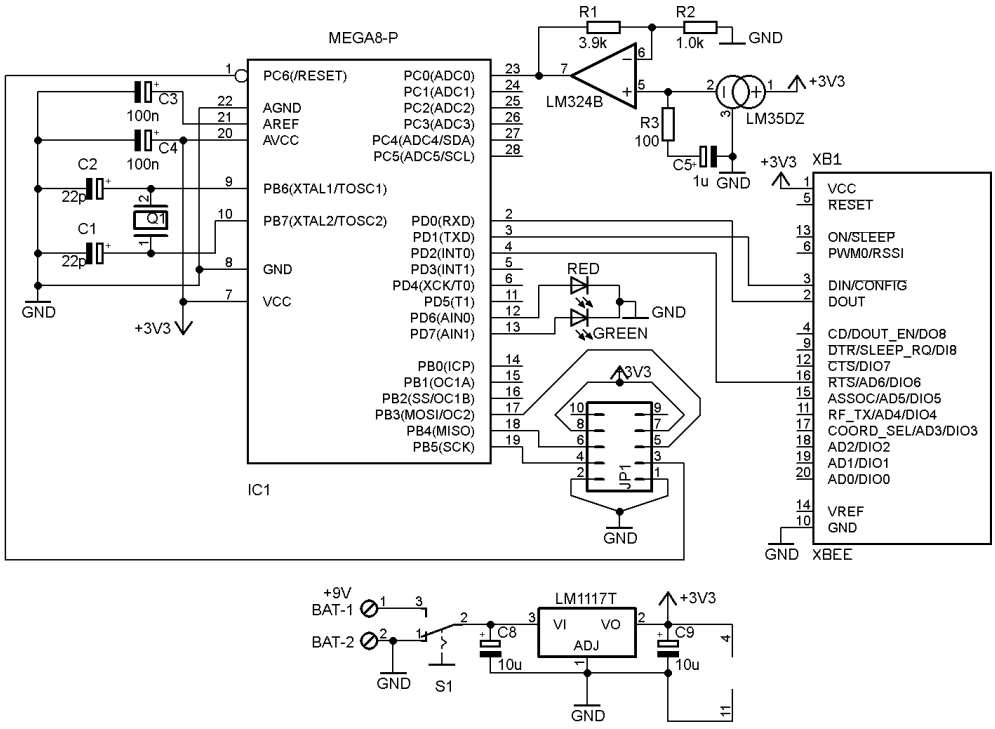

Figura 4.1: Módulo Sensor - Circuito Implementado. Eagle 5.4.0

Os componentes utilizados para sua implementação foram:

- 1 ATmega 168;
- CI's: LM1117, LM35DZ, LM324;
- 1 XBEE;
- 1 Cristal de $4,9152MHz$;
- Capacitores: 2 de $100nF$, 2 de $22pF$, 2 de $10\mu F$ e 1 de $1\mu F$;
- Resistores: 1 de $3,9k\Omega$, 1 de $1,0k\Omega$ e 1 de 100Ω;
- Componentes: 1 Bateria 9V, 1 conector de baterias, 2 LED's, 1 socket, 1 botão, 1 chave on/off e 1 conector header de 10 pinos.

Apresenta-se também, o layout da placa impressa na figura 4.2 e sua vista inferior na figura 4.3.

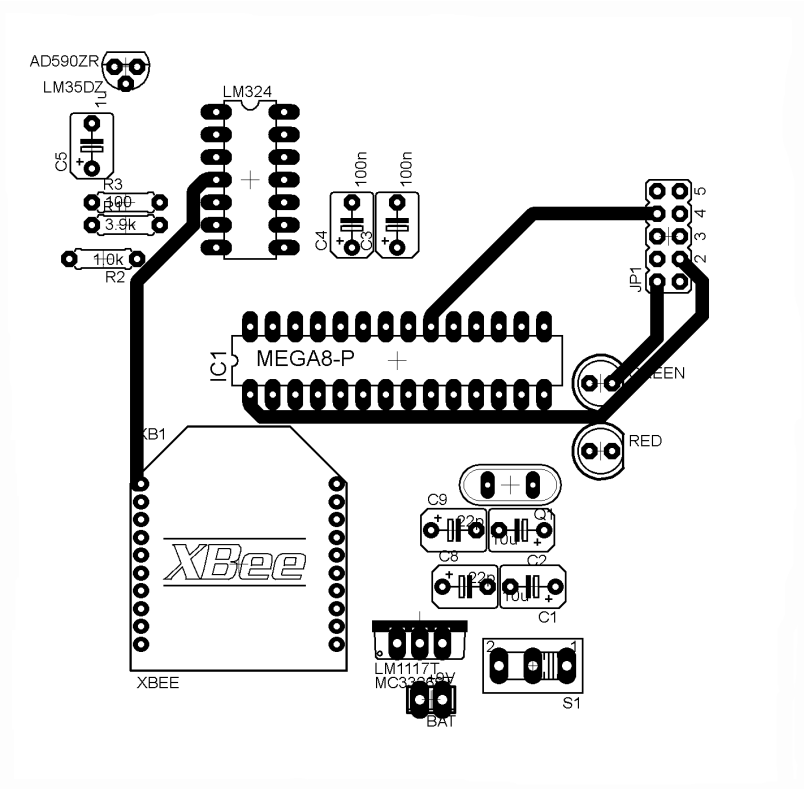

Figura 4.2: Módulo Sensor - Layout Vista Superior. Eagle 5.4.0

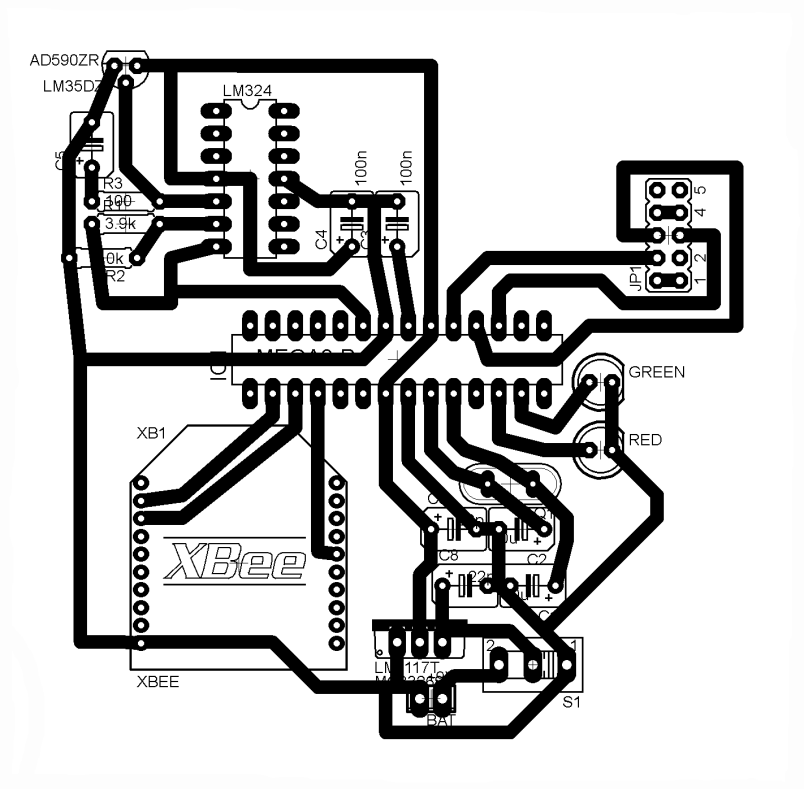

Figura 4.3: Módulo Sensor - Layout Vista Inferior. Eagle 5.4.0

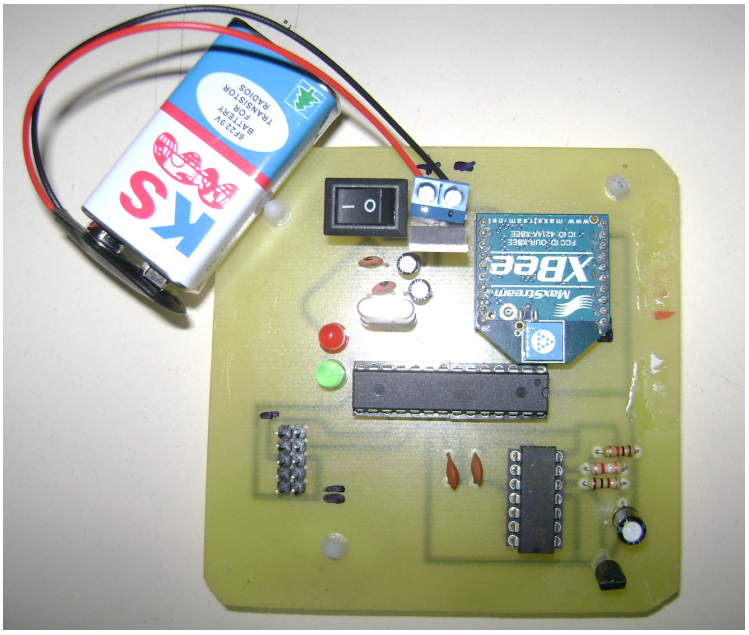

Figura 4.4: Módulo Sensor - PCI

4.2 Dispositivo Atuador

4.2.1 Características Principais

Ao outro módulo desenvolvido, nomeou-se Módulo ou Dispositivo Atuador BACnet-Wireless. A adequação da sua aplicação é semelhante a do módulo sensor já citado. Utilização em retroffiting de sistemas de automação predial já implantados em campo e que requereriam um aperfeiçoamento ou adequação da estrutura física. Pelo fato do módulo ser wireless, alterações da estrutura física podem ser evitadas, acarretando grande economia e velocidade de implantação do projeto.

