Thiago Felipe da Cunha

Controle centralizado de equipamentos de ar condicionado via rede sem fio ZigBee

São José – SC

Março / 2013

Thiago Felipe da Cunha

Controle centralizado de equipamentos de ar condicionado via rede sem fio ZigBee

Monografia apresentada à Coordenação do Curso Superior de Tecnologia em Sistemas de Telecomunicações do Instituto Federal de Santa Catarina para a obtenção do diploma de Tecnólogo em Sistemas de Telecomunicações.

Orientador:
Prof. Tiago Semprebom, Dr. Eng.

CURSO SUPERIOR DE TECNOLOGIA EM SISTEMAS DE TELECOMUNICAÇÕES
INSTITUTO FEDERAL DE SANTA CATARINA

São José – SC

Março / 2013

Monografia sob o título *"Controle centralizado de equipamentos de ar condicionado via rede sem fio ZigBee"*, defendida por Thiago Felipe da Cunha e aprovada em 22 de março de 2013, em São José, Santa Catarina, pela banca examinadora assim constituída:

Prof. Tiago Semprebom, Dr. Eng.
Orientador

Prof. Marcelo Maia Sobral, Dr. Eng.
IFSC

Prof. Márcio Henrique Doniak, M. Eng.
IFSC

“Se a educação sozinha não pode tranformar a sociedade,
tampouco sem ela a sociedade muda.”
Paulo Freire

Agradecimentos

Dedico esta conquista, como todas as demais, em primeiro lugar a minha mãe Joamara, a guerreira responsável por tudo o que sou e consegui conquistar até hoje.

Agradeço também a dona dos meus olhos Ana Paula, pela paciência, incentivo, compreensão e principalmente pelo carinho, essa vitória é nossa.

Agradeço também a todos os meus amigos e professores que me acompanharam durante estes três anos e meio, espero fazer jus aos conhecimentos que me foram passados por ambos. Em especial agradeço ao professor Tiago Semprebom um dos responsáveis pela realização deste trabalho, que aceitou me orientar com a bola já rolando. Obrigado pelos conselhos e pela ajuda.

Agradeço por último ao mundo, pela sua diversidade e constante mutação, com sua infinidade de coisas a serem descobertas e reinventadas.

Resumo

Em função do problema com o desperdício de energia devido a aparelhos de ar condicionado ligados desnecessariamente no campus do IFSC de São José, surgiu a ideia de desenvolver um sistema para o controle centralizado dos aparelhos de ar condicionado.

Após feito um estudo sobre o funcionamento básico do sistema eletrônico de controle dos aparelhos de ar condicionado, foi decidido criar um hardware para substituir a atual controladora eletrônica dos aparelhos. Como base para o desenvolvimento deste hardware foi escolhida a plataforma de prototipagem de código aberto arduino, devido a sua simplicidade de integração com diferentes tipos de sensores. Outra facilidade que motivou a escolha do arduino como plataforma de desenvolvimento foi a sua capacidade de comunicação nas redes sem fio ZigBee, padrão que complementa o padrão IEEE 802.15.4, escolhido como padrão de comunicação devido as facilidades que proporciona para implementação de uma rede sem fio de baixo custo, baixas taxas de comunicação e baixo consumo de energia; capaz de trabalhar em uma topologia de rede em malha (*mesh*), endereçar mensagens a cada módulo especificamente e rotear mensagens através dos outros módulos.

Para permitir o controle dos aparelhos de ar condicionado pelos usuários de forma amigável escolheu-se o desenvolvimento de uma interface Web através de um Servlet Java. Sua escolha deu-se em função da existência de facilidades para interação na rede ZigBee, através de uma API para o controle de módulos XBee, e na rede TCP/IP, para disponibilização da interface Web.

O circuito de acionamento do ar condicionado e das velocidades do seu ventilador através dos protótipos, baseados nos acionamentos descritos pela documentação do modelo de aparelho de ar condicionado escolhido, foram testados com sucesso. O acionamento do compressor conforme a temperatura desejada e a temperatura coletada também foi testada com sucesso. Os testes de controle remoto dos protótipos através do servidor de aplicação também ocorreram corretamente.

Abstract

From the problem of energy waste due to air conditioning units unnecessarily turned on in the IFSC, campus São José, emerged the idea of developing a centralized air conditioning controlling system.

After the study about the basic operation of the electronic control system of air conditioning units, it was decided to create a hardware controller to replace the current electronic driver. The open source prototyping platform arduino, has been choosed as base platform for the hardware development, because of its easiness to integrate with different types of sensors. Another motive that encouraged the choice of arduino as a development platform was its ability to communicate with ZigBee wireless networks, which complements the standard IEEE 802.15.4, choosen as communication standard due to facilities to wireless networks implementation with low cost, low rate and low power consuption; capable of work in mesh topology, address messages to each model and route messages through other modules.

To allow the air conditioners units control by users, was decided to deploy a Web server. A Java Servlet was choosed to do this because of its easiness for interaction both in ZigBee network via an API to control XBee modules and for communication with the prototypes and in TCP/IP network.

The activation circuit of air conditioning units and his fan speeds, based on activations described by the air conditioner documentation, has been tested successfully. The compressor activation according to the desired and collected temperature also been tested sucessfully. The remote control tests of prototypes by the application server also been occured sucessfully.

Sumário

Lista de Figuras

1 Introdução

O conceito de sistemas de automação pressupõe a capacidade de executar comandos, obter dados, regular parâmetros e controlar tarefas automaticamente. Sua concepção passa também pela parte de integração, de forma que todos os dispositivos que fazem parte deste sistema automatizado atuem de forma inteligente, visando uma maior eficiência e aproveitamento de recursos tanto individualmente quanto em conjunto (PINHEIRO, 2004b).

A partir deste conceito desenvolveu-se a automação predial, que tem como conceito o uso dos recursos oferecidos pela natureza de maneira racional. Seu foco está na melhoria do cotidiano dos habitantes dos edifícios, buscando trazer mais conforto, segurança e eficiência ao ambiente em questão (PINHEIRO, 2004a).

Hoje a automação predial é uma tendência, levando em conta também a demanda pela racionalização do consumo de energia, além dos outros benefícios supracitados. O mercado está em crescente expansão e diariamente surgem novas tecnologias que possibilitam a elaboração de novos projetos na área. Exemplos de aplicações dentro dentro da automação predial são sistemas de controle de acesso, controle de iluminação, controle de temperatura e sistemas de monitoramento.

1.1 Motivação

No IFSC campus de São José enfrenta-se um grande problema devido a falta de controle sobre os aparelhos de ar condicionado, o desperdício de energia com unidades ligadas desnecessariamente. No cenário atual, não é possível saber quais aparelhos estão ligados, se há alguma aula sendo ministrada naquele ambiente ou ainda obter dados sobre a temperatura dos mesmos.

Este problema acaba gerando um gasto energético que poderia ser evitado, se fosse possível obter e enviar dados a esses aparelhos de forma centralizada.

1.2 Objetivos

A proposta deste trabalho foi desenvolver um protótipo que substitua o atual controlador eletrônico dos aparelhos de ar condicionado. Este seria responsável pela aquisição de informação dos sensores de temperatura e estado do aparelho, e pelo acionamento do compressor e das velocidades do ventilador.

O protótipo deveria ser capaz de transmitir e receber dados de um servidor, onde uma aplicação web tornaria a interação do usuário com os aparelhos amigável.

Para comunicação entre os protótipos e o servidor central foi definido o uso de uma tecnologia para comunicação de curta distância, capaz de rotear as mensagens até o destino e capaz de trabalhar em uma topologia em malha, evitando a perda de informação devido a protótipos que estivessem fora de operação na rede.

1.3 Organização do texto

O texto está organizado da seguinte forma: No Capítulo 2 é apresentada a fundamentação teórica sobre o funcionamento do controle de temperatura e acionamento dos aparelhos de ar-condicionado. Apresenta-se também o embasamento sobre a plataforma de prototipagem Arduino, suas características de funcionamento e sua integração com a *shield* XBee. Por último uma breve revisão do padrão IEEE 802.15.4 e sua extensão, o ZigBee, é apresentada. O Capítulo 3 traz o cenário e o protocolo desenvolvido para comunicação entre os dispositivos e o Servidor de Aplicação. O Capítulo 4 versa sobre o desenvolvimento do protótipo no cenário proposto. No Capítulo 5 estão as especificações sobre o funcionamento do servidor de controle dos aparelhos de ar condicionado. Finalmente, o Capítulo 6 apresenta as conclusões e os trabalhos futuros que podem ser derivados deste trabalho.

2 *Fundamentação Teórica*

2.1 Aparelho de ar condicionado

Os aparelhos de ar condicionado funcionam com base no ciclo de refrigeração, ocasião onde em um circuito fechado, o gás refrigerante (fluido que absorve calor de uma substância do recinto a ser refrigerado) convertendo-se sucessivamente em vapor e líquido consiga reter o calor a baixa temperatura e alta pressão pela evaporação e expelir calor a alta temperatura e alta pressão pela condensação (JUNIOR., 2003a).

Esse ciclo conta com quatro componentes essenciais observados na Figura 2.1:

- **Compressor:** Absorve e comprime o vapor refrigerante.
- **Condensador:** Condensa o vapor refrigerante vindo do compressor a alta pressão, levando-o ao estado líquido. Para isso rejeita o calor retido no fluido refrigerante para alguma fonte de resfriamento (JUNIOR., 2003b).
- **Válvula de expansão:** Diminui a pressão do sistema através de uma expansão isoentálpica (onde não há troca de calor com o meio externo) e controla a vazão de fluido refrigerante que chega ao evaporador.
- **Evaporador:** Absorve o calor latente de vaporização e envia para o compressor, dando início a um novo ciclo de refrigeração. É nele onde ocorre uma mudança de estado do fluido refrigerante de líquido para gasoso (JUNIOR., 2003b).

O modelo de ar-condicionado em que o estudo é baseado é o Split Console Silvermaxi da fabricante Springer[1], a unidade condensadora de ar modelo 38KQD e a unidade evaporadora de ar modelo 42XQC. Ele utiliza uma placa eletrônica para controle das suas funções como controle de temperatura, ajuste de umidade, velocidade do ventilador, estado do aparelho, etc.

[1] http://www.springer.com.br

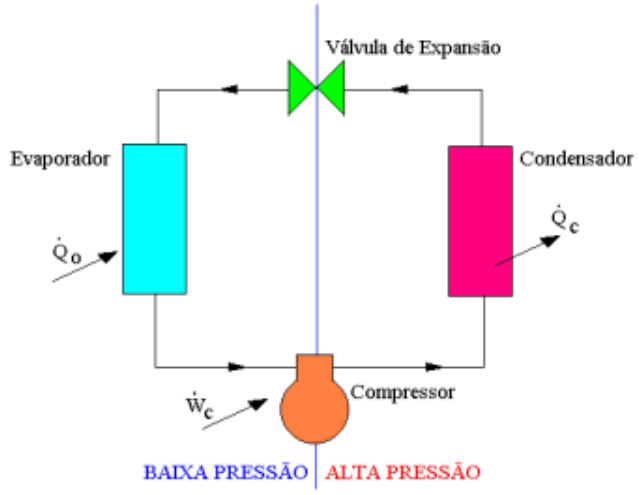


Figura 2.1: Esquema simplificado de refrigeração (JUNIOR., 2003a).

Nesta placa eletrônica é integrada uma interface de infravermelho que permite que sejam transmitidos comandos ao aparelho via controle remoto (SPRINGER, 2011).

Sua placa eletrônica possui conexões com o ventilador presente na evaporadora, que controla as três velocidades disponíveis de ventilação através de 3 fios (uma para cada velocidade) acionados por um relé. É ela também quem fornece o pino de conexão com o fase e o neutro da conexão elétrica. A placa eletrônica também possui conexão com a unidade condensadora, através de uma borneira (série de terminais para ligação de fios a um aparelho elétrico), que vem da condensadora. Por essa conexão é fornecido também o fase e o neutro para alimentar a condensadora, além do pino de acionamento do compressor, do motor do condicionador e da válvula reversora (para chaveamento quente/frio). O esquema das conexões completo pode ser visualizado no apêndice B (SPRINGER, 2011).

O seu funcionamento básico dar-se-á pelo chaveamento (liga/desliga) do compressor e do motor externo conforme a temperatura ambiente e a temperatura desejada. Para que não haja um chaveamento contínuo quando a temperatura oscila perto do limiar desejado é feito um controle de histerese eletrônica que desliga o compressor e o motor externo apenas quando a temperatura está alguns graus abaixo do limiar desejado.

2.2 Arduino

O projeto teve inicio em Ivrea na Itália em 2005. A ideia inicial era desenvolver uma plataforma de prototipagem de baixo custo para uso escolar. O arduino[2] foi desenvolvido em

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

hardware e software de código aberto baseado em uma simples placa com um microcontrolador e um ambiente de desenvolvimento (*Integrated Development Environment* – IDE) para escrita do software. Ele é uma derivação do Wiring[3]. O seu microcontrolador, da família ATmega, permite a atuação em uma enorme gama de atividades, desde o controle de interruptores, recuperação de informações de sensores, até controle de motores (MELLIS, 2009b).

2.2.1 Hardware

Sua placa base (Figura 2.2), que pode ser confeccionada artesanalmente (o esquema de montagem da placa pode ser baixado gratuitamente do site oficial[4]) ou ser adquirida pronta, consiste em um microcontrolador de 8-bits da Atmel e em conectores para facilitar o desenvolvimento de aplicativos e a interação com outros circuitos. O microcontrolador usado possui portas de entrada analógicas, portas para comunicação serial e entradas e saídas digitais (algumas destas saídas podem produzir sinais modulados por largura de pulso (*Pulse Width Modulation* – PWM)).

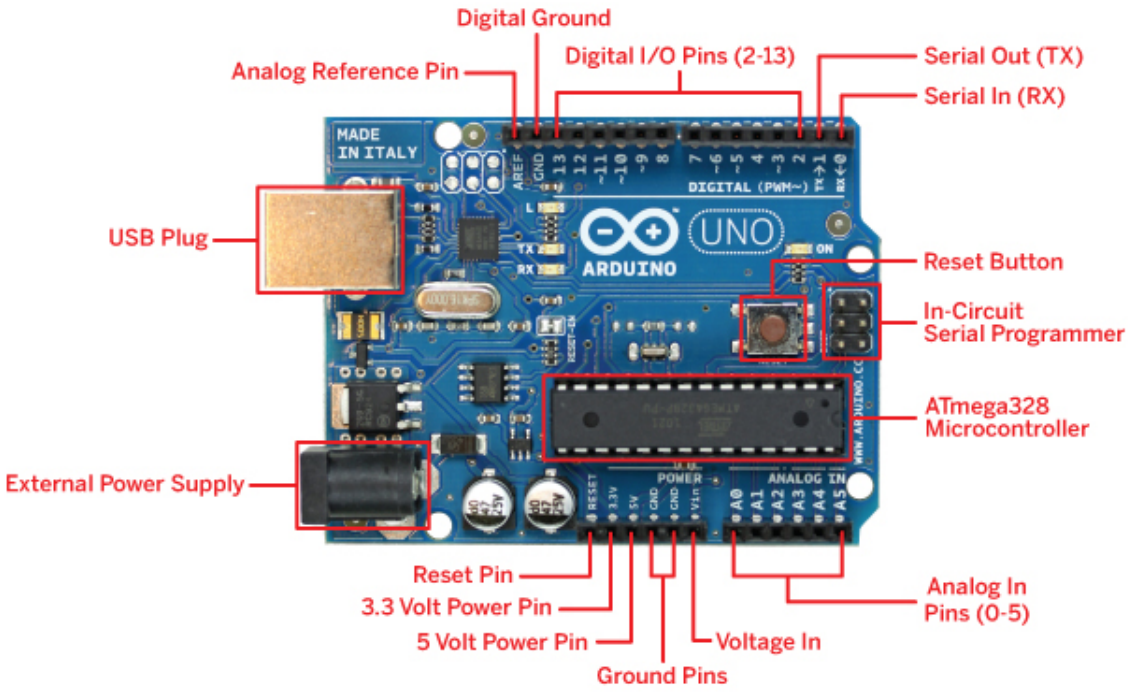


Figura 2.2: Placa base do kit Arduino e seus elementos.

Existem módulos que podem ser conectados diretamente as entradas da placa base, conhecidos como *shields*. Estes comunicam-se com o microcontrolador através dos pinos ou via o

[3]http://www.wiring.org.co
[4]http://arduino.cc/en/Main/ArduinoBoardSerialSingleSided3

barramento serial, que permite que vários *shields* sejam usados em paralelo. As *shields* permitem a expansão das funcionalidades do kit, pois possuem outras interfaces, como Ethernet ou Wi-Fi, que permitem a comunicação com outros padrões. Para a comunicação com as redes ZigBee é utilizado a *shield* XBee desenvolvida pela Digi, que permite a utilização do módulo XBee, também desenvolvido pela Digi, que possibilita a comunicação em redes ZigBee.

2.2.2 Software

O IDE do arduino é derivado do IDE usado na linguagem de programação Proocessing e no Wiring. Além de facilidades para programação, como o exemplo de código mostrado na Figura 2.3, o IDE permite também transferir o código compilado para o microcontrolador via porta serial. O software pode ser baixado gratuitamente do site oficial do projeto arduino[5]. Os projetos desenvolvidos nesta plataforma podem funcionar de forma independente ou em conjunto com aplicativos executando em um computador (Flash, Processing, MaxMSP).

A linguagem de programação utilizada para o desenvolvimento de aplicações para o arduino é baseada em C/C++ e encapsula diversos aspectos de baixo nível da programação do microcontrolador, o que torna a programação do microcontrolador mais amigável (MELLIS et al., 2007).

O exemplo mostrado na Figura 2.3 contém um código simples para piscar um LED conectado a uma porta digital do kit. As funções *setup()* e *loop()* são obrigatórias em todas as aplicações desenvolvidas para o Arduino. A função *setup()* é inserida no inicio da execução e pode ser utilizada para iniciar variáveis e configurar as portas do microcontrolador. Na função laço principal (*loop()*), que será repetida infinitas vezes, está toda a lógica da aplicação. As outras funções criadas só poderão ser chamadas de dentro destas duas funções.

2.3 IEEE 802.15.4

O padrão IEEE 802.15.4 (IEEE, 2006), desenvolvido e mantido pelo *Institute of Electrical and Electronics Engineers* (IEEE), faz parte de uma família de padrões de *Wireless Personal Area Network*s (WPANs) e estabelece parâmetros para redes sem fio privativas com baixas taxas de transmissão, baixo custo, curto alcance e baixo consumo de energia (*Low-Rate Wireless Personal Area Network*s – LR-WPANs). Este padrão surgiu devido algumas aplicações necessitarem de um meio de comunicação com baixo consumo de energia e baixo custo. Além disso

[5]http://arduino.cc/en/Main/Software

```
/*
  Blink
  Turns on an LED on for one second, then off for one second, repeatedly.

  This example code is in the public domain.
 */

// Pin 13 has an LED connected on most Arduino boards.
// give it a name:
int led = 13;

// the setup routine runs once when you press reset:
void setup() {
  // initialize the digital pin as an output.
  pinMode(led, OUTPUT);
}

// the loop routine runs over and over again forever:
void loop() {
  digitalWrite(led, HIGH); // turn the LED on (HIGH is the voltage level)
  delay(1000);             // wait for a second
  digitalWrite(led, LOW);  // turn the LED off by making the voltage LOW
  delay(1000);             // wait for a second
}
```

Figura 2.3: Exemplo de código para piscar um LED (MELLIS, 2009a).

possui outras características como o curto alcance de operação, fácil instalação, baixo custo e uma mínima infraestrutura necessária para sua implementação.

São utilizadas as bandas *Industrial, Scientific and Medical* (ISM), que não necessitam de licença, de 868 MHz (Europa), 915 MHz (América do norte) e 2450 MHz (Global) para a transferência de dados, podendo atingir taxas nominais de 20kb/s até 250kb/s.

2.3.1 Componentes de uma rede IEEE 802.15.4

Os dispositivos que podem participar de uma LR-WPAN podem ser classificados como, *Reduced-Function Device*s (RFDs) e *Full-Function Device*s (FFDs). Os FFDs podem operar em dois modos diferentes: como coordenador de uma rede pessoal local (*Personal Area Network* – PAN) ou no modo *device*, podendo se comunicar tanto com RFDs como com outros FFDs. Já os RFDs podem se comunicar apenas com outros FFDs, eles são projetados para dispositivos finais, que acessam a rede mas não fazem roteamento (IEEE, 2006).

2.3.2 Topologias de rede

O padrão suporta topologia em estrela e ponto a ponto (P2P), dependendo dos requisitos da aplicação, conforme a Figura 2.4 (IEEE, 2006). Na topologia em estrela, a comunicação é estabelecida entre os dispositivos e um controlador central único, chamado coordenador PAN, que pode possuir uma aplicação específica ou ser usado para iniciar, terminar ou rotear alguma mensagem dentro da rede.

A topologia P2P também possui um coordenador PAN, a diferença para a topologia em estrela é que todo dispositivo pode se comunicar com qualquer outro dispositivo no seu raio de alcance. Este tipo de topologia permite a formação de topologias de rede mais complexas como a topologia em malha (*mesh*). Esta possibilita múltiplos saltos para roteamento das mensagens de qualquer dispositivo até outro dentro da rede, função que deve ser implementada nas camadas superiores e não é tratada pelo padrão (IEEE, 2006).

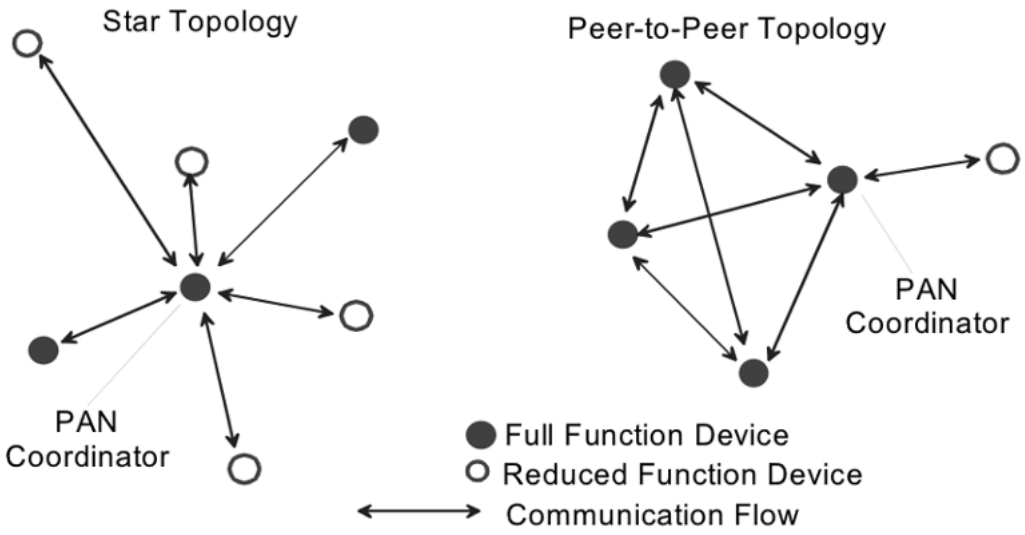


Figura 2.4: Exemplo de topologia em estrela e P2P (IEEE, 2006).

Todos os dispositivos operando em ambas topologias devem possuir um endereço único de 64 bits, usado para uma comunicação direta dentro da PAN. Um endereço menor, de 16 bits, pode ser alocado pelo coordenador PAN quando um dispositivo se associa a ele e também pode ser usado.

Cada PAN deve possuir um identificador único (PAN ID). Este permite a comunicação entre os dispositivos na rede utilizando o endereçamento de 16 bits e habilita a transmissão entre dispositivos em diferentes redes independentes. O mecanismo de escolha do identificador também não é tratado pelo padrão.

2.3.3 Arquitetura

A arquitetura IEEE 802.15.4 especifica as duas camadas mais baixas do modelo de cinco camadas TCP/IP. O padrão define o funcionamento da camada física (*Physical Layer* – PHY), que basicamente prove a transmissão e recepção no canal físico, e da subcamada de acesso ao meio (*Medium Access Control* – MAC), que provê o acesso aos canais físicos. Esta arquitetura pode ser vista na Figura 2.5 (IEEE, 2006).

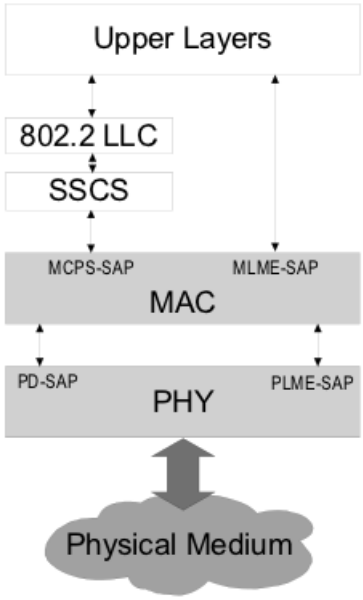


Figura 2.5: Arquitetura IEEE 802.15.4 no modelo TCP/IP (IEEE, 2006).

As camadas superiores, conforme a Figura 2.5, consistem na camada de rede, que provê as configurações de rede e roteamento de mensagens, e na camada de aplicação que provê a função pretendida do dispositivo. As definições destas camadas estão fora do escopo deste padrão e são abordadas por padrões que o utilizam, como o padrão ZigBee.

2.3.4 Estrutura do *superframe*

O padrão IEEE 802.15.4 permite a utilização de um estrutura de *superframe*, que propicia uma multiplexação no tempo (IEEE, 2006). O *superframe*, cujo formato é definido pelo coordenador, é divido em 16 *slots* de mesmo tamanho (Figura 2.6a), e delimitado pelos *beacons*, quadros de controle utilizados para sincronizar os dispositivos anexos na rede transmitidos no primeiro *slot* de cada *superframe* pelo coordenador. Além disso os *beacons* também são utilizados para identificação da PAN (PAN ID) e para descrição da estrutura dos *superframes*.

O *superframe* pode possuir opcionalmente uma porção ativa e inativa, conforme a Figura

2.6b. Durante a sua porção inativa o coordenador pode entrar em um modo de economia de energia, onde os dispositivos que fazem parte desta rede passam a maior parte do tempo em estado de sono (*sleep*) onde o rádio fica desligado, ouvindo periodicamente o meio para ver se há alguma mensagem pendente.

Qualquer dispositivo que deseja se comunicar durante o período de acesso com contenção (*Contention Access Period* – CAP) concorre com os outros dispositivos usando o modo de acesso com contenção *Carrier Sense Multiple Access With Collision Avoidance* (CSMA/CA). Se o coordenador decidir não usar a estrutura *superframe*, a transmissão dos *beacons* será encerrada.

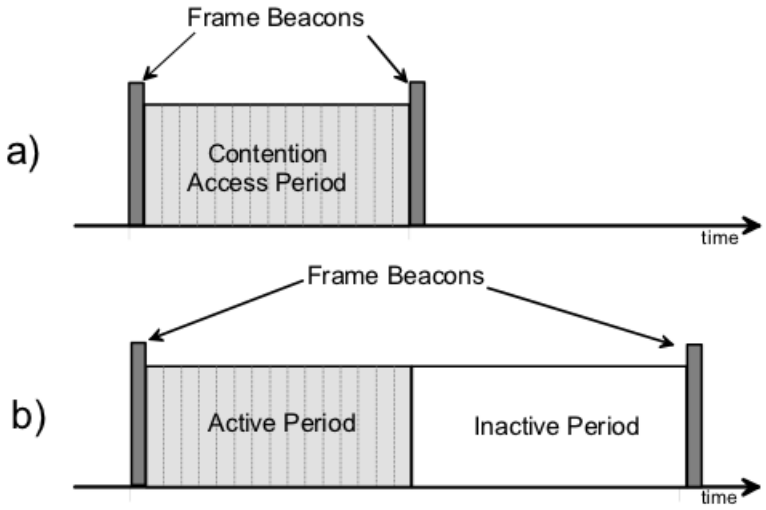


Figura 2.6: Estrutura *superframe* sem compartimentos garantidos (IEEE, 2006).

Para aplicações que necessitam de baixa latência ou uma largura de banda garantida, o coordenador PAN pode reservar uma parte do *superframe* ativo, chamada de *Guaranteed Time Slot*s (GTSs). Esses GTSs formam o *Contetion-Free Period* (CFP), que inicia no *slot* logo após o CAP (Figura 2.7). O coordenador PAN pode alocar até sete GTSs e os GTSs podem ocupar mais de um *slot*, levando em conta o limite de 7 *slots* do CFP. Nenhuma transmissão no CFP deve utilizar o CSMA/CA para o acesso ao meio, os dispositivos devem assegurar que suas transmissões completem um período IFS (*Interframe Space or Spacing*), período necessário para que a sub-camada MAC processe os dados recebidos pela camada PHY usado entre os quadros para que eles possam ser transmitidos sucessivamente, antes do fim do GTS. Uma porção suficiente do CAP é assegurada para o acesso baseado em contenção e outros dispositivos que queiram se conectar a rede.

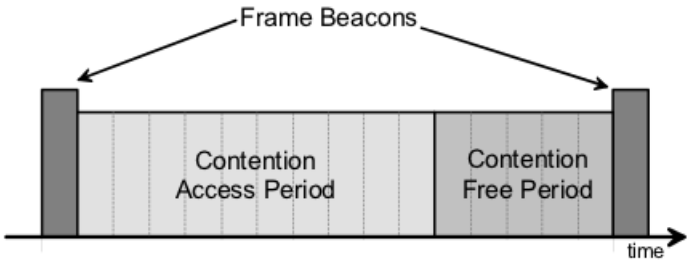


Figura 2.7: Estrutura *superframe* com compartimentos garantidos (IEEE, 2006).

2.3.5 Consumo de energia

O padrão IEEE 802.15.4 foi desenvolvido para dispositivos com baixo consumo energético. Os dispositivos podem entrar em um estado de espera (*sleep*), que proporciona uma economia de energia essencial para dispositivos alimentados por baterias. Neste estado de *sleep*, o dispositivo escuta o meio periodicamente para determinar se a uma mensagem pendente, podendo desligar o rádio durante o período em que não escuta o meio. Com isso, é possível, dependendo da aplicação, escolher entre baixo consumo de energia ou uma baixa latência (IEEE, 2006).