4.2.1.1 Hardware

O módulo atuador diferencia-se do sensor por apresentar um microcontrolador com mais recursos (ATmega32), principalmente sua memória de 32KB, além de apresentar uma entrada analógica, uma entrada digital e uma saída digital. Possibilitando também maiores condições de inclusão de novas entradas e saídas. Foram desenvolvidos dois módulos atuadores, sendo um em placa universal (figura 4.8) e um em PCI (figura 4.9). Seu circuito (figura 4.5) consiste, essencialmente em: Conexão entre AVR e XBee para comunicação serial com RTS; Implementação da estrutura necessária para gravação BSD através do conector header de 10 pinos; Dois LED's indicativos de Tx (Verde)e Rx (Vermelho); 1 LED azul indicativo de saída digital; 1 Jumper indicativo de entrada digital; Conexão da saída filtrada do LM35DZ ao conversor ADC através de um amplificador não-inversor de ganho 4,9.

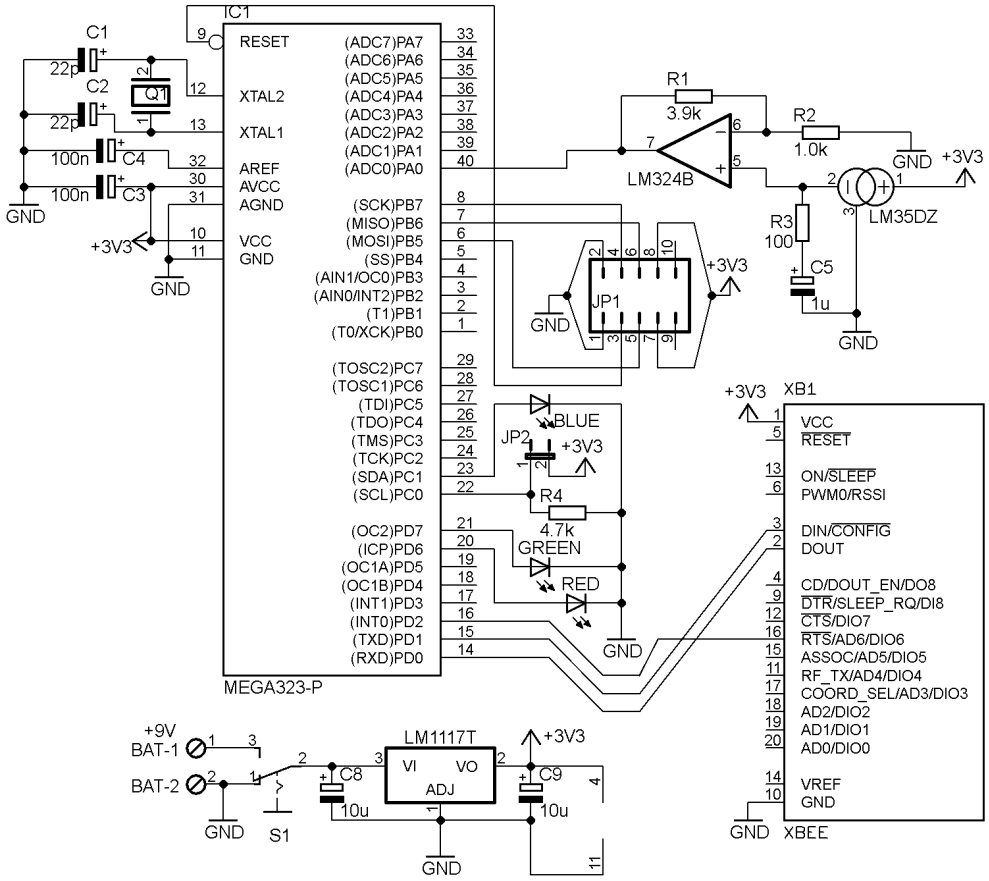

Figura 4.5: Módulo Atuador - Circuito Implementado. Eagle 5.4.0

Os componentes utilizados para sua implementação foram:

- 1 ATmega 32;
- CI's: LM1117, LM35DZ, LM324;
- 1 XBEE;
- 1 Cristal de $4,9152MHz$;
- Capacitores: 2 de $100nF$, 2 de $22pF$, 2 de $10\mu F$ e 1 de $1\mu F$;
- Resistores: 1 de $3,9k\Omega$, 1 de $4,7k\Omega$, 1 de $1,0k\Omega$ e 1 de 100Ω;
- Componentes: 1 Bateria 9V, 1 conector de baterias, 3 LED's, 1 socket, 1 botão, 1 chave on/off e 1 conector header de 10 pinos.

O layout da PCI é apresentado na figura 4.6 (vista superior) e figura 4.7 (vista inferior).

Apresenta-se as fotos dos módulos implementados. Primeiramente em placa universal (figura 4.8) e, em PCI (figura 4.9).