2.3.6 Segurança

Por ser um padrão que visa o baixo custo, os dispositivos que fazem parte deste possuem capacidade limitada em termos de poder de processamento, espaço de armazenamento e quantidade de energia; e pode-se dizer que não possuem uma base computacional confiável ou um gerador de números aleatórios de alta qualidade (IEEE, 2006).

O mecanismo de criptografia usado pelo padrão é baseado em chave simétrica (*Advanced Encryption Standard* (AES) em modo *Counter* (CTR)), utilizando chaves providas por processos das camadas superiores (IEEE, 2006). A criação e manutenção destas chaves não é abordada pelo padrão IEEE 802.15.4, sua implementação pelo padrão ZigBee poderá ser observada no item 2.4.4.

Este mecanismo provê a confidencialidade dos dados, segurança da autenticidade da fonte transmissora de dados e detecção de duplicidade de informação. Em sistemas que necessitam de uma maior segurança, como por exemplo sistemas de controle de acesso a portas ou alarmes, é recomendado complementar a segurança utilizando outros elementos de segurança nas camadas superiores, como implementando um sistema de controle de acesso.

2.4 ZigBee

O ZigBee (ZIGBEE ALLIANCE, 2008) (KINNEY, 2003), desenvolvido e mantido pela ZigBee Alliance[6], é um padrão que complementa o padrão IEEE 802.15.4 e define as camadas superiores não abordadas por este padrão. As camadas implementadas pelo ZigBee são a camada de rede e uma sub-camada de suporte a aplicação (Figura 2.8).

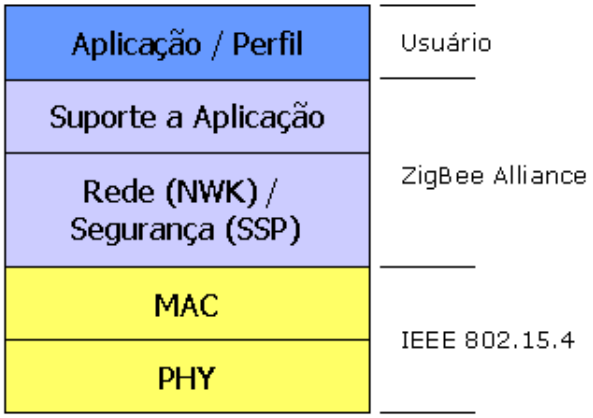


Figura 2.8: Camadas do protocolo ZigBee (FRIAS, 2012).

Ele é destinado a aplicações embarcadas que visem baixo consumo energético e baixas taxas de transferência de dados. Possui grande aplicação em redes de sensores sem fio, em função ao baixo custo e a possibilidade de implementação em uma topologia em malha (*mesh*), que propicia uma maior confiabilidade e maiores alcances.

2.4.1 Arquitetura ZigBee

O padrão divide os dispositivos em três classes, de acordo com a sua função na rede (Figura 2.9) (KINNEY, 2003). O coordenador ZigBee é o dispositivo responsável pela inicialização e manutenção dos dispositivos na rede ZigBee. É ele quem transmite os *beacons*, controla e armazena informações sobre os nós (dispositivos) da rede. Ele também faz o roteamento das mensagens entre os nós e pode operar como receptor de dados.

Em redes com topologia em malha (*mesh*) ou em árvore, a rede pode ser estendida através de roteadores ZigBee. Nas redes em árvore eles são responsáveis por transportar dados e mensagens de controle pela rede usando uma estratégia de roteamento hierárquico. Nas redes *mesh* é possível uma comunicação P2P completa, com o roteamento baseado no algoritmo *Ad Hoc On-Demand Distance Vector* (AODV).

[6] `http://www.zigbee.org`

Todos os outros dispositivos, conhecidos como dispositivos finais (*end-devices*), se comunicam com os coordenadores, mas não possuem nenhuma função de roteamento na rede.

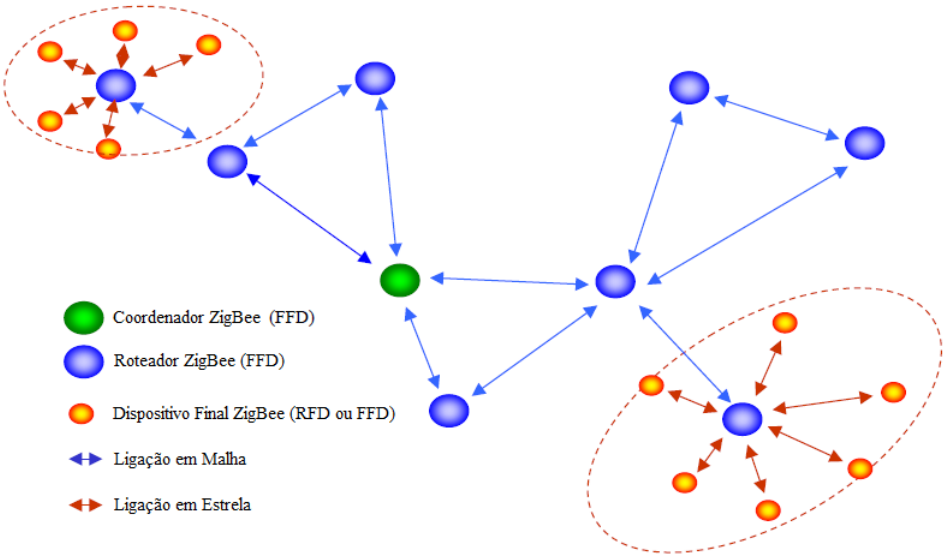


Figura 2.9: Componentes de uma rede ZigBee (KINNEY, 2003).

2.4.2 Camada de rede

A camada de rede neste padrão contém um grande conjunto de responsabilidades como: estabelecer uma nova rede, permitir o ingresso ou desassociação de um dispositivo a rede, configurar um novo dispositivo para operação, habilitar um coordenador ZigBee a associar os endereços dos novos dispositivos que se associarem a rede, permitir a sincronização com outros dispositivos através do rastreamento de *beacons* ou por votação, aplicar a segurança aos quadros enviados e remover a segurança dos quadros recebidos e efetuar o roteamento dos pacotes aos seus respectivos destinos.

O algoritmo de roteamento do ZigBee possibilita o equilíbrio entre o custo, consumo de energia e complexidade, de acordo com as especificações da aplicação (ZIGBEE ALLIANCE, 2008). No AODV a rede permanece passiva até que uma transmissão seja iniciada, a partir deste ponto o nó da rede que quer se comunicar envia uma mensagem em *broadcast* requisitando uma conexão. Os outros nós encaminham esta mensagem, armazenando o nó do qual recebeu a mesma. Quando um nó receber esta mensagem e já possuir uma rota para o nó de destino desejado, ele retorna uma mensagem ao nó que iniciou a comunicação por uma rota temporária. O nó que fez a requisição inicia então a comunicação com o nó desejado utilizando a rota recebida que possuir o menor número de saltos sobre outros nós. Entradas não utilizadas nas tabelas de roteamento são recicladas após um determinado tempo. Quando há alguma falha

através de uma determinada rota, é retornado de erro no roteamento ao nó que esta transmitindo e o processo de estabelecimento de rota é refeito (PERKINS; BELDING-ROYER; DAS, 2003).

2.4.3 Camada de aplicação

A camada de aplicação do protocolo ZigBee é formada por 3 partes: a sub-camada de suporte a aplicação, o *ZigBee Device Object* (ZDO) e os objetos definidos especificamente pelo desenvolvedor da aplicação (ZIGBEE ALLIANCE, 2008).

Entre os seus serviços estão o de manter as tabelas para o *binding*, habilidade de combinar dois dispositivos baseado nos seus serviços e necessidades e encaminhar mensagens entre dispositivos diretamente ligados. Outro serviço é o de descoberta (habilidade de determinar quais dispositivos estão operando no espaço de operação do dispositivo).

As responsabilidades do ZDO incluem determinar a função do dispositivo na rede (coordenador ZigBee, roteador ZigBee ou *end-device*), iniciar e/ou responder as requisições *binding* e estabelecer uma relação segura entre os dispositivos na rede.

O ZigBee possui ainda uma *Application Programming Interface* (API), conjunto de rotinas estabelecidos para interação com softwares que apenas desejem utilizar as suas funcionalidades, que permite que a inicialização de dispositivos de rede ZigBee e a transmissão de mensagens sejam através dos mesmos seja feita sem que seja necessário entrar em detalhes de implementação do ZigBee (FALUDI, 2011).

2.4.4 Segurança

O ZigBee faz a segurança das mensagens transmitidas sob um único salto usando o mecanismo de segurança da camada MAC, mas para mensagens com múltiplos saltos delega a segurança as camadas superiores (KINNEY, 2003). Possui confiabilidade na troca de mensagens, devido a ser concebido de forma simples, usando o algoritmo de criptografia AES e adotando o algoritmo de segurança para o roteamento AODV.

O AODV em conjunto com o AES expõem uma gama de rotinas de segurança, em virtude da possibilidade de combinações para o estabelecimento de rotas e ações seguras (por padrão 32 bits, mas vai até 128 bits). Esses conjuntos tem por objetivo a manutenção da integridade de segurança, tendo em vista que a camada MAC é responsável pelo processamento de segurança, e que as camadas superiores controlam esse processo, ajustando as chaves de criptografia e determinando os níveis de segurança que deverão ser usados (GISLASON, 2008).

2.5 XBee

Os módulos XBee, desenvolvidos pela Digi International[7], proveem conectividade sem fio de baixo custo para dispositivos em redes *mesh* ZigBee (DIGI INTERNATIONAL, 2012). O seu hardware (Figura 2.10 consiste de uma pequena placa com transmissores capazes de se comunicar nas redes ZigBee. Há ainda uma variedade de antenas que podem ser usadas de acordo com o modelo do módulo que vai ser usado.

Figura 2.10: Módulo XBee.

Atualmente os módulos são divididos em duas versões: a Série 1 que são os primeiros módulos desenvolvidos para prover comunicação P2P e não são capazes de prover comunicação *mesh* de forma nativa, e a Série 2 que habilita a comunicação em redes com topologia *mesh* de forma que não seja necessário implementar o roteamento manualmente. Ambas as séries possuem dois tipos diferentes de potência de transmissão, a série regular que prove maior economia de energia, e a série PRO, que proporciona maior alcance de cobertura (porém com maior gasto energético) (FALUDI, 2011).

2.5.1 Modos de operação

O módulos XBee podem operar em dois modos:

- **Modo AT (Transparente):** O módulo atua como um substituto da porta serial. Tudo que seria enviado pelo pino de comunicação serial é enviado via antena de rádio frequência (RF) de forma transparente. A vantagem deste modo é a fácil configuração, já que basta configurar um dos rádios para atuar como coordenador e os outros como roteadores/*end-devices*, e deixar todos dentro do mesmo PAN-ID que eles já conseguem se comunicar.

[7] http://www.digi.com

Neste modo não é possível encaminhar mensagens a um módulo especifico, além disso também não é possível encaminhar mensagens através de outros módulos.

- **Modo API:** Sua configuração é mais complexa, comparado ao modo AT, porém permite a comunicação de forma estruturada devido as mensagens serem tratadas como pacotes. Nestes quadros API XBee além dos dados que devem ser transmitidos, constam informações como o endereço de origem e destino, o que possibilita o endereçamento das mensagens a pacotes específicos. Neste modo também é possível que as mensagens sejam encaminhadas através de outros módulos até o seu destino (FALUDI, 2011).

Para configuração do módulo a Digi disponibiliza o software X-CTU[8], capaz de configurar desde a função do dispositivo na rede até questões de segurança. Os detalhes do uso do X-CTU podem ser obtidos em seu manual[9].

[8] http://www.digi.com/support/productdetail?pid=3352
[9] http://ftp1.digi.com/support/documentation/90001003_A.pdf

3 Cenário

3.1 Arquitetura do sistema

O arquitetura do sistema (Figura 3.1) foi desenvolvida de forma centralizada. Existem dois componentes básicos no nosso sistema, o Servidor de Aplicação e os protótipos.

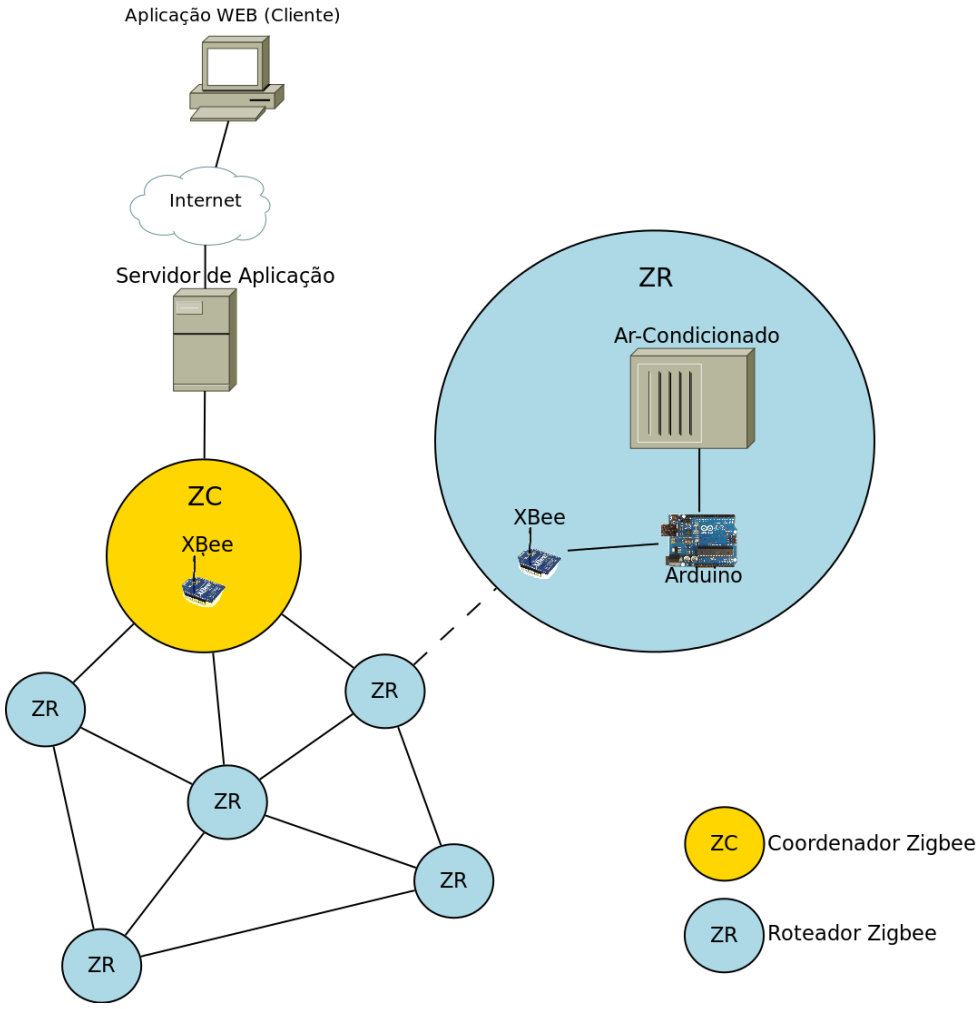


Figura 3.1: Arquitetura do sistema.