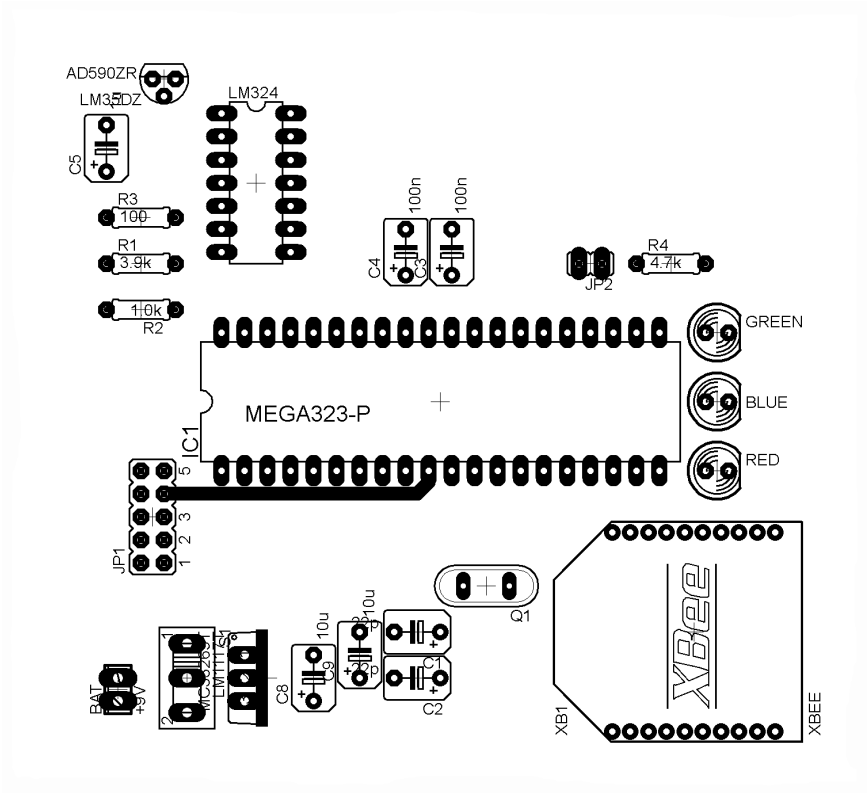

Figura 4.6: Módulo Atuador - Layout Vista Superior. Eagle 5.4.0

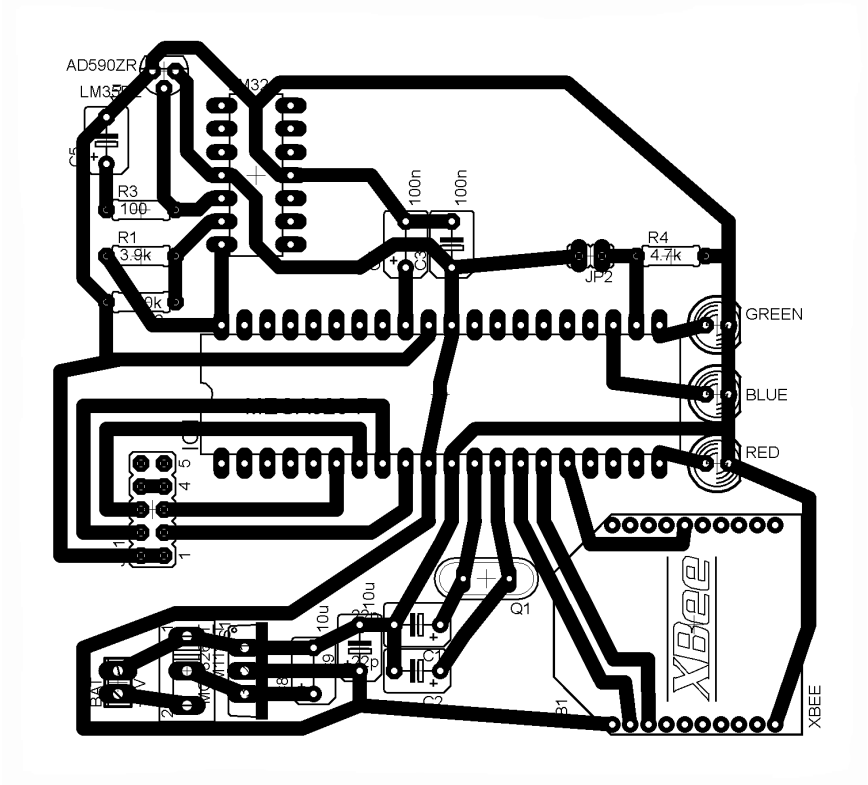

Figura 4.7: Módulo Atuador - Layout Vista Inferior. Eagle 5.4.0

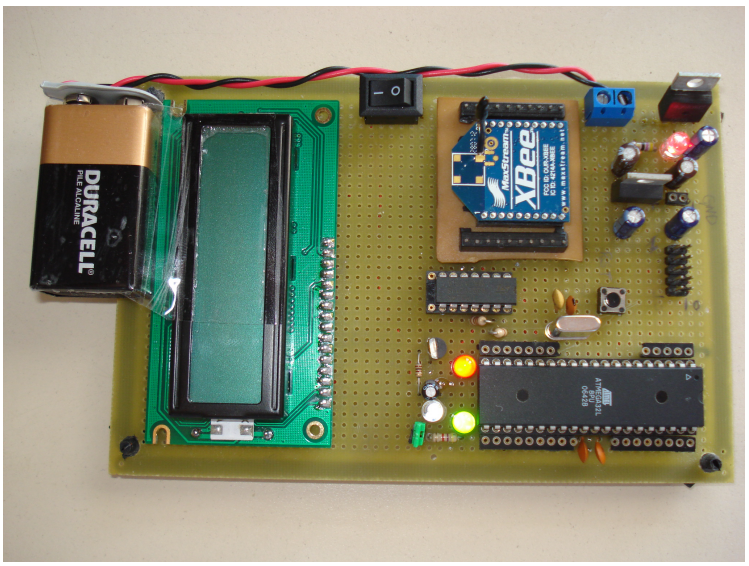

Figura 4.8: Módulo Atuador - Placa Universal

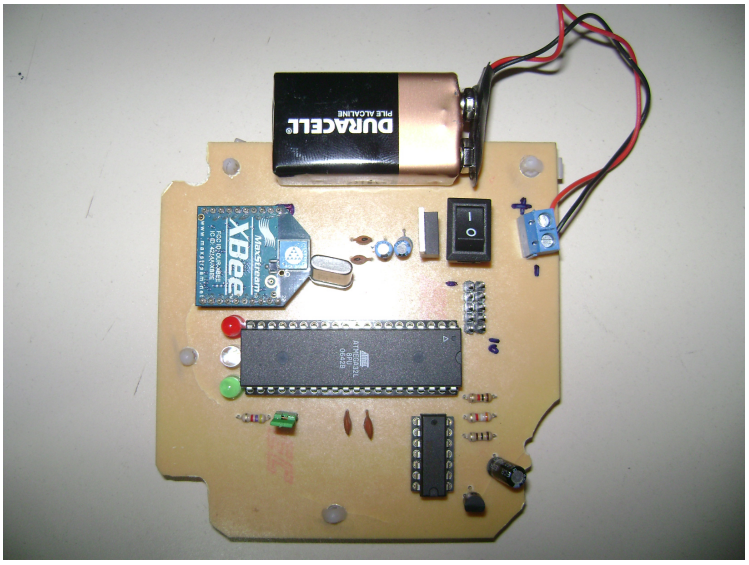

Figura 4.9: Módulo Atuador - PCI

4.2.2 Firmware

Foram desenvolvidas duas versões de firmwares para serem embarcados nos microcontroladores. Uma versão com menos recursos, voltada a dispositivos com menos memória, como no caso do Módulo Sensor. Uma outra versão com mais recursos e maior capacidade de reconfiguração e expansão, voltada ao Módulo Atuador. Os firmwares dos módulos se diferenciam também pela variação nas entradas e saídas dos módulos. No desenvolvimento do software, utilizou-se da pilha BACnet disponível no SourceForge.net, em *http://sourceforge.net/projects/bacnet/*, desenvolvida por Stephen T. Karg (*http://steve.kargs.net/about/*), adequando a pilha às necessidades surgidas durante a realização dos trabalhos.

Nas seções a seguir são apresentados os diagramas sequencial e de atividades, além das máquinas de estados do MS/TP estado Mestre e Receive. Os códigos-fonte e suas respectivas documentações

se encontram no CD fornecido.

4.2.2.1 Diagramas Sequencial e de Atividades

O diagrama da figura 4.10 mostra, genericamente, as chamadas das funções de modo a completar todo o ciclo de atividades. O programa começa chamando funções de inicialização que configuram o controlador, a função *init()* configura seus registradores de I/O's e os LED's de transmissão, além de chamar a função que configura os registradores de timer. Então a função para setar o valor do baud rate é chamada, sendo que esta chama a função de inicialização da RS485, configurando seus registradores para comunicação. A função *dlmstp_set_max_master()* configura o máximo endereço que um nó master pode ter, em seguida seta-se o número máximo de frames de informação que um nó pode enviar. No caso do módulo apresentar AI's, são feitas as chamadas das funções de leitura, conversão e atribuição de valor à variável temperatura lida. Em *input_switch_read()* é feito o tratamento do endereçamento do device, seguido pelo setting das variáveis de endereço usadas na camada de enlace. Em *start_convertion()* são feitas as configurações dos registradores de conversão A/D, iniciando o ADC para leitura da AI. A função *task_milliseconds()* cria um looping onde são chamados os timers dos LED's de transmissão. Então, é feita a chamada da função da camada de enlace de dados e o valor por ela retornado é atribuído à *pdu_len* e, havendo pacote é chamada a função manipuladora de pacotes de rede.

O diagrama de atividades da figura 4.11 apresenta os procedimentos adotados pela camada de enlace, que são: Atribuição do buffer de entrada à variável respectiva; Verifica o recebimento de qualquer tipo de frame, seja válido ou não; Em caso de não recebimento, chama a máquina de estados relativos ao MSTP modo receive e, enquanto não receber algum frame e não entrar em estado idle, permanece neste looping. Recebendo algum frame e passando ao estado idle, poderá ser chamada a máquina de estados relativa ao MSTP em modo mestre. Quando não houver mais necessidade de transição rápida de estado, é deixado o looping da máquina de estados e verifica-se se existem mais pacotes pendentes para serem recebidos. Finalmente, é chamada a função de tratamento do pacote de rede.

A atividade da função manipuladora de pacotes de rede, figura 4.12, consiste no tratamento campo a campo, do pacote recebido. Este é decodificado e seus respectivos valores atribuídos às variáveis correspondentes, gravando por fim o offset do pacote de aplicação. É chamada então a função manipuladora de pacotes de aplicação, que faz a verificação se é um pedido de serviço que requer confirmação ou não. Caso requeira confirmação, um exemplo de serviço deste tipo é o read property, que é tratado pelo seu manipulador. Caso não requeira, um exemplo seria o serviço Who Is, que será tratado também pelo seu respectivo manipulador. Finalizando, em qualquer um desses casos, a ação é retornada à função *task_milliseconds()* (apresentada no diagrama da figura 4.10) recomeçando toda a rotina e assim o sistema permanece até que sofra alguma interferência externa.

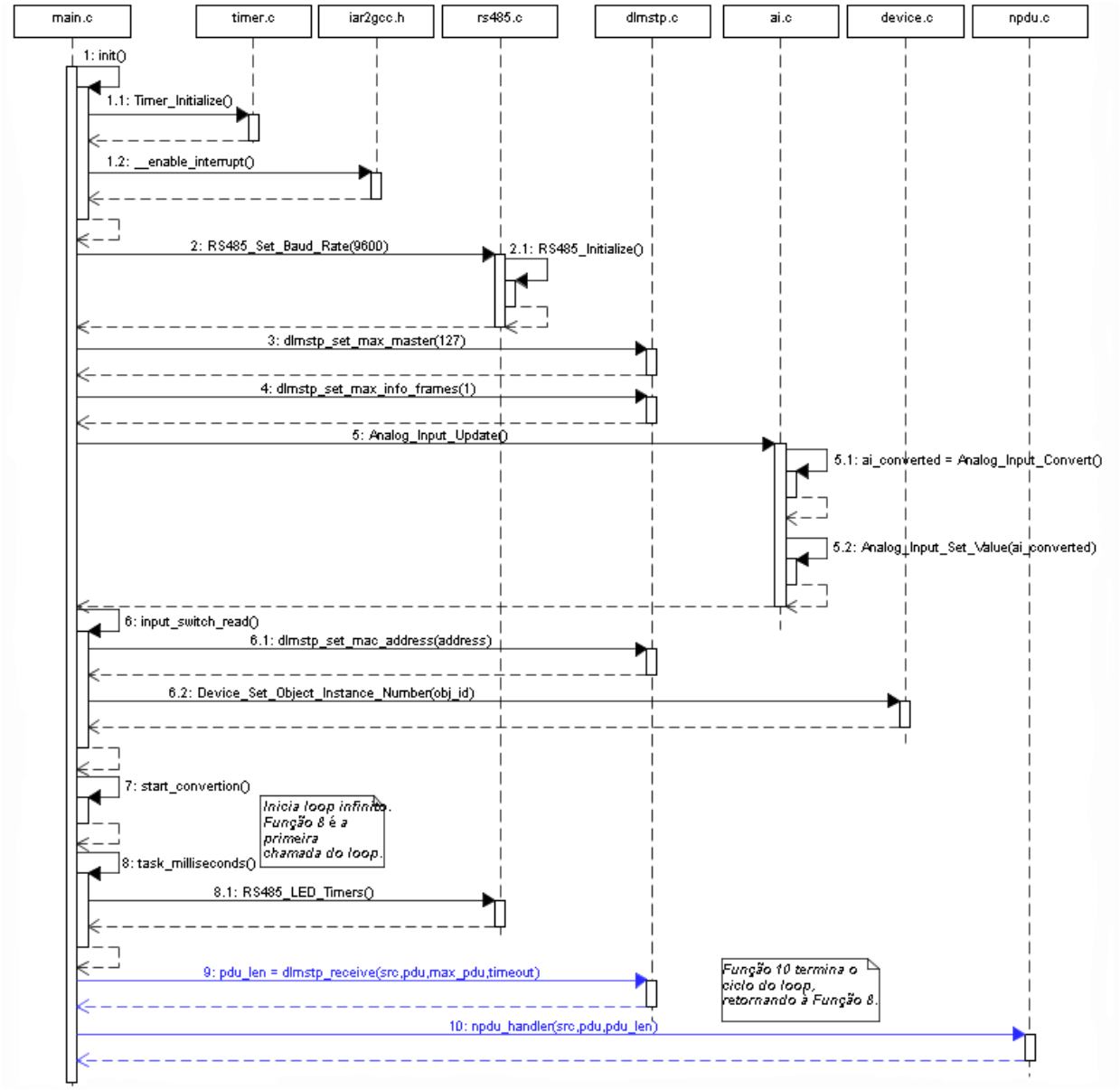

Figura 4.10: Diagrama Sequencial. Editor UML Jude.