O Servidor de Aplicação é o responsável por disponibilizar a interface Web para interação dos usuários com os dispositivos. Ele também possui uma interface para comunicação com

protótipos na rede ZigBee, através de um módulo XBee, e foi definido como o Coordenador ZigBee (ZC). De acordo com as configurações definidas pelo usuário via interface Web são enviadas mensagens pelo servidor de aplicação aos protótipos através do módulo XBee.

Os protótipos atuam de forma reativa, apenas respondendo as requisições feitas pelo servidor de aplicação , seja com ações como desligamento do aparelho ou com o retorno de informações como a temperatura coletada pelo sensor. Estas requisições são recebidas pelo módulo XBee, que são configurados para atuar como Roteador ZigBee (ZR). Eles possuem conexões com os aparelhos de ar condicionado e efetuam a parte de chaveamento do compressor, controle do ventilador e controle de estado dos mesmos.

3.2 Comunicação

Para troca de informações entre os protótipos e o Servidor de Aplicação foi desenvolvido um protocolo para conexão, requisição de informações e envio de comandos. As mensagens desse protocolo são enviadas no *payload* do quadro de dados do ZigBee.

3.2.1 Estrutura do protocolo

O protocolo é estruturado como um vetor, o primeiro campo deste vetor armazena o tipo de mensagem que esta sendo transmitida, o segundo campo armazena o tipo de dispositivo destino desta mensagem e do terceiro campo em diante são armazenados os dados da mensagem (*payload*).

3.2.2 Tipos de mensagens

Foram criadas diferentes mensagens visando facilitar a interação entre o Servidor de Aplicação e os protótipos. As mensagens podem ser dos tipos:

- **CONEXAO:** Observada na Figura 3.2, é enviada pelo Servidor de Aplicação quando é feita a varredura a procura de novos dispositivos na rede. É também enviada pelo protótipo ao Servidor de Aplicação em resposta ao pedido de conexão enviado pelo mesmo.

Figura 3.2: Sequência de troca de mensagens de conexão.

- **REQUEST_INFO:** Observada na Figura 3.3, é enviada pelo Servidor de Aplicação para requisição de informações sobre o protótipo.
- **RESPONSE_INFO:** Observada na Figura 3.3, é enviada pelo protótipo em resposta a requisição de informações do Servidor de Aplicação.

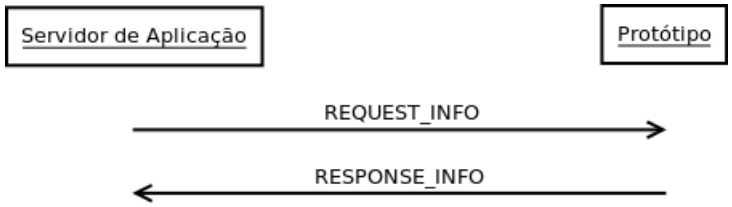


Figura 3.3: Sequência de troca de mensagens de requisição de informações entre o Servidor de Aplicação e os protótipos.

- **MUDA_ESTADO:** Observada na Figura 3.4, é enviada pelo Servidor de Aplicação para requisição de mudança de estado (ligado/desligado) do protótipo.

Figura 3.4: Sequência de troca de mensagens para mudança de estado do aparelho de ar condicionado.

- **MUDA_TEMPERATURA:** Observada na Figura 3.5, é enviada pelo Servidor de Aplicação para requisição de mudança de temperatura do protótipo.

Figura 3.5: Sequência de troca de mensagens para mudança de temperatura.

- **MUDA_VELOCIDADE:** Observada na Figura 3.6, é enviada pelo Servidor de Aplicação para requisição de mudança de velocidade do protótipo.

Figura 3.6: Sequência de troca de mensagens para mudança da velocidade do ventilador interno.

3.2.3 Tipos de dispositivo

O protocolo foi pensado de modo a suportar outros tipos de dispositivos futuramente, portanto este campo serve para indicar o tipo de dispositivo que o protótipo é. A princípio há apenas o tipo AR_CONDICIONADO.

4 Protótipos

4.1 Montagem

A montagem do protótipo foi feita conforme o esquema da Figura 4.1.

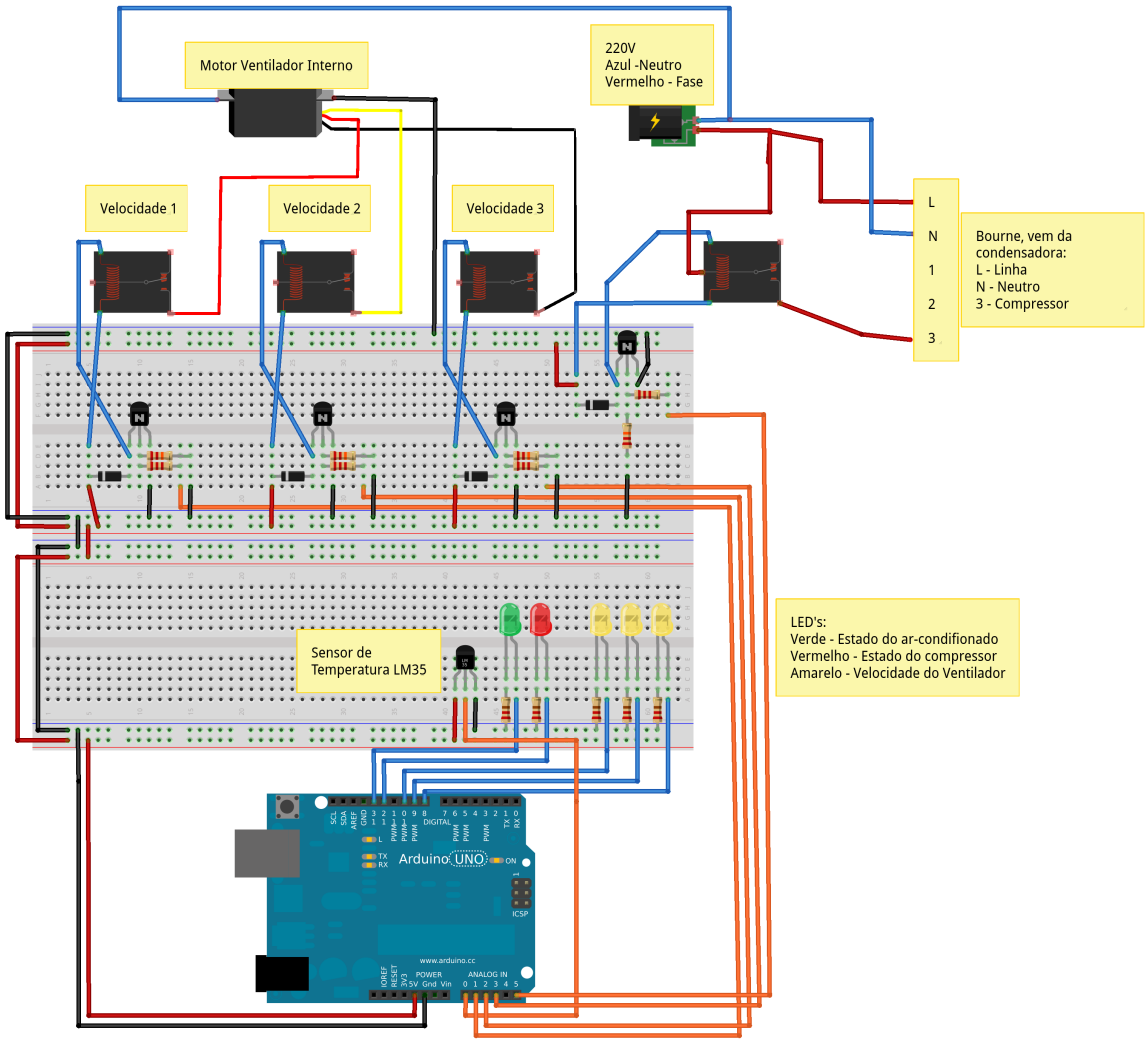


Figura 4.1: Esquema de montagem do protótipo.

O protótipo é capaz de ligar/desligar o aparelho de ar condicionado, acionar as velocidades do ventilador e controlar o funcionamento do compressor e do motor externo. Ele também coleta a temperatura ambiente e disponibiliza *leds* para informação dos seus estados.

Para aquisição da temperatura ambiente foi utilizado um sensor de temperatura LM35. Seu pino de saída foi ligado a uma entrada analógica do arduino, o de alimentação foi conectado aos 5V fornecidos pelo próprio arduino e o de terra conectado diretamente ao terra. É possível observar a configuração do LM35 na Figura 4.2.

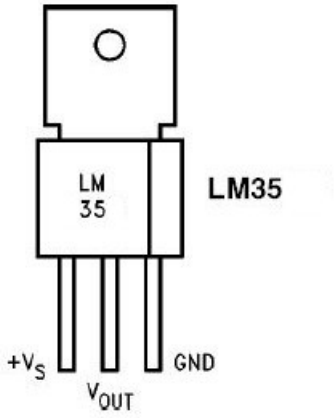


Figura 4.2: Configuração do sensor de temperatura LM35.

Leds foram usados para sinalização dos estados do protótipo. Um foi usado para indicar o estado de operação do aparelho, outro para indicar o estado de operação do compressor e outros três para indicação da velocidade do ventilador. A ligação dos *leds* foi feita conectando o ânodo do *led* em uma das portas digitais do arduino e o cátodo ligado em série com um resistor de 220 Ω ao terra.

Para o controle das velocidades do ventilador e do estado de operação do compressor foi necessária a elaboração de um circuito de acionamento, levando em conta as diferentes tensões de operação entre o arduino e o aparelho de ar condicionado. Conforme pode ser observado na Figura 4.3 foi usado um transistor NPN BC547B controlado por uma porta de saída analógica do arduino. Este por sua vez aciona um relé PCE-106D1H, que aciona o circuito de ativação de uma das velocidades do ar-condicionado ou do compressor e do motor externo. Foi colocado um diodo 1N4148 em paralelo ao relé para proteção do transistor, garantido que não haja sobretensão no transistor quando este for desativado. Os *datasheets* dos componentes transistor NPN BC547B, relé PCE-106D1H e diodo 1N4148 podem ser observados no Apêndice C.

A montagem final do protótipo pode ser observada na Figura 4.4

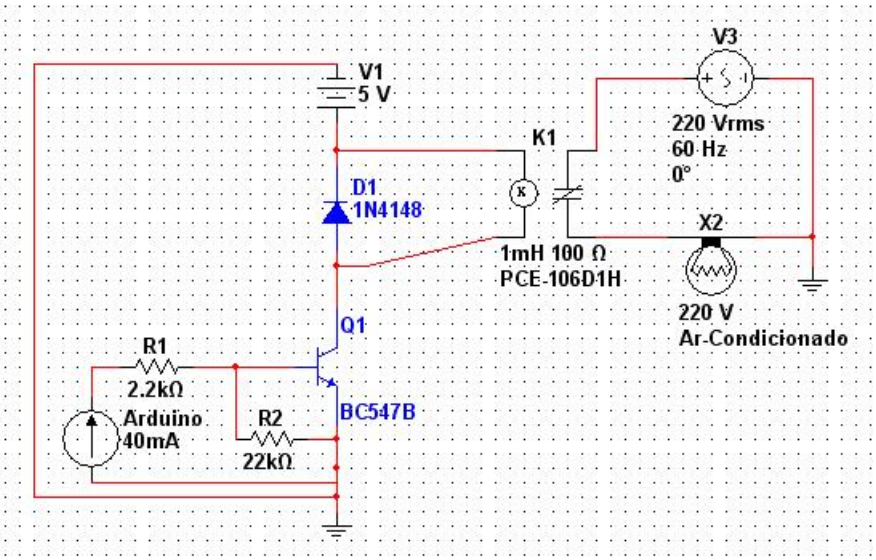


Figura 4.3: Circuito de acionamento das velocidades do ventilador e do estado do compressor e do motor externo.

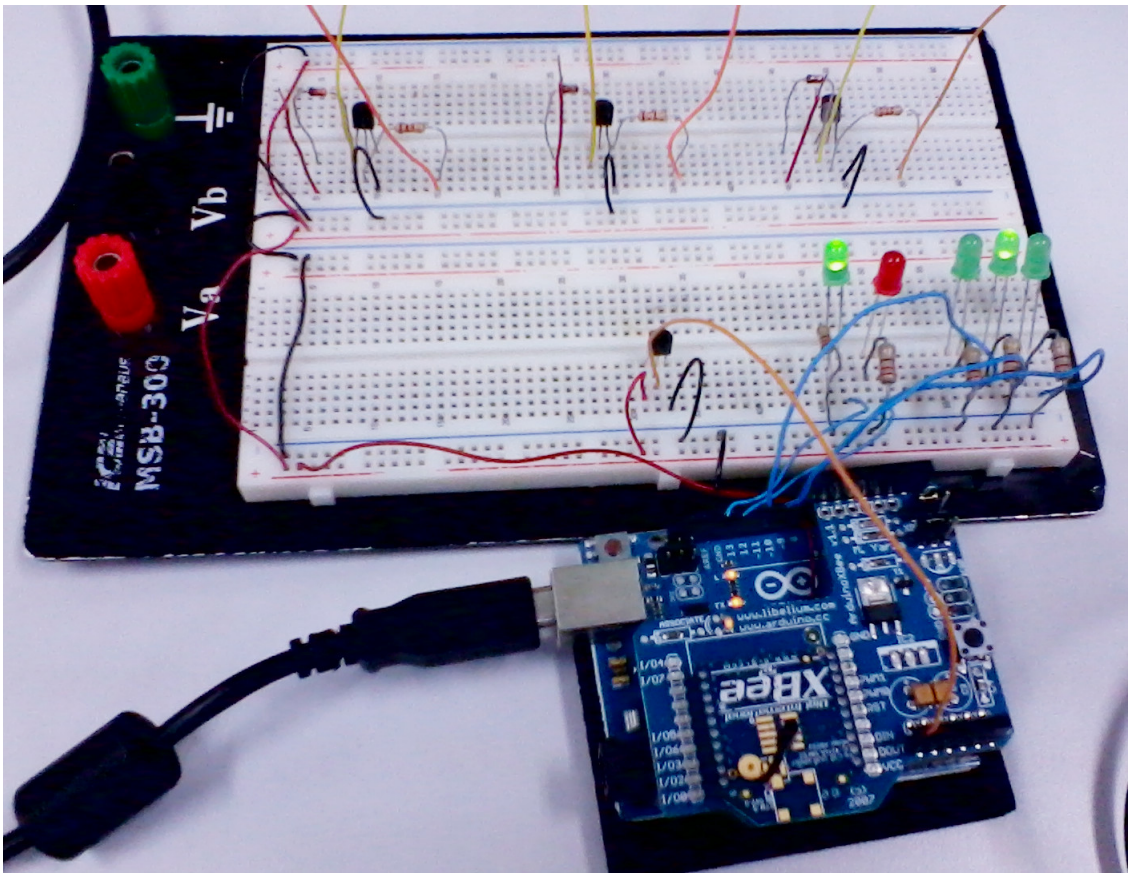


Figura 4.4: Protótipo montado.