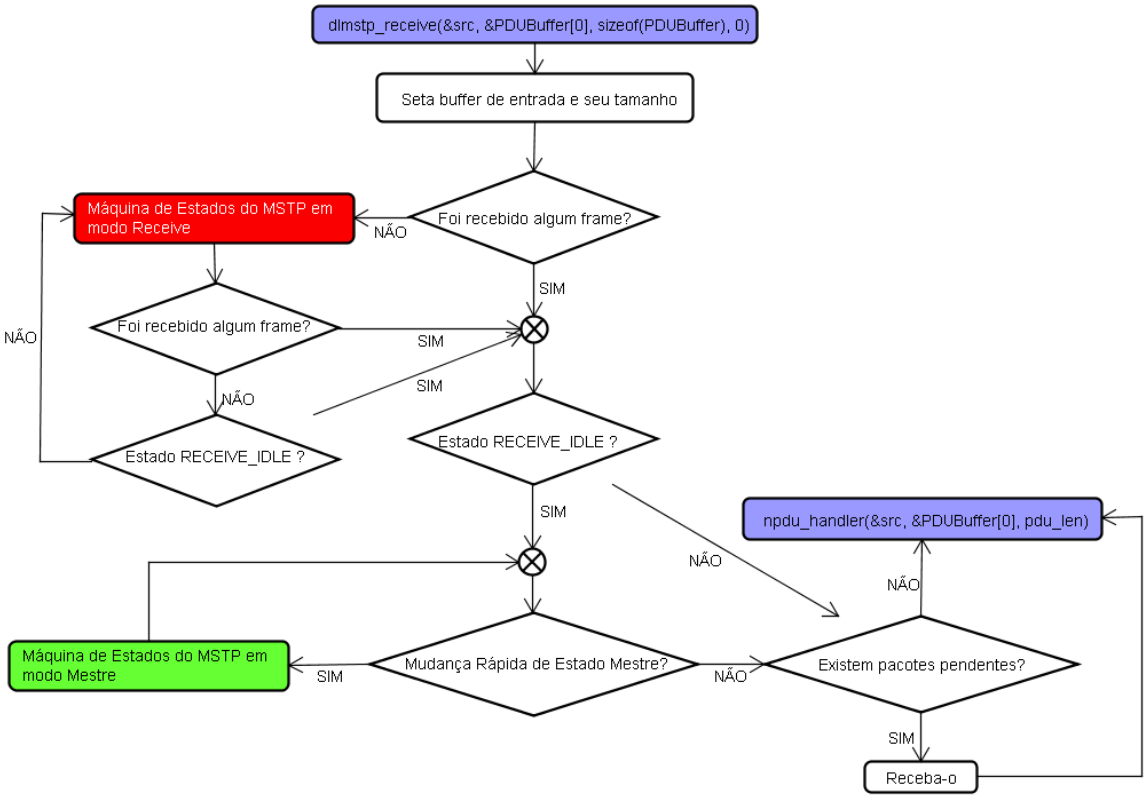

Figura 4.11: Diagrama de Atividades - Camada de Enlace

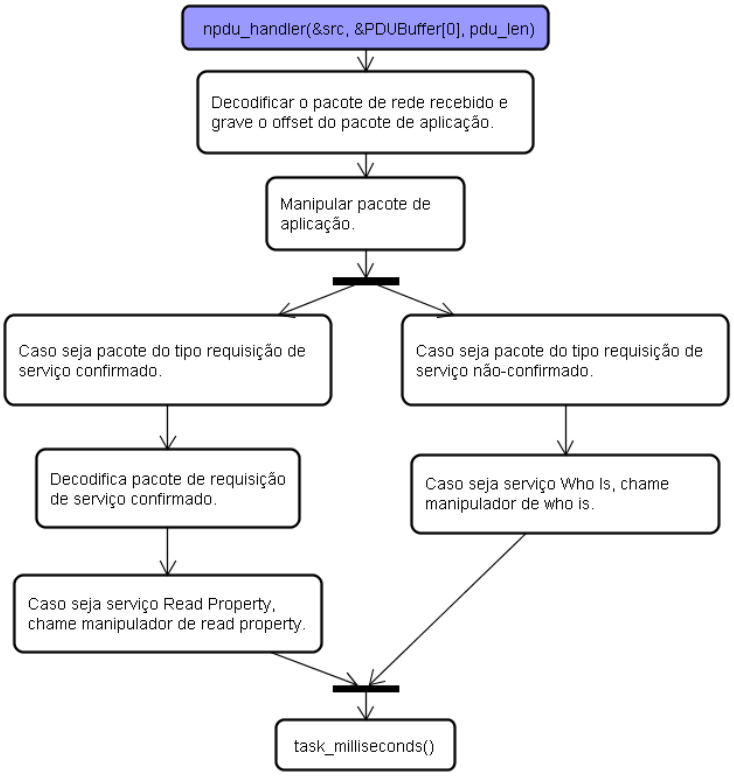

Figura 4.12: Diagrama de Atividades - Camada de Rede

4.2.2.2 Data Link Layer - MS/TP

A presente seção mostra as máquinas de estados finitos relativas ao MSTP em modo receive e em modo master. Tomando como estado inicial IDLE, será analisado o frame MSTP recebido e haverá 3 possibilidades: Recebimento de um octeto que não corresponde ao preâmbulo1 (0x55), sendo este octeto ignorado e permanece-se em IDLE; Recebimento de um erro, permanecendo em IDLE; Recebimento do primeiro octeto correspondente ao preambulo1, passando ao estado de analise do preambulo (IDLE –> PREAMBLE).

Estando agora em PREAMBLE, tem-se as possibilidades: Erro de Timeout, retornando ao IDLE; Erro de recebimento, retornando a IDLE; Recebimento de um octeto relativo ao preambulo1, sendo considerado octeto repetido, permanecendo em PREAMBLE; Recebimento de um octeto que não é o de preambulo2 (0xFF), retornando a IDLE; Recebimento do octeto de preambulo2, passando ao estado HEADER.

Assumindo então HEADER, teremos as possibilidades: Recebimento do octeto identificador do tipo de frame, permanecendo em HEADER mas passando a um subestado de tratamento do tipo do frame que incrementa o índice do header; Recebimento do octeto indicador do endereço de destino ou endereço fonte, permanecendo em HEADER e fazendo index++; Recebimento dos octetos identificadores do tamanho do campo data, permanecendo em HEADER e fazendo index++; Erro de timeout ou erro de recebimento, passando ao estado IDLE.

Em HEADERCRC, é validado o valor do CRC fixado no header da mensagem. Neste estado, tem-se as possibilidades: Caso o valor do HeaderCRC não seja 0x55, é setada a flag de recebimento de frame inválido e retorna a IDLE; Caso o valor do HeaderCRC seja 0x55 mas o valor do endereço não seja igual ao da estação que recebe ou nem mesmo o endereço de broadcast (255), retorna-se a IDLE; Caso o valor do HeaderCRC seja 0x55 e o valor do endereço esteja valido e o comprimento do campo data é maior que o comprimento do buffer de entrada, então é setada a flag de frame inválido por ser um frame muito longo, retornando a IDLE; Caso o valor do HeaderCRC seja 0x55 e o endereço valido, mas o comprimento do campo data é zero, é setada a flag de frame valido mas está sem campo de dados, retoma-se IDLE; Caso o valor do HeaderCRC seja 0x55 e o endereço seja valido e existe campo de dados coerente, é setada a variável DataCRC para 0xFFFF e passa de HEADERCRC para DATA.

Sendo DATA o presente estado, tem-se: Erro de timeout por passar do valor do Tframe_abort, retornando a IDLE; Erro de recebimento, retornando a IDLE; Recebimento dos octetos de dados, armazenando-os no buffer de entrada, fazendo index++ e permanecendo no estado DATA; Quando index atingir o comprimento do campo de dados, então armazena o valor de DataRegister em DataCRC e incrementa index++, permanecendo em DATA; Quando index atingir um valor igual ao comprimento do campo de dados+1, então armazenar o valor de DataRegister em DataCRC e passar ao estado DATACRC.

No estado DATACRC temos duas possibilidades para validação do CRC da mensagem de dados: CRC ruim se o valor do DataCRC não for 0xF0B8, passando a IDLE; CRC bom caso o valor do DataCRC seja 0xF0B8, retornando a IDLE e completando o ciclo da máquina por todos os estados.