4.2 Arduino

Para o projeto dos protótipos foram utilizados placas arduino do modelo Uno, devido a disponibilidade das mesmas no IFSC.

Os protótipos atuam de acordo com os comandos recebidos pelo Servidor de Aplicação. Após a fase de inicialização (*setup*) dos pinos do arduino de acordo com as suas funções e da interface serial para comunicação via serial com o módulo XBee, sua rotina principal (Figura 4.5) inclui verificar regularmente se há alguma mensagem recebida pela rede XBee e processar as mensagens enviadas (conforme visto no item 3.2.2). Além disso, quando o estado selecionado pelo usuário for o ligado, o protótipo controla o chaveamento do compressor e do motor externo de acordo com a temperatura desejada e coleta a temperatura do sensor de temperatura LM35. Nela também é acionada a velocidade do ventilador selecionada pelo usuário, e é feito um controle de histerese para que não haja chaveamento continuo do compressor quando há uma pequena oscilação na temperatura coletada pelo sensor.

Para comunicação na rede ZigBee através do módulo XBee foi utilizada a API xbee-arduino[1] que além da comunicação através do módulo, permite acesso as informações das mensagens como o endereço de origem e o *payload* (informação transmitida). O código completo utilizado para programação dos protótipos se encontra no link *http://goo.gl/BYd8X*.

4.3 Módulo XBee

O módulo XBee foi acoplado ao arduino através da *shield* XBee, que permite a instalação do módulo XBee sem a perda de funcionalidades da placa arduino. Para configuração do módulo XBee foi utilizado o software X-CTU disponibilizado pela Digi em conjunto com a unidade XBee Explorer USB (Figura 4.6), que através de um cabo mini USB permite a comunicação com o módulo. Os passos de configuração podem ser observados no Apêndice A.

Feito isto bastou acoplar o módulo XBee a *shield*, acoplar a *shield* ao arduino e colocar os *jumpers* da *shield* no modo XBee, conforme visto na Figura 4.7.

Para verificar a comunicação entre os módulos XBee, inicialmente realizamos testes de comunicação no modo AT (transparente), que exigem uma configuração bastante simples. Fizemos um teste simples, pelo próprio X-CTU, enviando uma mensagem em *broadcast* do coordenador ZigBee e observando as mensagens recebidas por outro módulo também através do

[1] http://code.google.com/p/xbee-arduino/

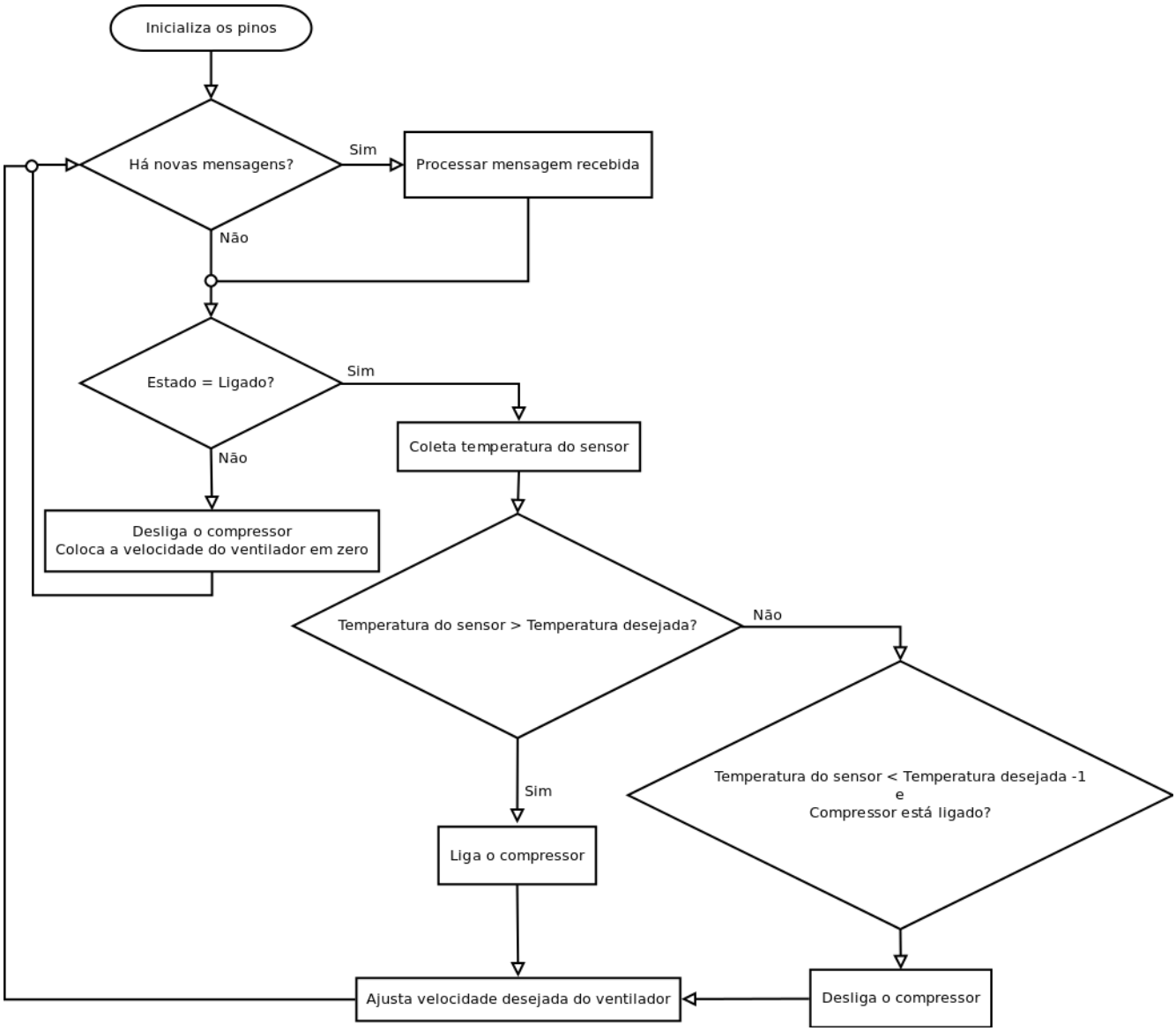


Figura 4.5: Fluxograma de funcionamento dos protótipos.

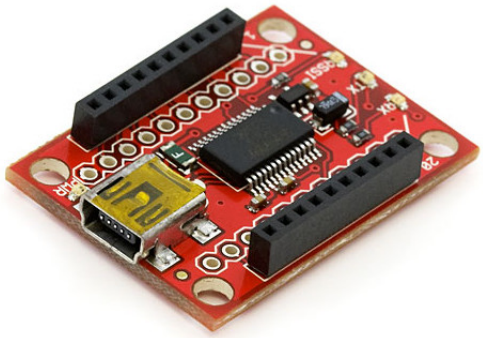


Figura 4.6: XBee Explorer USB.

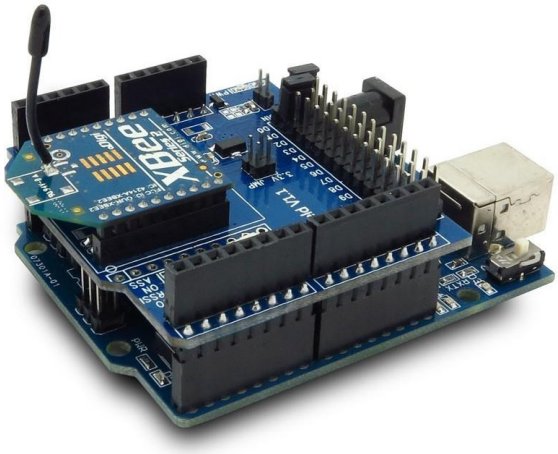


Figura 4.7: Arduino com a *shield* e o módulo XBee acoplados.

X-CTU. Estes testes ocorreram conforme o esperado. Este modo porém mostrou-se limitado por não permitir o endereçamento de mensagens especificas a cada módulo.

Partimos então para a implementação dos testes no modo API, utilizando um código nos arduinos que fazia um *led* piscar a cada mensagem recebida pelo módulo XBee endereçada ao mesmo. Utilizando três arduinos em conjunto com módulos XBee, configuramos um deles como coordenador enviando uma mensagem a um dos outros dois módulos, configurados no modo roteador (Figura 4.8). Para verificar o roteamento colocamos o módulo roteador que não iria receber a mensagem entre o coordenador e o outro módulo, a uma distância em que era possível verificar se a comunicação cessaria ao desconectar o módulo do meio. Estes testes ocorreram conforme o esperado.

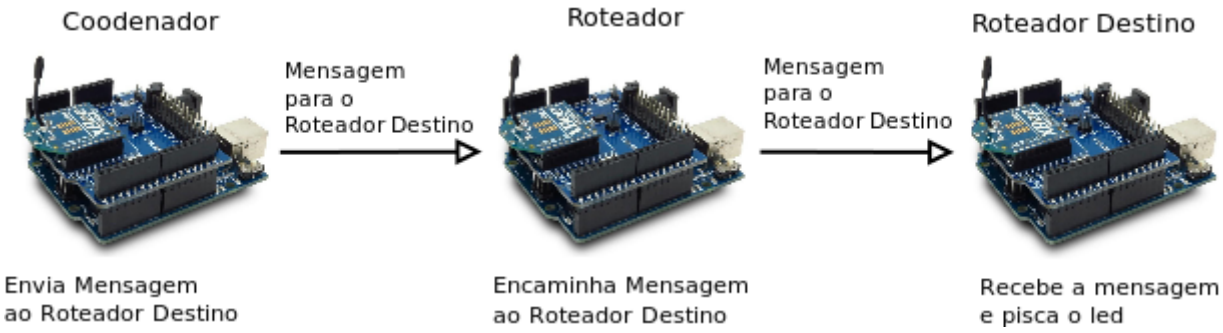


Figura 4.8: Cenário de teste do modo API.

5 *Servidor de Controle*

5.1 Servidor de Aplicação

Como Servidor de Aplicação utilizamos um computador pessoal com GNU/Linux Ubuntu[1] Desktop 10.04 LTS x86. Os dados dos protótipos foram inicialmente coletados através da interface serial do arduino (via cabo USB) e disponibilizados em uma interface Web, utilizando o servidor Web Apache[2] em conjunto com a linguagem de programação PHP (*PHP: Hypertext Preprocessor*)[3].

Quando partimos para os testes de implementação com os módulos XBee no modo API, a solução escolhida para disponibilização dos dados, servidor Web em conjunto com a linguagem de programação PHP, mostrou-se bastante complexa devido a não existência de uma API específica, até o momento do estudo, para o gerenciamento dos módulos XBee via PHP.

Como a aplicação precisava comunicar-se tanto na rede ZigBee (para comunicação com os protótipos) quanto na rede TCP/IP (para disponibilização da interface de controle web), escolheu-se o servidor de aplicação Glassfish. Este permite a criação de *servlets*, que são programas escritos em Java[4] com acesso direto as requisições HTTP recebidos pelo Servidor de Aplicação.

Para comunicação com a rede ZigBee foi utilizada a API xbee-api[5], que possui funções prontas para conexão e troca de mensagens na rede ZigBee, através do módulo XBee.

[1] http://www.ubuntu.com/
[2] http://www.apache.org/
[3] http://www.php.net
[4] http://www.oracle.com/technetwork/java/index.html
[5] http://code.google.com/p/xbee-api/

5.2 Módulo XBee

Para comunicação com a rede XBee no servidor foi utilizado um módulo XBee acoplado a uma unidade XBee Explorer USB, que através de um cabo mini USB permite a comunicação com o módulo.

Para sua configuração foi utilizado o software X-CTU. A configuração ocorreu conforme feita com os protótipos no item 4.3, exceto a função do módulo, que será *ZNET 2.5 COODINATOR API* (Figura 5.1).

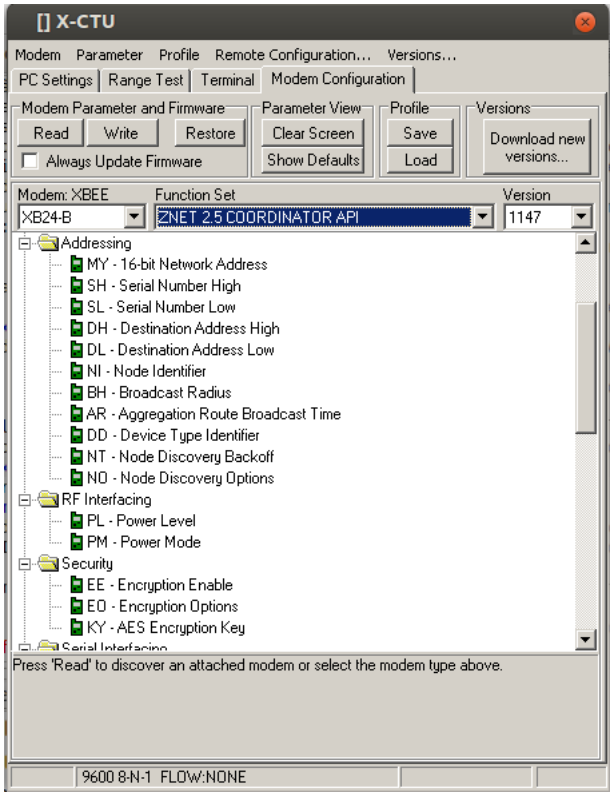


Figura 5.1: Módulo de configuração do coordenador - X-CTU.

5.3 Lógica de funcionamento

O Servidor de Aplicação atua como coordenador dos protótipos. O código para interação com os módulos XBee foi confeccionado usando parte do código desenvolvido no projeto Droidlar (EUZEBIO, 2011).

É possivel observar o funcionamento geral do Servidor de aplicação na Figura 5.2.

Na classe *Principal.java* além de disponibilizadas as informações sobre os dispositivos conectados, são processadas as requisições feitas pelo usuário (Figura 5.3), através da interface

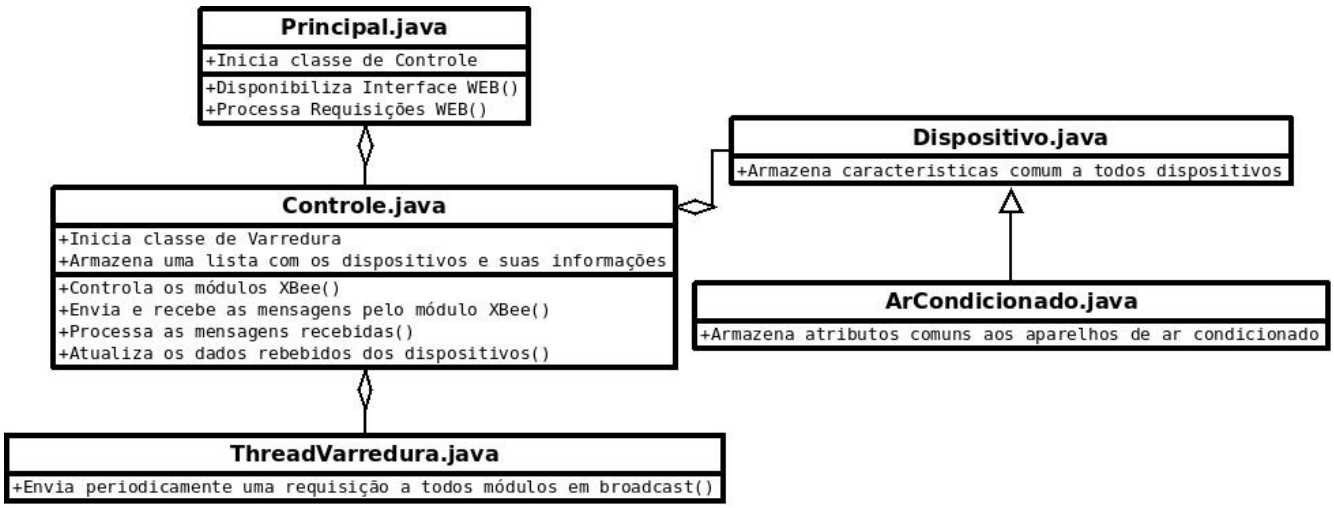


Figura 5.2: Funcionamento Servidor de Aplicação.