Agora trataremos dos estados possíveis do MSTP em modo mestre, conforme apresentados na

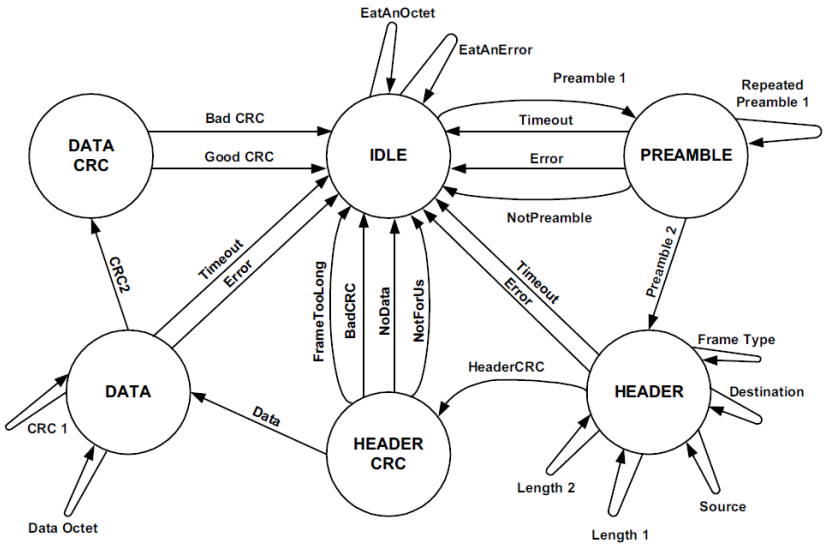

Figura 4.13: Diagrama de Estados em modo Receive, fonte [1]

figura 4.14.

Inicialmente tem-e o estado IDLE: Caso o timer seja maior ou igual ao Tno_token, então o token foi perdido e tem-se a transição IDLE –> NO_TOKEN; Caso seja recebido um frame invalido, aguarda o recebimento de um valido, IDLE –> IDLE; Caso seja recebido um frame valido, mas que não tenha endereço ou o tipo do frame não seja valido ou tipo de propriedade validos, IDLE –> IDLE; Caso seja recebido um frame valido do tipo token e com o endereço do presente nó, então IDLE –> USE_TOKEN; Caso seja recebido um frame valido do tipo Poll for Master, então enviar o frame de Reply to Poll for Master e IDLE –> IDLE; Caso seja recebido um frame valido do tipo que não espera resposta, então indicar uma recepção correta e IDLE –> IDLE; Caso recebido um frame valido e do tipo que espera respostas, então indicar recepção correta e IDLE –> ANSWER_DATA_REQUEST.

Estando agora em USE_TOKEN: Caso nao exista frame para transmitir, então USE_TOKEN –> DONE_WITH_TOKEN; Caso exista um frame aguardando transmissão mas que seja de um tipo que não requeira respostas, então enviar o frame parado e fazer a mudança USE_TOKEN –> DONE_WITH_TOKEN; Caso exista um frame esperando a transmissão e seja de um tipo que requeira respostas, então transmite o frame parado e USE_TOKEN –> WAIT_FOR_REPLY;

Estando em WAIT_FOR_REPLY: Caso o timer seja maior ou igual ao tempo de time-out de respostas, então assumir que a requisição tenha falhado e, WAIT_FOR_REPLY –> DONE_WITH_TOKEN; Caso receba um frame invalido, então altere: WAIT_FOR_REPLY –> DONE_WITH_TOKEN; Caso o frame seja inesperado (endereço invalido, tipo do frame invalido), então WAIT_FOR_REPLY –> IDLE;

Estando em DONE_WITH_TOKEN: Se o contador de frames for menor que Nmax_info_frames, então pode enviar mais frames de informação antes de passar o token e DONE_WITH_TOKEN –> USE_TOKEN; Caso exista somente um nó mestre na rede, incremente o contador de token e o de frame e DONE_WITH_TOKEN –> USE_TOKEN; Caso haja mais de um nó mestre na rede

e se o endereço do próximo nó é igual ao endereço do presente nó, então incrementar contador de tokens e transmitir o token para o próximo nó, e DONE_WITH_TOKEN –> PASS_TOKEN;

Estando em PASS_TOKEN: Caso o timer seja menor que o tempo de uso do token e o contador de eventos é maior que o numero minimo de octetos, então assume que um frame foi enviado para o novo usuário do token, e PASS_TOKEN –> IDLE; Caso o timer seja maior ou igual ao tempo de uso, e o contador de tentativas seja menor que Nretry_token, então incrementar o contador de tentativas e transmitir o token para o próximo nó e PASS_TOKEN –> PASS_TOKEN; Caso o timer seja maior ou igual ao tempo de uso, e o contador de tentativas seja maior ou igual ao Nretry_token, então assume que o próximo nó tenha falhado e procura por um novo nó sucessor, PASS_TOKEN –> POLL_FOR_MASTER;

Estando em NO_TOKEN: Caso o timer seja menor que o tempo sem token e o contador de eventos seja maior que o numero minimo de octetos, então algum outro nó existe num endereço menor, NO_TOKEN –> IDLE; Caso o endereço do presente nó seja o menor presente na rede, então envie um frame de transmissão Poll For Master e zerar o contador de tokens, de tentativas e de evento e NO_TOKEN –> POLL_FOR_MASTER de modo que ache um novo sucessor;

Estando em POLL_FOR_MASTER: Caso recebido um frame valido, de endereços validos e do tipo Reply to Poll for Master, então setar o proximo endereço como o endereço da fonte e transmitir o frame para o próximo nó, POLL_FOR_MASTER –> PASS_TOKEN; Caso receba um frame inexperado (endereço ou tipo do frame invalido), então indica a presença de tokens multiplos, então POLL_FOR_MASTER –> IDLE; Caso seja mestre único na rede, então POLL_FOR_MASTER –> USE_TOKEN; Caso não seja único mestre da rede e tenha recebido um frame invalido, então transmita o token para próximo nó e POLL_FOR_MASTER –> PASS_TOKEN; Caso o endereço da próxima estação e da atual sejam iguais e que não seja único mestre da rede, então setar para único mestre para indicar que o presente nó é único mestre, POLL_FOR_MASTER –> USE_TOKEN.

E finalmente, estando em ANSWER_DATA_REQUEST: Caso uma resposta esteja disponível pelas camadas superiores com o delay de respostas depois da recepção no final do octeto de requisição, então transmitir o frame de resposta e ANSWER_DATA_REQUEST –> IDLE; Caso não exista resposta disponível pelas camadas superiores, então uma resposta imediata não é possível e qualquer resposta deve aguardar até que o presente nó receba o token, ANSWER_DATA_REQUEST –> IDLE.

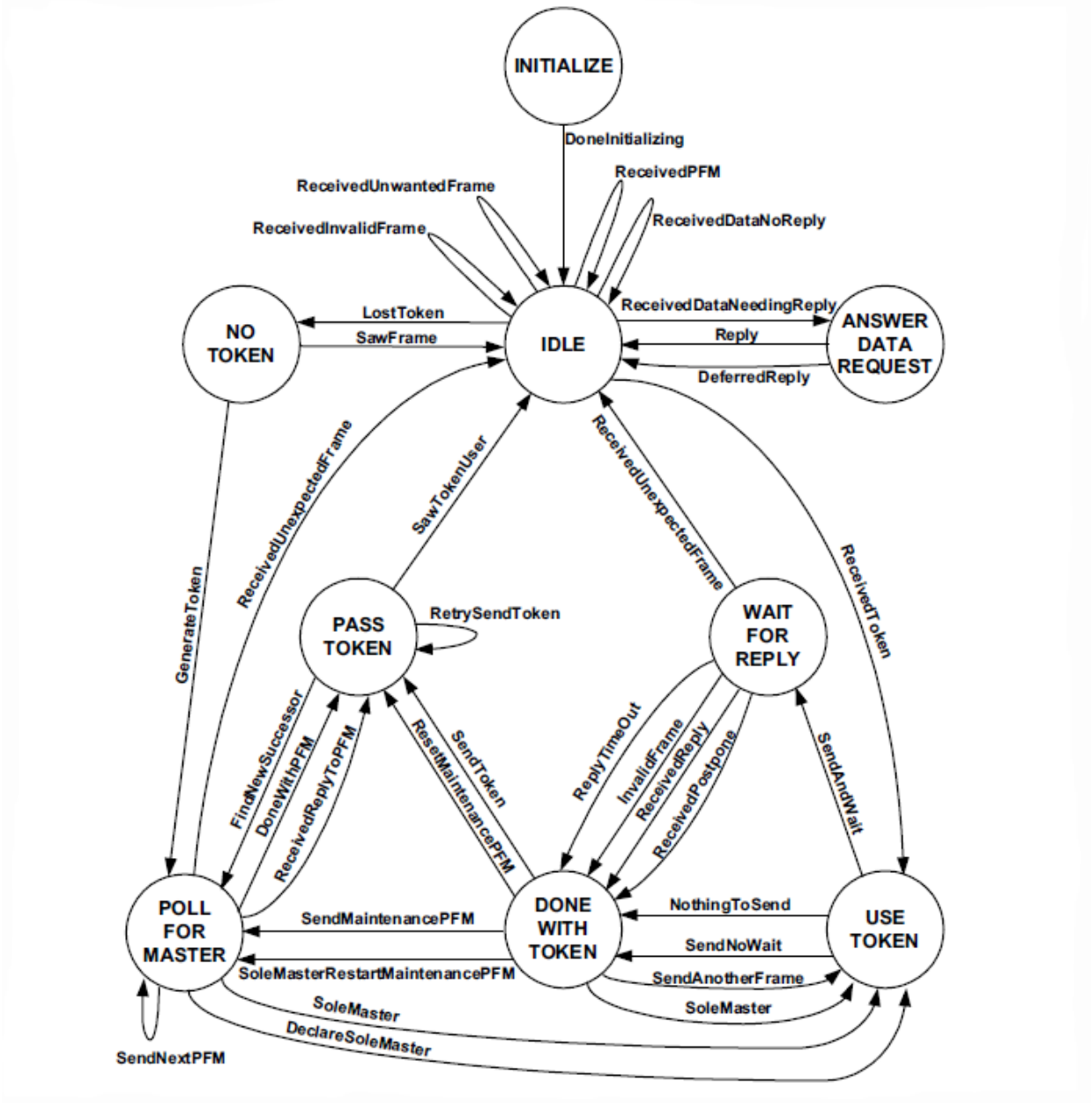

Figura 4.14: Diagrama de Estados em modo Mestre, fonte [1]

Capítulo 5

Resultados Experimentais

Neste capítulo serão apresentados os resultados obtidos através dos dispositivos móveis implementados.

Os devices desenvolvidos, quando ligados, entram em estado de transmissão enviando frames que podem ser visualizados na figura 5.1, recebidos pelo XBee que está conectado na CONUSBBEE ligada ao computador de monitoramento.

Espera-se que os frames recebidos atendam ao formato estabelecido ao MS/TP, na tabela 5.1 é feita uma listagem do esperado e do obtido. Onde 0x?? indica octeto indeterminado ou variável.

Campo	Esperado	Recebido
PREAMBLE	0x55 e 0xFF	0x55 e 0xFF
FRAME TYPE	0x00 a 0xFF	0x01 (Poll)
DESTINATION ADDRESS	0x??	Variável
SOURCE ADDRESS	0x??	0x0D
LENGTH	0x?? e 0x??	0x00 (no data)
HEADER_CRC	0x??	Variável
DATA	0x??	—- (no data)
DATA_CRC	0x?? e 0x??	—- (no data)

Tabela 5.1: Tabela Comparativa: Experado x Recebido

Octetos de Preambulo corretos, pois devem ser exatamente 0x55 e 0xFF. Octeto de tipo de frame correto, pois ao ser ligado o módulo inicia procurando por mais devices na rede, enviando tipo 0x01 que corresponde a frames de Poll for Master. Endereço de destino variável, pois o device está procurando novos dispositivos na rede, que são de endereços desconhecidos. Então procura-se incrementando os endereços até chegar ao valor máximo de 0xFF. Endereço de fonte igual a 0x0D, que vale 13 em decimal, exatamente o valor configurado via software. Octetos do campo de tamanho são 00, pois um frame do tipo 0x01 não apresenta campo de dados. Octeto do HeaderCRC variável pois é calculado de acordo com os demais octetos e o octeto de endereço de destino também está em constante incremento. Octeto DATA ausente por ser do tipo Poll for Master.

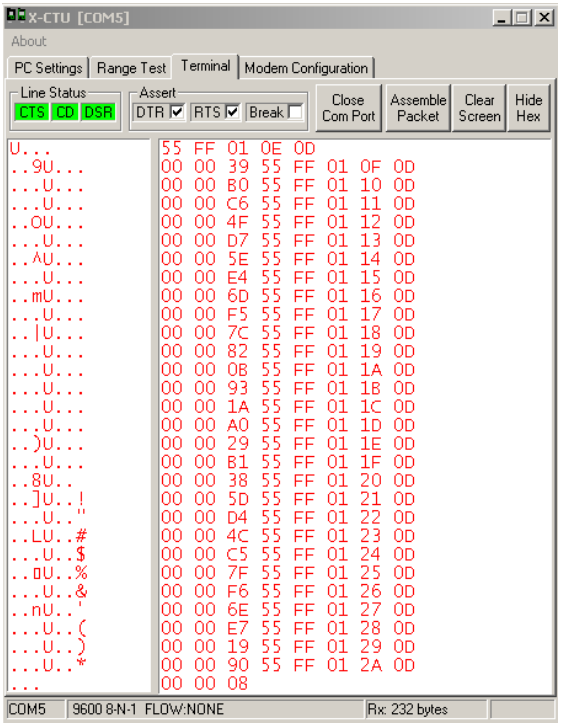

Figura 5.1: XCTU - Recebimento dos Frames MSTP.

5.1 Utilizando o CAS

O software CAS BACnet explorer obteve êxito no monitoramento dos devices, na figura 5.2 observa-se que o device monitorado foi o modulo sensor, pois apresenta apenas uma AI que é a temperatura medida. Nota-se, pelo CAS que: O dispositivo foi reconhecido como BACnet MSTP, pertencente à rede 0, seu BACnet_ID é 86090 - Sensor LARA. Em seguida tem-se a medição do último update realizado, seu endereço MAC e seu vendor_name GRAV-LAVSI. Foi lida também sua AI[0], de nome Temperatura Local e valor presente 22,416422^{o}C. À direita tem-se a tabela de monitoramento das variáveis. Observa-se a precisão mostrada no valor da entrada analógica Temperatura Local. Para ambientes prediais, tal precisão é desnecessária, mas não deixa de ser uma característica a mais que pode ser utilizada em trabalhos que requeiram precisão.

Agora para o módulo atuador temos (figura 5.3): O dispositivo foi reconhecido como BACnet MSTP, pertencente à rede 0, seu BACnet_ID é 86013 - Atuador LAVSI. Em seguida tem-se a medição do último update realizado, seu endereço MAC e seu vendor_name GRAV-LAVSI. Foram lidas também suas: AI[0], de nome Temperatura Ambiente e valor presente 25,126101^{o}C; BI[0], de nome Status Ar Condicionado e valor presente 1; BO[0], de nome LED Azul e valor presente 1; AO[0] de nome Valvula 1 e valor presente 0,00. À direita tem-se a tabela de monitoramento das variáveis. Além disso, a BO e a AO admitem escrita, *write property*.

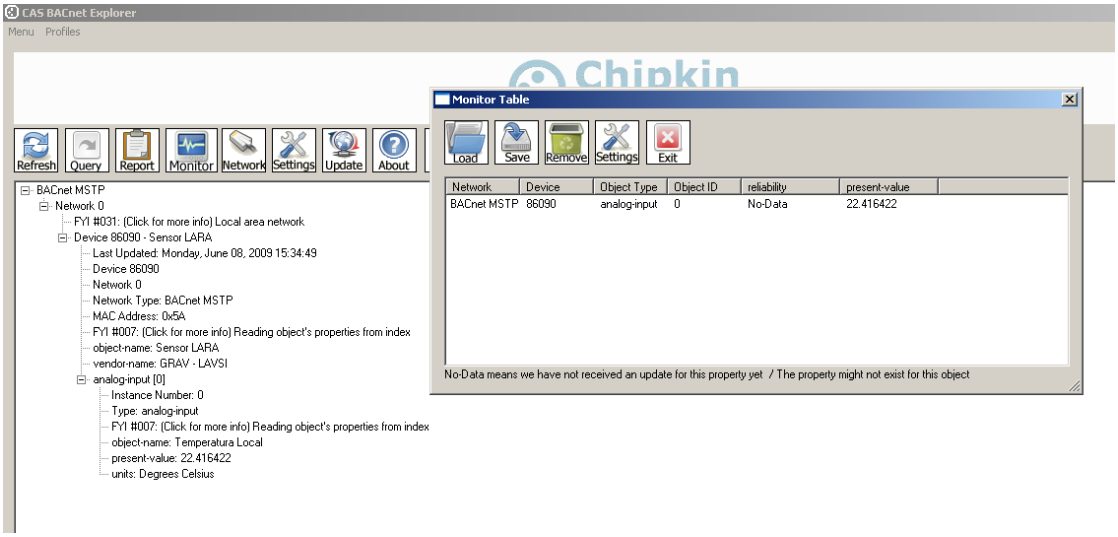

Figura 5.2: CAS BACnet Explorer - Propriedades do Modulo Sensor.

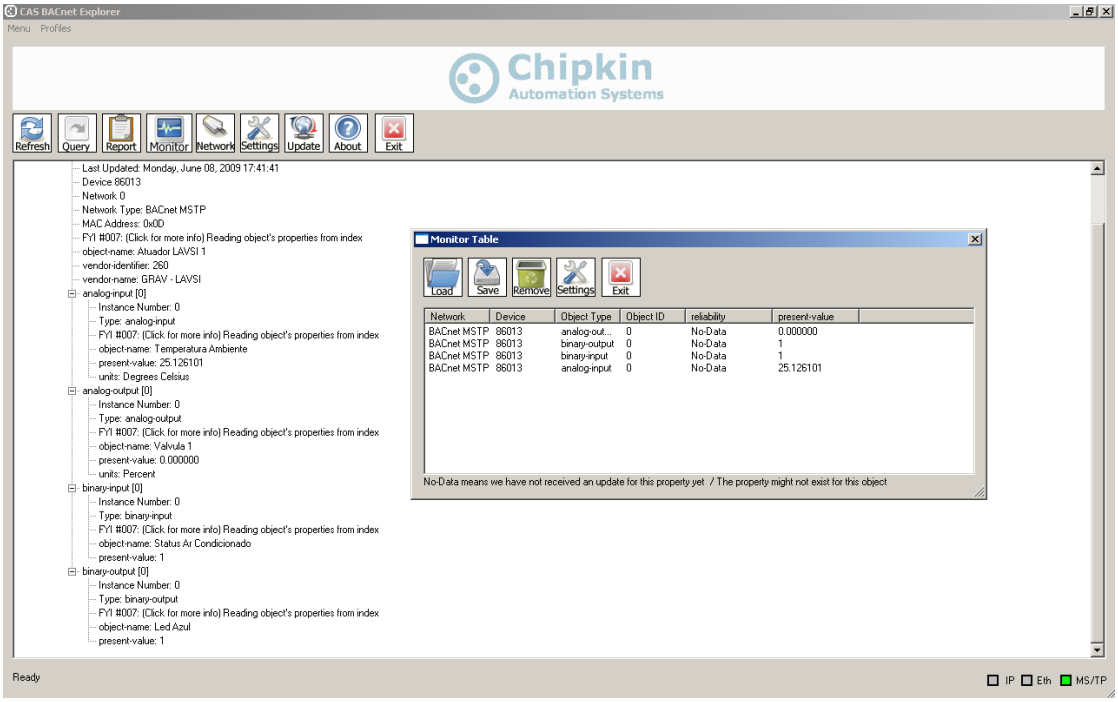

Figura 5.3: CAS BACnet Explorer - Propriedades do Modulo Atuador.