Web (Figura 5.4).

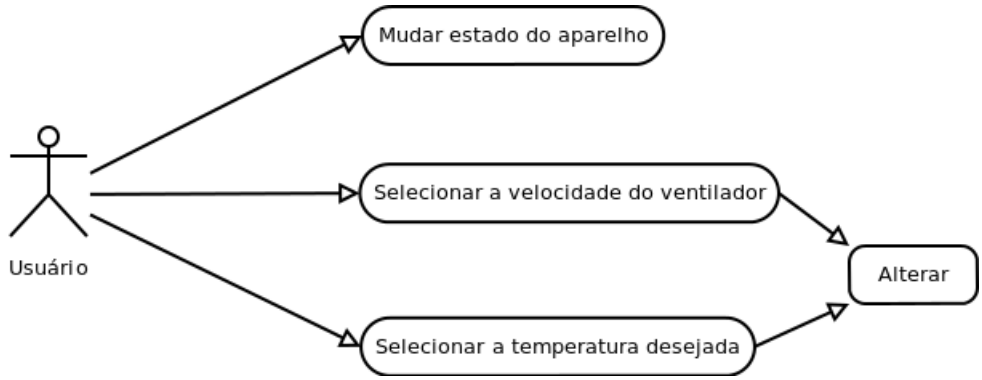


Figura 5.3: Casos de uso.

Nela é inicializada a classe *Controle.java*, responsável pelo controle do módulo XBee conectado ao servidor, além do envio e recebimento das mensagens XBee. Para este controle sobre o módulo XBee foi utilizado a API xbee-api[6]. Nesta classe são processadas as mensagens recebidas e é feita a verificação e atualização dos dados dos dispositivos.

Na classe *Principal.java* também é iniciada a classe *ThreadVarredura.java*, que é responsável por enviar uma requisição de conexão a todos os módulos em *broadcast* a cada 5 minutos. Os módulos que já estavam conectados descartam este pedido de conexão, para estes é enviado uma requisição de atualização dos dados para verificar se o módulo continua operando.

[6]http://code.google.com/p/xbee-api/

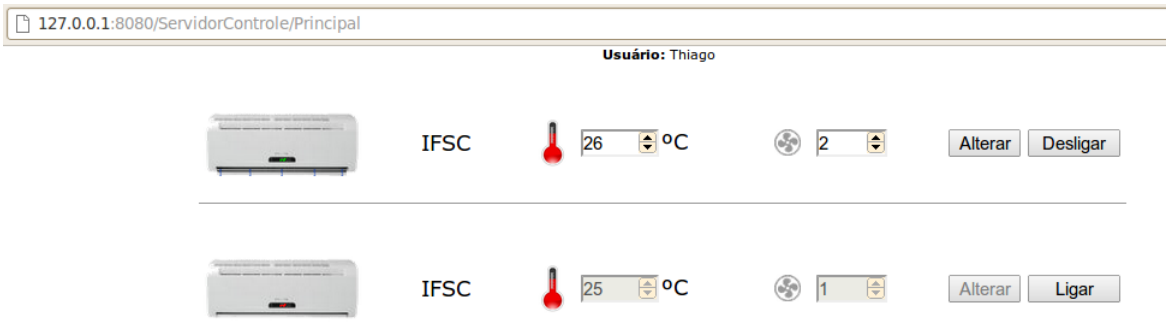


Figura 5.4: Interface Web.

6 *Conclusões*

Partindo do problema de desperdício energético em função de aparelhos de ar condicionados ligados sem necessidade no campus do IFSC de São José, propomos desenvolver um sistema de controle centralizado de aparelhos de ar condicionado. A ideia era substituir a atual controladora eletrônica por uma nova, confeccionada sob a plataforma de prototipagem arduino. Esta deveria ser capaz de comunicar-se através da rede sem fios ZigBee com um servidor central, responsável pela coordenação dos protótipos e disponibilização de uma interface Web para o controle dos aparelhos de ar condicionado pelos usuários de forma prática.

A princípio desenvolveu-se o código para coleta dos dados do sensor de temperatura e acionamento do compressor, motor externo e das velocidades do ventilador via arduino. Após concluída esta parte, partimos então para a integração do arduino com os módulos XBee.

Para verificar a comunicação entre os módulos XBee, inicialmente realizamos testes de comunicação no modo AT (transparente), que ocorreram conforme o esperado. Pela limitação deste modo, devido a não possibilidade de endereçamento de mensagens especificas a cada módulo, partiu-se então para a implementação dos testes no modo API. Realizados testes de endereçamento e roteamento de pacotes através de outros módulos no modo API e estes comprovaram a eficácia da comunicação via rede sem fio ZigBee neste modo de comunicação.

Após esta etapa deu-se inicio ao desenvolvimento do código em Java para o Servidor de Aplicação dos protótipos, tendo como ponto de partida o código desenvolvido no projeto Droidlar (EUZEBIO, 2011). Visando projetos futuros, tentou-se deixar o sistema aberto a possibilidade de novos tipos de dispositivos e novas funções para o próprio aparelho de ar condicionado, sem a necessidade de muitas alterações no código para tal. Desenvolveu-se ainda um código em *HyperText Markup Language* (HTML) para criar a interface de controle dos aparelhos de ar condicionado pelos usuários.

Também foi projetado um circuito para o acionamento das velocidades do ventilador, compressor e motor externo dos aparelhos de ar condicionado. Os testes de funcionamento do acionamento das velocidades do ventilador foram feitos com um motor com principio de acio-

namento igual ao modelo utilizado pelo aparelho escolhido para o projeto. O teste de acionamento do compressor e do motor externo, levando em conta que só é necessário conectar o fase para ligá-los, foi simulado com uma lâmpada conectada há um relé acionado por uma porta do arduino, conforme o circuito já mostrado na Figura 4.1.

Após finalizados todos os testes de acionamento foi constatado que a solução proposta funciona mas pode ser aperfeiçoada. Algumas ideias para a melhoria do projeto são introduzidas a seguir.

6.1 Trabalhos futuros

A partir do estudo realizado neste trabalho uma densa gama de outras possibilidades podem ser exploradas. O projeto facilita a derivação em outras ideias devido ao uso de código aberto tanto na parte de hardware quanto de software.

Um trabalho futuro seria incrementar o sistema atual com outras funções presentes nos aparelhos de ar condicionado como: controle de umidade, posição das pás que direcionam o ar, controle de congelamento da serpentina e função quente/frio. Seria interessante também criar rotinas para ligar e desligar os aparelhos baseado em horários pré-determinados e controlar a temperatura automaticamente de acordo com a presença de pessoas no recinto. Expandir a quantidade de aparelhos de ar condicionado suportados pelo sistema também seria bom, levando em conta as especificações de funcionamento de cada aparelho.

Outra melhoria seria a inserção de outros tipos de dispositivos como lâmpadas, sistemas de alarme, portas e portões eletrônicos. Um sistema de controle de acesso, para que apenas usuários selecionados possam controlar os dispositivos conectados poderia ser desenvolvido também. O Servidor de Aplicação poderia ainda de embarcado em um dispositivo micro controlado, o que traria economia e mobilidade ao sistema. Poderia ainda ser expandida a quantidade de dispositivos controlados simultaneamente por cada arduino, atualmente cada arduino controla apenas um aparelho, visando diminuir os custos.

APÊNDICE A – Configuração dos módulos XBee dos protótipos

Por ser um software desenvolvido para Windows, foi necessário utilizar o software Wine[1] para sua execução no linux. O Wine é um software que implementa a API do Microsoft Windows, permitindo a execução de aplicativos Windows nativamente em outras plataformas.

Para o Wine conseguir acessar com a porta USB no linux é necessário criar um link simbólico da porta USB do linux para o Wine conseguir se comunicar com a mesma. O comando para criar este link simbólico depende do nome que o linux vai colocar na porta USB e do nome desejado no Wine. O linux cria um arquivo para cada porta USB com um dispositivo conectado em /dev com o nome de ttyUSBx, onde o 'x' é substituído por um número de acordo com as portas USB com dispositivos conectados. O comando utilizado para criar o link simbólico no caso de a porta se chamar ttyUSB1 e o nome escolhido para acesso via Wine seja COM1 seria:

ln -s /dev/ttyUSB1 ~/.wine/dosdevices/COM1

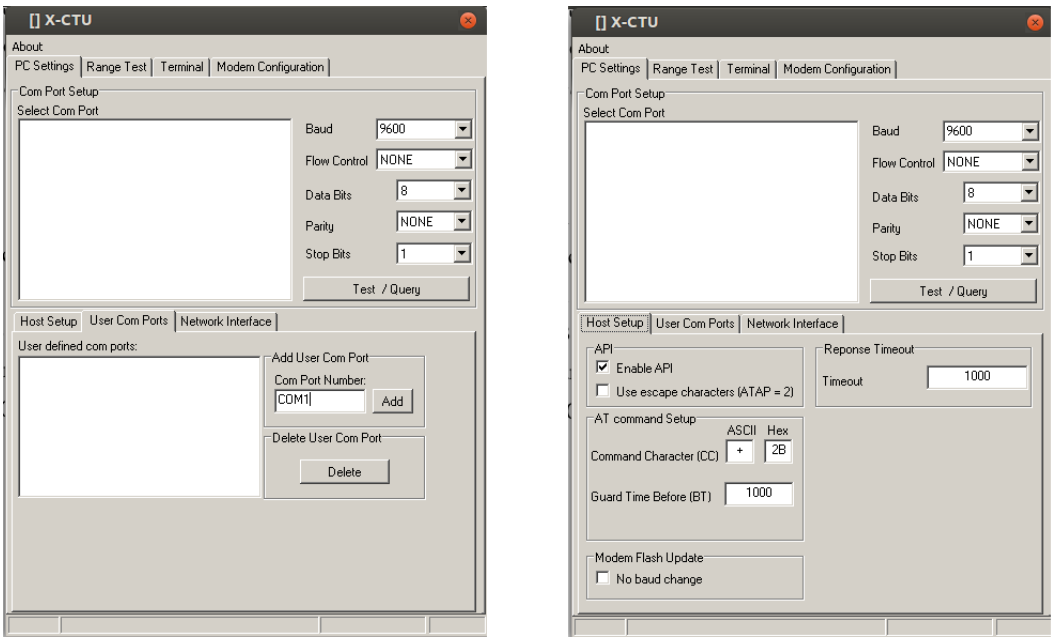
Também foi necessário configurar as permissões de leitura e escrita da porta USB através do comando:

chmod a+rw /dev/ttyUSB1

No X-CTU para configuração dos módulos foi necessário adicionar a porta para conexão com o módulo XBee, abrindo a aba inferior *User Com Ports*, digitar o nome a porta de acordo com o nome simbólico dado anteriormente (COM1) em *Com Number Port* e clicar em *Add* em *Add User Com Port* (Figura A.1(a)).

Pelo fato de trabalharmos com os módulos XBee no modo de operação API, foi necessário habilitá-lo na aba inferior *Host Setup*, em API marcar *Enable API* (Figura A.1(b)). Através do botão *Test/Query* foi possível testar a comunicação com o módulo e obter o modelo de XBee usado.

[1] http://www.winehq.org/



(a) Adicionando porta para conexão com o módulo.

(b) Habilitando modo API.

Figura A.1: Configuração do X-CTU.

Com o modelo foi possível clicar na aba superior *Modem Configuration* e selecionar o modelo (no caso XB24-B). Os protótipos atuam como roteadores, portanto foi selecionado a sua função como *ZNET 2.5 ROUTER/END DEVICE API* (Figura A.2).

A maior parte das configurações foi deixada como padrão, as configurações alteradas foram:

- ***Node Identifier***, dentro da categoria *Addressing*, configurada de acordo com o nome desejado para o nó, ex. ROTEADOR1.
- ***API Enable***, dentro da categoria *Serial Interfacing*, configurada com o modo API usado, no caso 2.

Após o término da configuração, foi necessário apenas escrever as novas configurações no módulo e esperar a mensagem de escrita com sucesso.

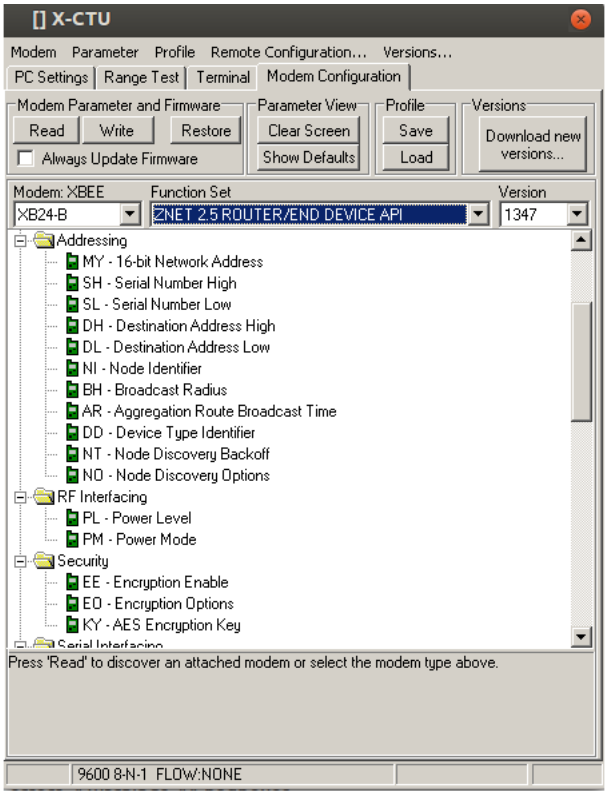


Figura A.2: Módulo de configuração dos protótipos - X-CTU.