Capítulo 6

Conclusões e Perspectivas

Hoje, cada vez mais, o mundo está mostrando a sua tendência e sua adesão à necessidade da automação de ambientes prediais. Mesmo tendo distintas justificativas, seja para comodidade, conforto, praticidade, todos se uniformizam quando é abordada a questão do impacto ambiental e da economia. Fatores diretamente ligados à automação predial e ao paradigma *Ambient Intelligence*. Defensor da automação inteligente, visando o conforto mas colocando em prática técnicas mais refinadas de controle e interação entre os equipamentos, acarretando numa considerável evolução quanto à economia de energia.

Este trabalho faz parte de um conjunto de trabalhos do LAVSI, *Laboratório de Automação, Visão e Sistemas Inteligentes*, que dentre outros estudos, trabalha também com o paradigma *Ambient Intelligence* visando alcançar objetivos como conforto térmico e economia energética. Surgiu então, a necessidade de dispositivos que permitissem flexibilidade de desenvolvimento, de localização, de baixo consumo energético e que implementassem um protocolo já difundido no mercado da automação predial. Visou-se a implementação de dispositivos BACnet com sua comunicação wireless por meio do protocolo ZigBee. Inicialmente pretendia-se construir uma interface entre o BACnet e a camada de rede do ZigBee. No entanto, a versão dos transceivers XBees utilizados não implementam tal camada, mas somente as MAC e PHY do IEEE 802.15.4. Utilizou-se então os XBees como maneira de alterar a comunicação MS/TP serial wired para MS/TP serial wireless. Os módulos então passaram a ser chamados BACnet-ZigBee ou BACnet-Wireless. A implementação do protocolo partiu da pilha BACnet desenvolvida por Steve Karg, sendo feitas as devidas alterações para adequação ao que foi proposto e aos recursos do trabalho. Durante a execução, foram encontrados vários problemas como: erros de timeout e falhas na comunicação wireless (ver anexo I); necessidade de microcontroladores com mais memória e portas; necessidade do monitoramento da comunicação.

Foram construídos dois tipos de dispositivos que chamou-se módulo sensor e módulo atuador. O primeiro voltado ao AVR168, com o intuito de sensoriar ambientes, o segundo ao AVR32, também sensoriando além de dispor de mais opções de configuração. Ao todo foram implementados 4 módulos, sendo 2 sensores e 2 atuadores, em versões protótipos na placa universal e na placa impressa. Para a supervisão dos módulos, utilizou-se o software CAS BACnet Explorer. Além de ter sido feita também uma tela gráfica utilizando o software ActionView, mas não obtivemos

sucesso na configuração para comunicação com os módulos.

O trabalho pode ser melhorado através da implementação de estratégias de economia energética pela utilização dos modos *sleep* dos microcontroladores e do XBee. Criação de uma rede de coordenadores de *sleep*(ver [8]) e o endereçamento dos módulos via hardware. Sugere-se também a implementação em dispositivos mais modernos, como o ZigBit ou o ATmega128.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] AMERICAN SOCIETY OF HEATING, REFRIGERATION, AND AIR-CONDITIONING ENGINEERS INC. *BACnet: A Data Communication Protocol for Building Automation and Control Networks - ANSI/ASHRAE Standard 135-2004*. Atlanta.

[2] COUTO, F. L.; FIGUEREDO, L. F. C. *Medição Móvel de Conforto Térmico para Rede de Automação Predial Wireless*. Universidade de Brasília: Trabalho de Graduação Eng. Mecatrônica, Faculdade de Tecnologia, 2008.

[3] DVORAK, J. IEEE 802.15.4 and ZigBee Overview. *Motorola*, p. 26, 2005.

[4] DIGI INTERNATIONAL. *XBee/XBee-PRO OEM RF Modules*. 2008.

[5] TANENBAUM, A. S. *Redes de Computadores*. 4ed.: Editora Campus, 2003.

[6] KOPETZ, H. *Real-Time Systems: Design Principles for Distributed Embedded Applications*. Boston: Kluwer Academic Publishers, 1997.

[7] PARK, T. J.; CHON, Y. J.; PARK, D. K.; HONG, S. H. BACnet over ZigBee: A new approach to wireless datalink channel for BACnet. 2007.

[8] ZIGBEE ALLIANCE. *ZigBee Document 053474r17 - ZigBee Specification*. January 2008.

[9] PINHEIRO, J. M. As Redes com ZigBee. *Eletrônica.org*, 2004.

[10] LIU, Q.; REN, P. Design and Implementation of MS/TP in Embedded System. *Second IEEE Conference on Industrial Electronics and Applications*, 2007.

[11] FIGUEREDO, L. F. C. Tutorial XBee. *Laboratório de Automação, Visão e Sistemas Inteligentes, Departamento de Engenharia Elétrica*, Universidade de Brasília.

[12] ATMEL CORPORATION. *Datasheet - ATmega168*. 2007.

[13] ATMEL CORPORATION. *Datasheet - ATmega32*. 2007.

[14] BORGES, G. A. et al. Desenvolvimento com Microcontroladores Atmel AVR. *Laboratório de Controle e Visão por Computador - LCVC, Universidade de Brasília*, 2008.

ANEXOS

I. CONFIGURAÇÕES DE CLOCK

Um dos problemas encontrados foi a configuração do valor do clock dos AVRs. Inicialmente pretendia-se utilizar o clock interno (1MHz) devido tal frequência já ser suficiente para a aplicação que se pretende. No entanto, deparou-se com inúmeros erros de falha de comunicação chegando até a ser interrompida. A seguir apresenta-se algumas tabelas encontradas em *http://www.wormfood.net/avrbaudcalc.php*

32.768 Khz		
Baud	UBRR	% of error
300	6	2.5
600	2	12.1
1200	1	17.2
2400	0	17.2
4800	0	134.4
9600	0	368.8
14400	0	603.1
19200	0	837.5
28800	0	1306.3
38400	0	1775.0
57600	0	2712.5
76800	0	3650.0
115200	0	5525.0
230400	0	11150.0

1 Mhz		
Baud	UBRR	% of error
300	207	0.2
600	103	0.2
1200	51	0.2
2400	25	0.2
4800	12	0.2
9600	6	7.5
14400	3	7.8
19200	2	7.8
28800	1	7.8
38400	1	22.9
57600	0	7.8
76800	0	22.9
115200	0	84.3
230400	0	268.6

1.2288 Mhz		
Baud	UBRR	% of error
300	255	0.0
600	127	0.0
1200	63	0.0
2400	31	0.0
4800	15	0.0
9600	7	0.0
14400	4	6.3
19200	3	0.0
28800	2	12.5
38400	1	0.0
57600	0	25.0
76800	0	0.0
115200	0	50.0
230400	0	200.0

1.8432 Mhz		
Baud	UBRR	% of error
300	383	0.0
600	191	0.0
1200	95	0.0
2400	47	0.0
4800	23	0.0
9600	11	0.0
14400	7	0.0
19200	5	0.0
28800	3	0.0
38400	2	0.0
57600	1	0.0
76800	1	33.3
115200	0	0.0
230400	0	100.0

Figura I.1: Tabelas Clock x Bauds x Taxa de Erros X UBRR

4.608 Mhz		
Baud	UBRR	% of error
300	959	0.0
600	479	0.0
1200	239	0.0
2400	119	0.0
4800	59	0.0
9600	29	0.0
14400	19	0.0
19200	14	0.0
28800	9	0.0
38400	7	6.7
57600	4	0.0
76800	3	6.7
115200	2	20.0
230400	0	20.0

4.9152 Mhz		
Baud	UBRR	% of error
300	1023	0.0
600	511	0.0
1200	255	0.0
2400	127	0.0
4800	63	0.0
9600	31	0.0
14400	20	1.6
19200	15	0.0
28800	10	3.1
38400	7	0.0
57600	4	6.3
76800	3	0.0
115200	2	12.5
230400	0	25.0

5 Mhz		
Baud	UBRR	% of error
300	1041	0.0
600	520	0.0
1200	259	0.2
2400	129	0.2
4800	64	0.2
9600	32	1.4
14400	21	1.4
19200	15	1.7
28800	10	1.4
38400	7	1.7
57600	4	7.8
76800	3	1.7
115200	2	10.6
230400	0	26.3

5.0688 Mhz		
Baud	UBRR	% of error
300	1055	0.0
600	527	0.0
1200	263	0.0
2400	131	0.0
4800	65	0.0
9600	32	0.0
14400	21	0.0
19200	16	3.0
28800	10	0.0
38400	7	3.0
57600	5	9.1
76800	3	3.0
115200	2	9.1
230400	0	27.3

Figura I.2: Tabela Clock x Bauds x Taxa de Erros x UBRR

Estava-se utilizando inicialmente o clock de 1MHz e Baud de 9600. Nota-se pela tabela do respectivo clock que nessas configurações o registrador de BaudRate (UBRR) deve ser 6 e ocorre um erro de 7,5%. Este erro era exatamente a causa das perdas de comunicação e passamos a utilizar o clock de 4,9152MHz com o mesmo Baud de 9600 o erro fica anulado.

II. CONFIGURAÇÃO - NETBEANS

Utilizou-se como ambiente de desenvolvimento o IDE NetBeans 6.5.1. Encontrado gratuitamente em *www.netbeans.org*. Apresenta-se algumas observações quanto à configuração do programa.

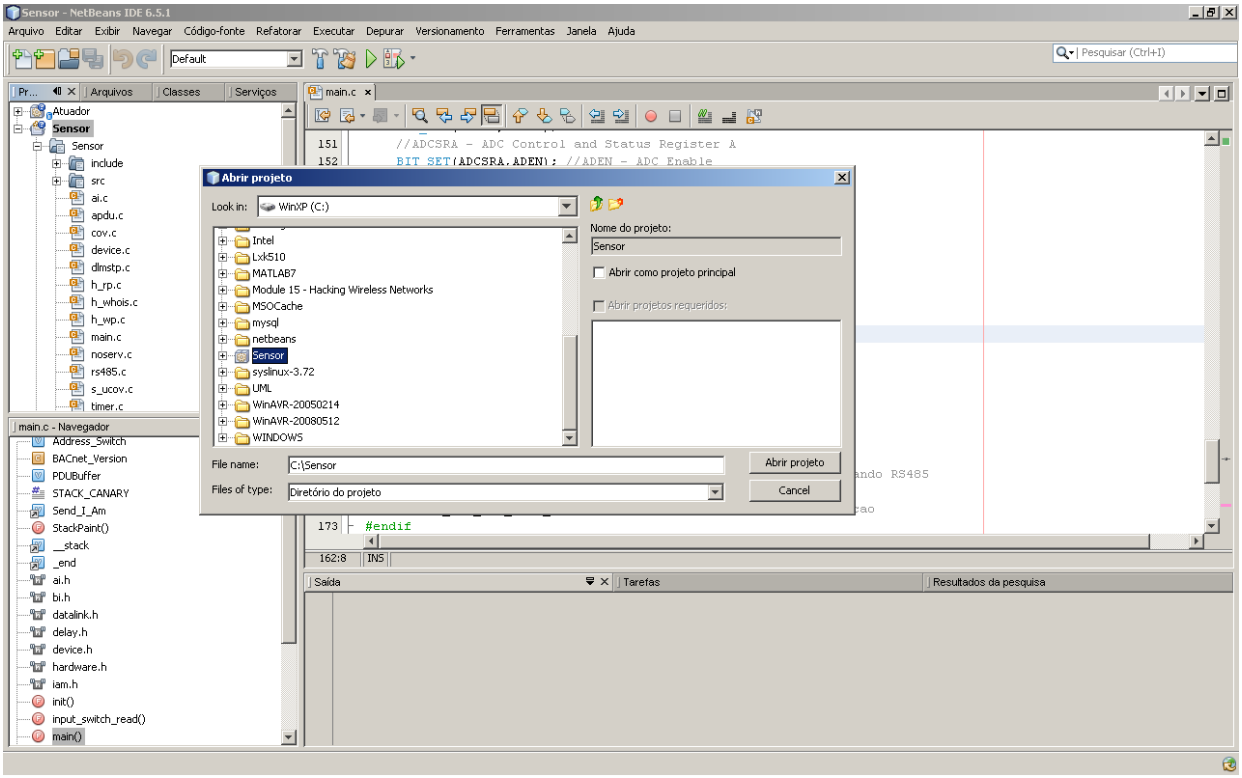

Figura II.1: IDE NetBeans - Abrindo Projeto

Na figura II.1 é mostrada como deve ser feita a importação de um projeto no software. Clicando em **Arquivo** e em **Abrir Projeto** será mostrada a tela acima, onde pode-se selecionar o respectivo projeto.

Ilustra-se também como foi feita a configuração de compilação do IDE após instalado o complilador AVR-GCC, encontrado juntamente ao Win-AVR.

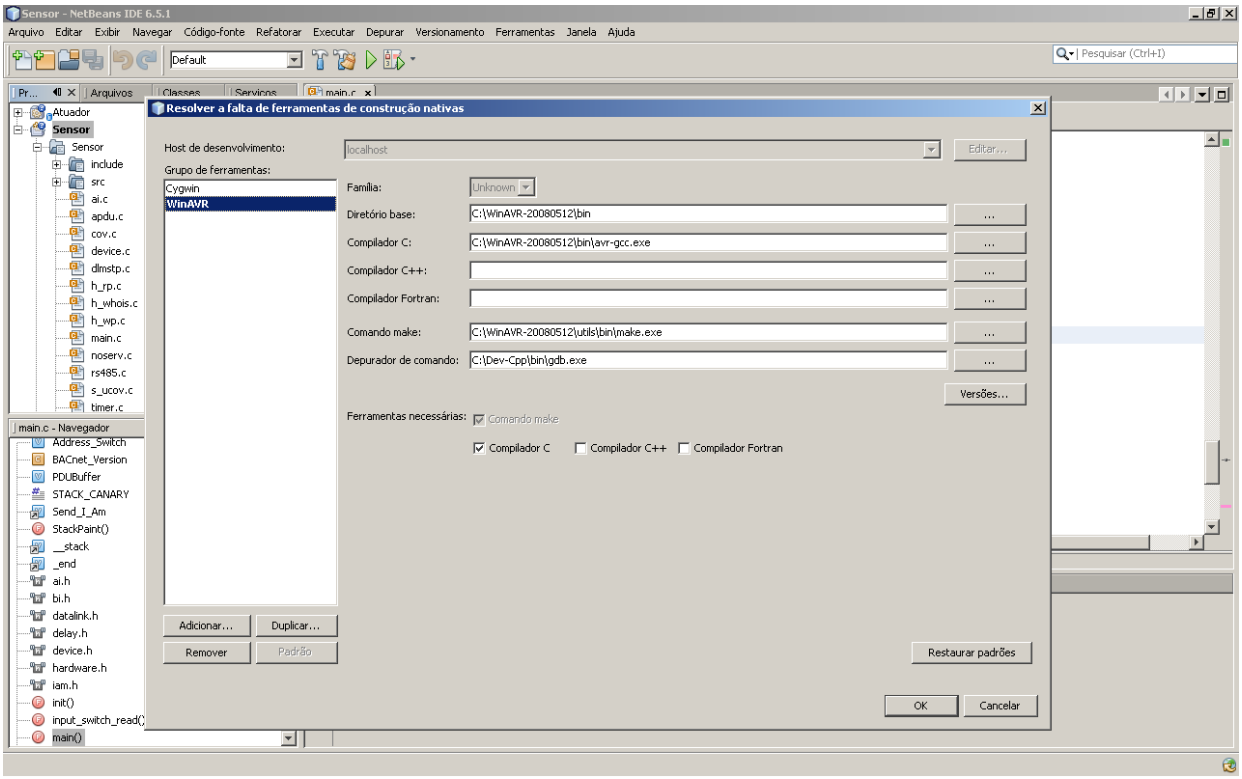

Figura II.2: IDE NetBeans - Configuração do Projeto

Clicando com o botão direito do mouse sobre o projeto, selecione **Limpar e Construir** e será mostrada a tela da figura II.2. Deve-se criar um novo grupo de ferramentas clicando em **Adicionar**. No exemplo foi criado o grupo de ferramentas WinAVR. No campo **Diretório Base** deve ser inserido o caminho para o diretório raiz onde é encontrado o executável do compilador. No campo **Compilador C** deve-se colocar o caminho para o executável do compilador. O mesmo deve ser feito quanto ao comando make. Selecione o campo **Compilador C** e clique em OK e, se tudo estiver corretamente configurado, o projeto será devidamente construído.

III. DOXYGEN

A documentação do firmware disponibilizada no CD, foi gerada através do software doxygen, encontrado em *http://www.stack.nl/ dimitri/doxygen/*. Alguns dos comandos utilizados nos códigos-fonte:

- \param - parâmetro de uma função
- \return - valor de retorno da função
- \struct - variável do tipo struct
- \enum - enumerações
- \fn - documentação sobre a função e sua descrição
- \var - variáveis do programa
- \file - descrição sobre o arquivo em si
- \def - diretiva define

Exemplos:

```
/**
* \file nome_do_arquivo.c
* Descrição resumida sobre o arquivo e suas funcionalidades.
*/

/**
* \fn tipo_do_valor_de_retorno nome_da_funcao(tipo_do_argumento nome_do_argumento)
* Descrição sobre a função.
* \param nome_do_argumento
* Descrição sobre o argumento.
* \return valor_de_retorno
* Descrição sobre o valor de retorno.
/*
tipo_do_valor_de_retorno nome_da_funcao(tipo_do_argumento nome_do_argumento)
{
Escopo da funcao.
}
```

IV. SOFTWARE - HARDWARE.H

```
/**\file hardware.h
Arquivo com algumas configurações de hardware.*/

#ifndef HARDWARE_H
#define HARDWARE_H

#define F_CPU 4915200UL

#include "iar2gcc.h"
#include "avr035.h"

#define LED_GREEN_INIT() BIT_SET(DDRD, DDD7)
#define LED_GREEN_ON() BIT_CLEAR(PORTD, PD7)
#define LED_GREEN_OFF() BIT_SET(PORTD, PD7)

#endif
```

V. DESCRIÇÃO DO CONTEÚDO DO CD

No CD que acompanha este trabalho, encontra-se:

- **Data Sheets:** Pasta contendo os data sheets utilizados no projeto.

  Diretório: \DataSheets

- **Esquemáticos e Diagramas:** Pastas contendo os esquemáticos e diagramas do projeto.

  Diretório: \Diagramas Eagle5.4.0

  Diretório: \Diagramas Jude

- **Firmware:** Pasta dos projetos do NetBeans com os códigos fonte que foram gravados nos AVRs.

  Diretório: \Firmware

  Diretório: \Sensor

  Diretório: \Atuador

- **Docs:** Pasta da documentação dos firmwares.

  Diretório: \Sensor

  Diretório: \Atuador

- **Relatório:** Arquivo eletrônico em PDF com este relatório.