APÊNDICE B – Esquemas elétricos do aparelho de ar condicionado

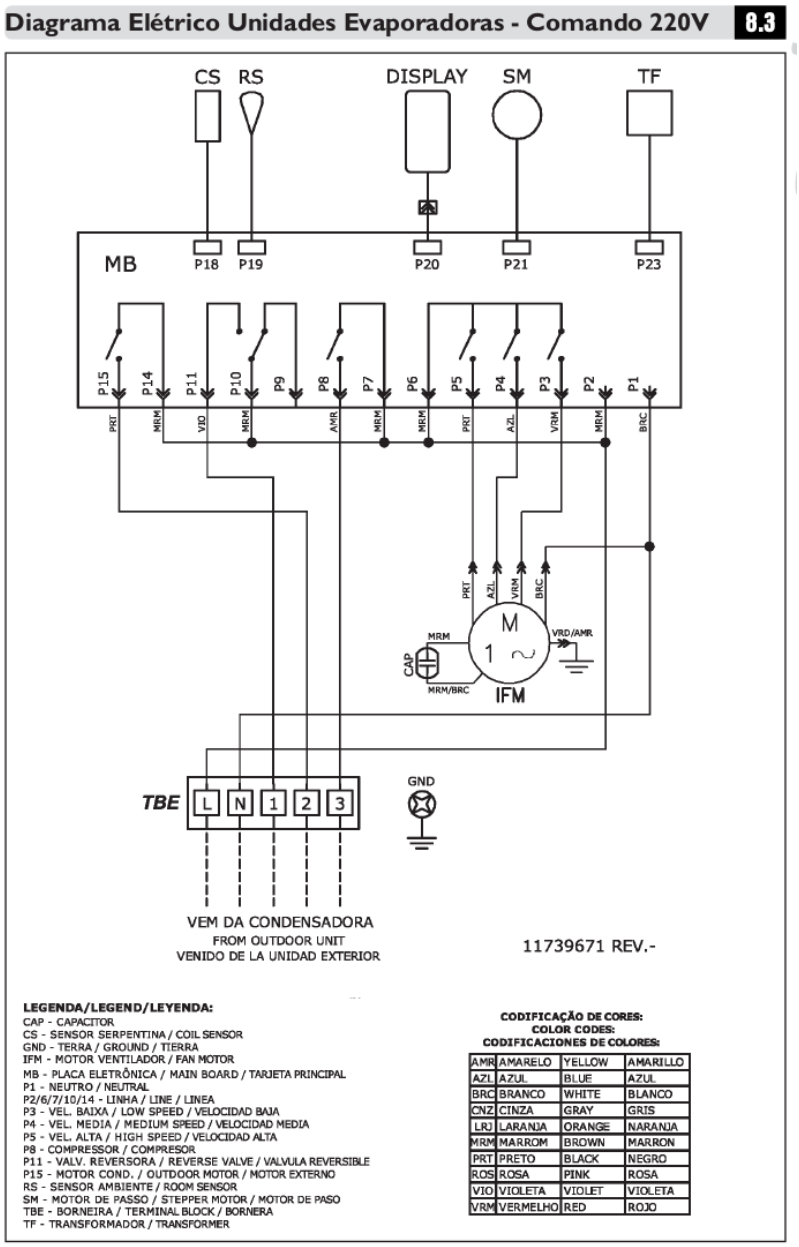


Figura B.1: Esquema elétrico da condensadora.

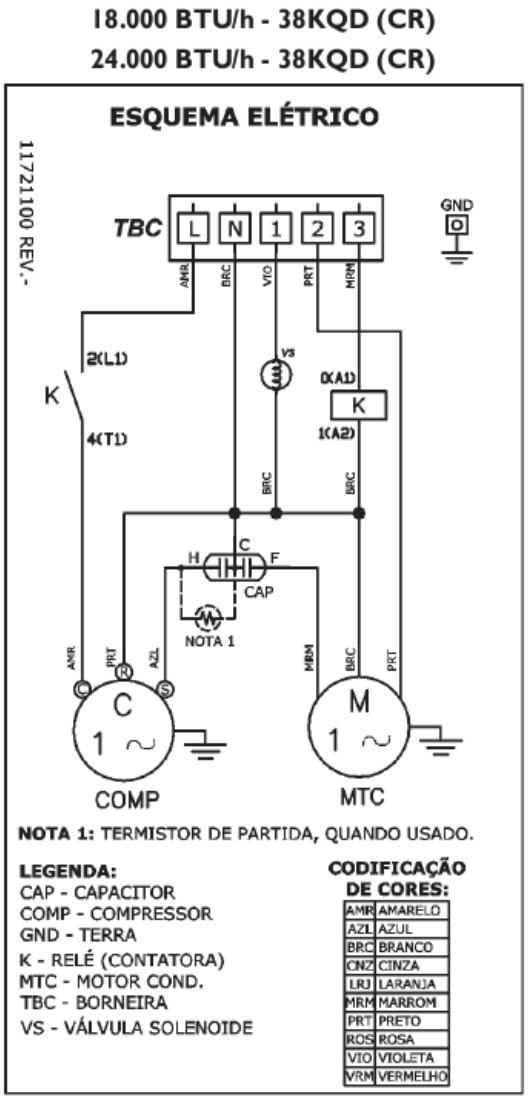


Figura B.2: Esquema elétrico da evaporadora.

APÊNDICE C – Componentes utilizados na montagem

Switching diode

1N4148 / 1N4150 / 1N4448 / 1N914B

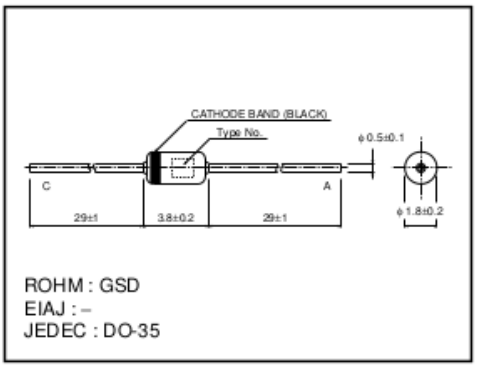
∗This product is available only outside of Japan.

●Applications
High-speed switching

●Features
1) Glass sealed envelope. (GSD)
2) High speed.
3) High reliability.

●Construction
Silicon epitaxial planar

●External dimensions (Units : mm)



ROHM : GSD
EIAJ : –
JEDEC : DO-35

●Absolute maximum ratings (Ta = 25°C)

Type	V_{RM} (V)	V_R (V)	I_{FM} (mA)	I_O (mA)	I_F (mA)	I_{FSM} 1μs (A)	P (mW)	Tj (°C)	Topr (°C)	Tstg (°C)
1N4148	100	75	450	150	200	2	500	200	–65~+200	–65~+200
1N4150	50	50	600	200	250	4	500	200	–65~+200	–65~+200
1N4448 (1N914B)	100	75	450	150	200	2	500	200	–65~+200	–65~+200

●Electrical characteristics (Ta = 25°C)

Type	V_F (V) @0.1mA	@0.25mA	@1mA	@2mA	@5mA	@10mA	@20mA	@30mA	@50mA	@100mA	@200mA	@250mA	BV (V) Min. @5μA	@100μA	I_R (μA) Max. @25°C	V_R (V)	@150°C	V_R (V)	C_t (pF) V_R=0 f=1MHz	t_{rr} (ns) V_R=6V I_F=10mA R_L=100Ω
1N4148						1.0							75	100	0.025	20	50.0	20	4	4
															5.0	75				
1N4150			0.54			0.66			0.76	0.82	0.87		–	50	0.1	50	100.0	50	2.5	4
			0.62			0.74			0.86	0.92	1.0									
1N4448 (1N914B)					0.62								–	100	0.025	20	50.0	20	4	4
					0.72					1.0					5.0	75				

The upper figure is the minimum V_F and the lower figure is the maximum V_F value.

Figura C.1: Diodo 1N4148.

●Electrical characteristic curves (Ta = 25°C)

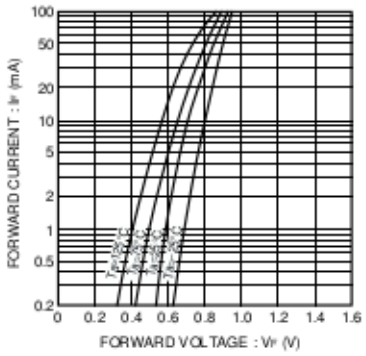


Fig. 1 Forward characteristics

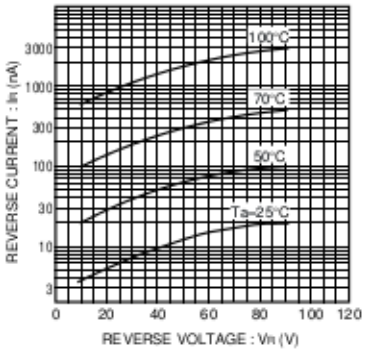


Fig. 2 Reverse characteristics

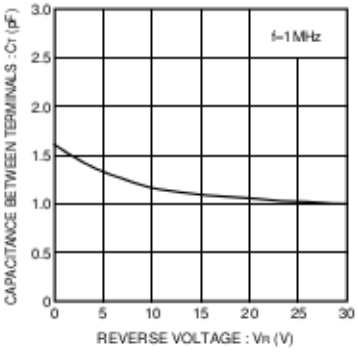


Fig. 3 Capacitance between terminals characteristics

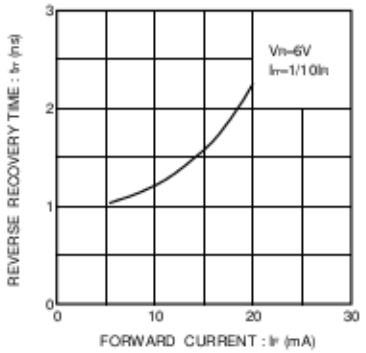


Fig. 4 Reverse recovery time characteristics

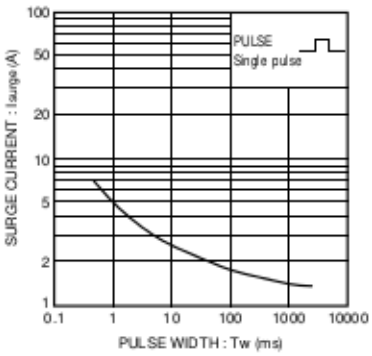


Fig. 5 Surge current characteristics

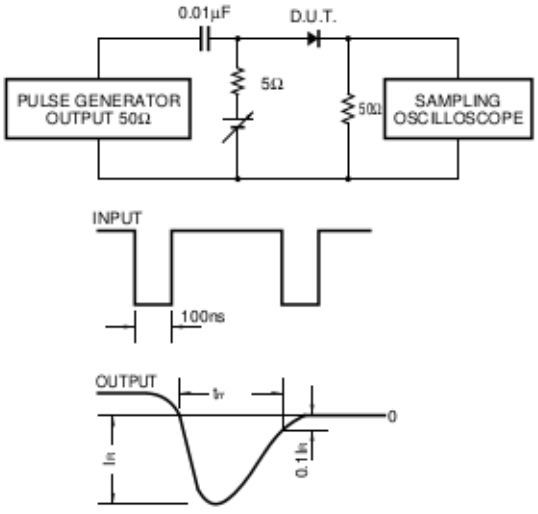


Fig. 6 Reverse recovery time (t_{rr}) measurement circuit

Figura C.2: Diodo 1N4148 - cont.

PCE series

10 Amp Miniature Power PC Board Relay

Appliances, HVAC, Office Machines

UL File No. E82292
CSA File No. LR48471
VDE File No. 6175

Users should thoroughly review the technical data before selecting a product part number. It is recommended that user also seek out the pertinent approvals files of the agencies/laboratories and review them to ensure the product meets the requirements for a given application.

Features

- Small, low profile package, 10 Amp switching capacity.
- 1 Form A and 1 Form C contact arrangements.
- UL Class F (155°C) insulation system standard
- Immersion cleanable, sealed version available.
- Applications include appliance, HVAC, security system, garage opener control, emergency lighting.

Contact Data @ 20°C

Arrangements: 1 Form A (SPST-NO) and 1 Form C (SPDT).
Material: Ag Alloy, AgSnO.
Max. Switching Rate: 300 ops./min. (no load).
30 ops./min. (rated load).
Expected Mechanical Life: 10 million operations (no load).
Expected Electrical Life: 100,000 operations (rated load).
Minimum Load: 100mA @ 5VDC.
Initial Contact Resistance: 100 milliohms @ 1A, 6VDC.

Contact Ratings

Ratings: 10A @ 250VAC resistive,
10A @ 120VAC resistive,
10A @ 28VDC resistive.

3A @ 250VAC inductive (cosø= 0.4),
3A @ 120VAC inductive (cosø= 0.4),
3A @ 28VDC inductive (L/R=7msec).

Max. Switched Voltage: AC: 250V.
DC: 28V.
Max. Switched Current: 10A.
Max. Switched Power: 2,500VA, 280W.

Initial Dielectric Strength

Between Open Contacts: 750VAC 50/60 Hz. (1 minute).
Between Coil and Contacts: 2,000VAC 50/60 Hz. (1 minute).
Surge Voltage Between Coil and Contacts: 4,000V (1.2 / 50µs).

Initial Insulation Resistance

Between Mutually Insulated Elements: 1,000M ohms min. @ 500VDC.M.

Coil Data

Voltage: 6 to 48VDC.
Nominal Power: 360 mW.
Coil Temperature Rise: 35°C max., at rated coil voltage.
Max. Coil Power: 130% of nominal.
Duty Cycle: Continuous.

Coil Data @ 20°C

PCE				
Rated Coil Voltage (VDC)	**Nominal Current (mA)**	**Coil Resistance (ohms) ± 10%**	**Must Operate Voltage (VDC)**	**Must Release Voltage (VDC)**
6	60	100	4.50	0.30
9	40	225	6.75	0.45
12	30	400	9.00	0.60
24	15	1,600	18.00	1.20
48	7	6,400	36.00	2.40

Operate Data

Must Operate Voltage: 75% of nominal voltage or less.
Must Release Voltage: 5% of nominal voltage or more.
Operate Time: 10 ms max.
Release Time: 5 ms max.

Environmental Data

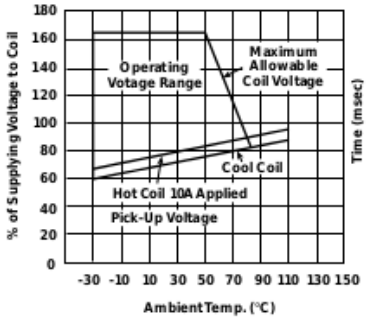
Temperature Range:
Operating: -30°C to +70°C
Vibration, Mechanical: 10 to 55 Hz., 1.5mm double amplitude
Operational: 10 to 55 Hz., 1.5mm double amplitude.
Shock, Mechanical: 1,000m/s² (100G approximately).
Operational: 100m/s² (10G approximately).
Operating Humidity: 20 to 85% RH. (Non-condensing).

Mechanical Data

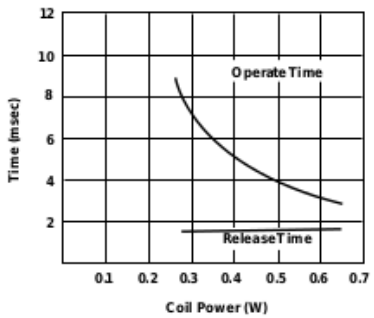
Termination: Printed circuit terminals.
Enclosure (94V-0 Flammability Ratings):
PCE: Sealed plastic case with knock-off nib for ventilation
Weight: 0.32 oz (11g) approximately.

Reference Data

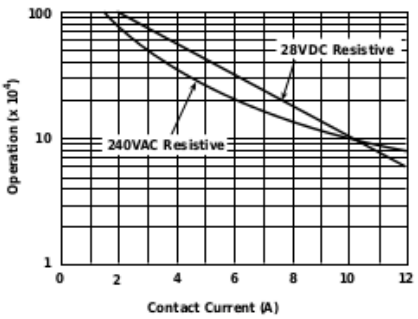
Coil Temperature Rise



Operate Time



Life Expectancy



Note: This data is based on the max. allowable temperature for E type insulation coil (115°C).

Figura C.3: Relé PCE-106D1H.

Ordering Information

Typical Part Number ▶	PCE	-1	24	D	1	M		,000

1. Basic Series:
PCE = Miniature Power PC board relay.

2. Termination:
1 = 1 pole

3. Coil Voltage:
06 = 6VDC 12 = 12VDC 48 = 48VDC
09 = 9VDC 24 = 24VDC

4. Coil Input:
D = Standard

5. Contact Material:
1 = AgCdO 2 = AgSnO

6. Contact Arrangement:
Blank = 1 Form C, SPDT M = 1 Form A, SPST-NO

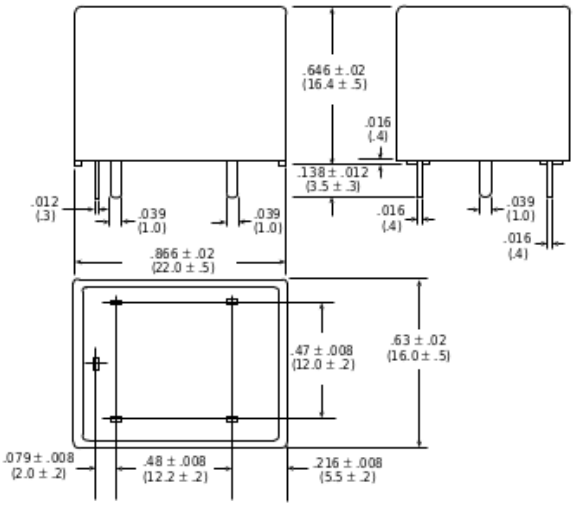
7. Enclosure:
Blank = Flux-tight plastic case. H = Sealed plastic case with knock-off nib for ventilation

8. Suffix:
,000 = Standard model Other Suffix = Custom model

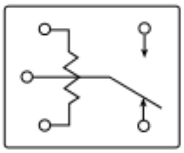
Our authorized distributors are more likely to maintain the following items in stock for immediate delivery.

PCE-112D1MH,000 PCE-112D1H,000
PCE-124D1MH,000 PCE-124D1H,000

Outline Dimensions



Wiring Diagram (Bottom View)



PC Board Layout (Bottom View)

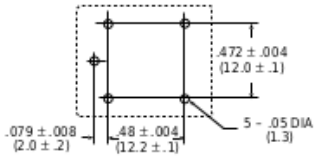
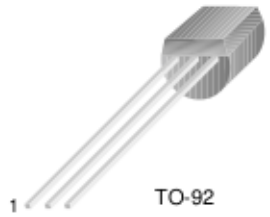


Figura C.4: Relé PCE-106D1H - cont.

BC546/547/548/549/550

Switching and Applications

- High Voltage: BC546, V_{CEO}=65V
- Low Noise: BC549, BC550
- Complement to BC556 ... BC560



TO-92

1. Collector 2. Base 3. Emitter

NPN Epitaxial Silicon Transistor

Absolute Maximum Ratings T_a=25°C unless otherwise noted

Symbol	Parameter	Value	Units
V_{CBO}	Collector-Base Voltage : BC546	80	V
	: BC547/550	50	V
	: BC548/549	30	V
V_{CEO}	Collector-Emitter Voltage : BC546	65	V
	: BC547/550	45	V
	: BC548/549	30	V
V_{EBO}	Emitter-Base Voltage : BC546/547	6	V
	: BC548/549/550	5	V
I_C	Collector Current (DC)	100	mA
P_C	Collector Power Dissipation	500	mW
T_J	Junction Temperature	150	°C
T_{STG}	Storage Temperature	-65 ~ 150	°C

Electrical Characteristics T_a=25°C unless otherwise noted

Symbol	Parameter	Test Condition	Min.	Typ.	Max.	Units
I_{CBO}	Collector Cut-off Current	V_{CB}=30V, I_E=0			15	nA
h_{FE}	DC Current Gain	V_{CE}=5V, I_C=2mA	110		800	
V_{CE} (sat)	Collector-Emitter Saturation Voltage	I_C=10mA, I_B=0.5mA		90	250	mV
		I_C=100mA, I_B=5mA		200	600	mV
V_{BE} (sat)	Base-Emitter Saturation Voltage	I_C=10mA, I_B=0.5mA		700		mV
		I_C=100mA, I_B=5mA		900		mV
V_{BE} (on)	Base-Emitter On Voltage	V_{CE}=5V, I_C=2mA	580	660	700	mV
		V_{CE}=5V, I_C=10mA			720	mV
f_T	Current Gain Bandwidth Product	V_{CE}=5V, I_C=10mA, f=100MHz		300		MHz
C_{ob}	Output Capacitance	V_{CB}=10V, I_E=0, f=1MHz		3.5	6	pF
C_{ib}	Input Capacitance	V_{EB}=0.5V, I_C=0, f=1MHz		9		pF
NF	Noise Figure : BC546/547/548	V_{CE}=5V, I_C=200μA		2	10	dB
	: BC549/550	f=1KHz, R_G=2KΩ		1.2	4	dB
	: BC549	V_{CE}=5V, I_C=200μA		1.4	4	dB
	: BC550	R_G=2KΩ, f=30~15000MHz		1.4	3	dB

h_{FE} Classification

Classification	A	B	C
h_{FE}	110 ~ 220	200 ~ 450	420 ~ 800

Figura C.5: Transistor BC547B.

Typical Characteristics

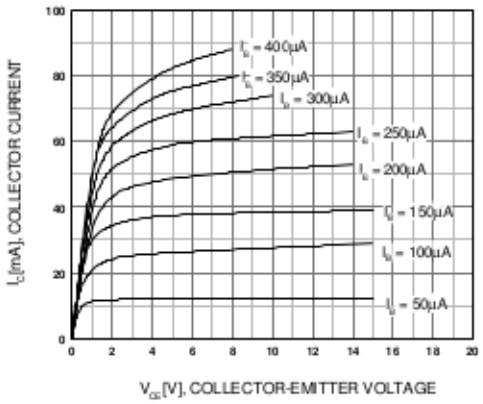


Figure 1. Static Characteristic

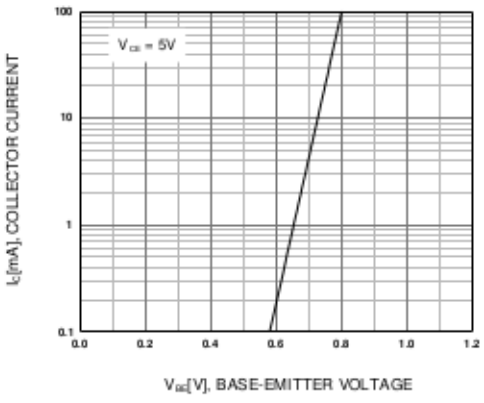


Figure 2. Transfer Characteristic

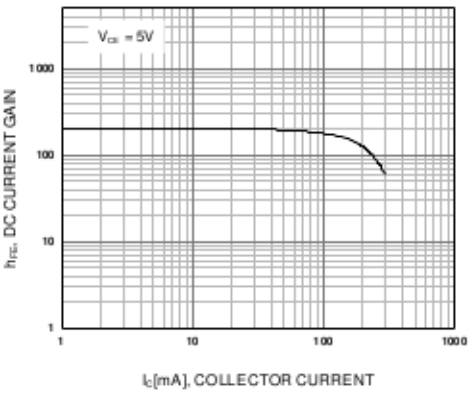


Figure 3. DC current Gain

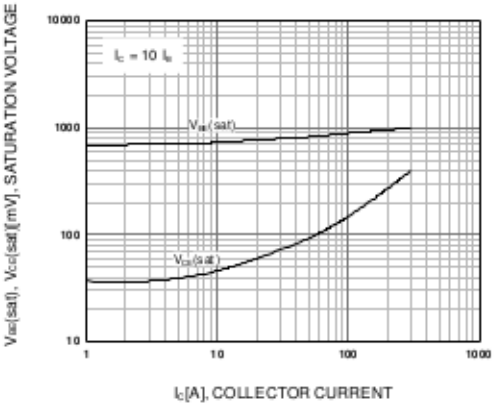


Figure 4. Base-Emitter Saturation Voltage Collector-Emitter Saturation Voltage

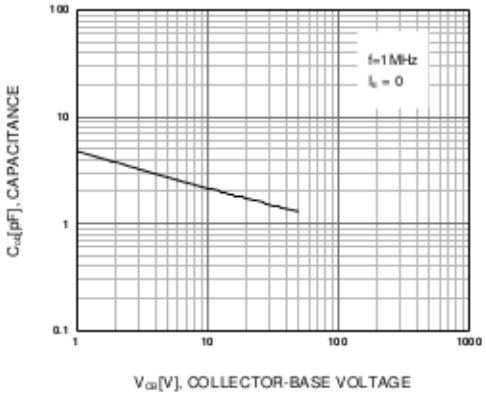


Figure 5. Output Capacitance

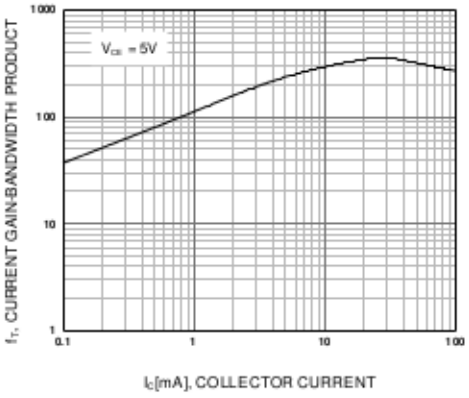


Figure 6. Current Gain Bandwidth Product

Figura C.6: Transistor BC547B - cont.

Lista de Abreviaturas

AES *Advanced Encryption Standard*

AODV *Ad Hoc On-Demand Distance Vector*

API *Application Programming Interface*

CAP *Contention Access Period*

CFP *Contetion-Free Period*

CTR *Counter*

CSMA/CA *Carrier Sense Multiple Access With Collision Avoidance*

FFD *Full-Function Device*

GTS *Guaranteed Time Slot*

HTML *HyperText Markup Language*

IDE *Integrated Development Environment*

IEEE *Institute of Electrical and Electronics Engineers*

IFS *Interframe Space or Spacing*

ISM *Industrial, Scientific and Medical*

LR-WPAN *Low-Rate Wireless Personal Area Network*

MAC *Medium Access Control*

P2P *Peer-to-peer*

PAN *Personal Area Network*

PHP *PHP: Hypertext Preprocessor*

PHY *Physical Layer*

PWM *Pulse Width Modulation*

RFD *Reduced-Function Device*

WPAN *Wireless Personal Area Network*

ZDO *ZigBee Device Object*

Referências Bibliográficas

DIGI INTERNATIONAL. Xbee® zb zigbee® modules - digi international. In: ____. 2012. Disponível em: <http://www.digi.com/products/wireless-wired-embedded-solutions/zigbee-rf-modules/zigbee-mesh-module/xbee-zb-module#overview>. Acesso em: 22 jun. 2012.

EUZEBIO, M. V. de M. *DroidLar - Automação residencial através de um celular Android*. [s.n.], 2011. Disponível em: <http://wiki.sj.ifsc.edu.br/wiki/images/b/b7/TCC_MichelEusebioMello.pdf>. Acesso em: 03 mar. 2013.

FALUDI, R. In: ____. *Building Wireless Sensor Networks*. [S.l.: s.n.], 2011.

FRIAS, R. N. *ZigBee: Camadas*. Teleco, 2012. Disponível em: <http://www.teleco.com.br/tutoriais/tutorialzigbee/pagina_3.asp>. Acesso em: 26 jun. 2012.

GISLASON, D. In: ____. *Zigbee Wireless Network and Transceivers*. [S.l.: s.n.], 2008.

IEEE, C. S. *IEEE Standard for Information technology - Telecommunications and information exchange between systems - Local and metropolitan area networks - Specific requirements: Part 15.4: Wireless Medium Access Control(MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (WPANs), Revision of IEEE Std 802.15.4-2003*. IEEE, 2006. Disponível em: <http://standards.ieee.org/getieee802/download/802.15.4-2006.pdf>. Acesso em: 7 jun. 2012.

JUNIOR., L. C. M. *Refrigeração e Ar-Condicionado - Parte I: Histórico, Ciclo de Compressão, Diagramas de Mollier, Refrigerantes*. [s.n.], 2003. Disponível em: <http://wiki.sj.ifsc.edu.br/wiki/images/b/1b/RAC_I.pdf>. Acesso em: 28 fev. 2013.

JUNIOR., L. C. M. *Refrigeração e Ar-Condicionado - Parte II: Ciclo de compressão, Balanço de Energia, Trocadores de Calor, Dispositivos de Expansão*. [s.n.], 2003. Disponível em: <http://wiki.sj.ifsc.edu.br/wiki/images/b/bb/RAC_II.pdf>. Acesso em: 28 fev. 2013.

KINNEY, P. *ZigBee Technology: Wireless Control that Simply Works*. ZigBee Alliance, 2003. Disponível em: <http://www.zigbee.org/imwp/idms/popups/pop_download.asp?contentID=5162>. Acesso em: 4 jun. 2012.

MELLIS, D. A. *Arduino - Blink*. Arduino, 2009. Disponível em: <http://arduino.cc/en/Tutorial/Blink>. Acesso em: 26 jun. 2012.

MELLIS, D. A. *Arduino - Introduction*. Arduino, 2009. Disponível em: <http://www.arduino.cc/en/Guide/Introduction>. Acesso em: 11 jun. 2012.

MELLIS, D. A. et al. *Arduino: An Open Electronics Prototyping Platform*. Arduino, 2007. Disponível em: <http://web.media.mit.edu/ mellis/arduino-chi2007-mellis-banzi-cuartielles-igoe.pdf>. Acesso em: 20 jun. 2012.

PERKINS, C.; BELDING-ROYER, E.; DAS, S. *Ad hoc On-Demand Distance Vector (AODV) Routing*. IETF, 2003. Disponível em: <https://tools.ietf.org/html/rfc3561>. Acesso em: 26 jan. 2013.

PINHEIRO, J. M. S. *Falando de Automação Predial*. [s.n.], 2004. Disponível em: <http://www.projetoderedes.com.br/artigos/artigo_falando_de_automacao_predial.php>. Acesso em: 14 mar. 2013.

PINHEIRO, J. M. S. *Sistemas de Automação*. [s.n.], 2004. Disponível em: <http://www.projetoderedes.com.br/artigos/artigo_sistemas_automacao.php>. Acesso em: 14 mar. 2013.

SPRINGER. *Manual de Intalação, Operação e Manutenção*. 2011. Disponível em: <http://www.springer.com.br/Download/300-2-2>. Acesso em: 23 fev. 2013.

ZIGBEE ALLIANCE. *ZigBee Specification, Document 053474r17*. ZigBee Alliance, 2008. Disponível em: <http://www.zigbee.org/Standards/ZigBeeSmartEnergy/Specification.aspx>. Acesso em: 4 jun. 2012.