CENTRO UNIVERSITÁRIO DE BRASÍLIA - UniCEUB

CURSO DE ENGENHARIA DE COMPUTAÇÃO

Marco Aurélio de Souza

Sistema de Automação Residencial para Iluminação

Orientador: José Julimá Bezerra Júnior

Brasília

Dezembro, 2010

Marco Aurélio de Souza

Sistema de Automação Residencial para Iluminação

Trabalho apresentado ao Centro Universitário de Brasília (UniCEUB) como pré-requisito para a obtenção de Certificado de Conclusão de Curso de Engenharia de Computação.

Orientador: Prof. José Julimá Bezerra Júnior

Brasília

Dezembro, 2010

Este Trabalho foi julgado adequado para a obtenção do Título de Engenheiro de Computação, e aprovado em sua forma final pela Faculdade de Tecnologia e Ciências Sociais Aplicadas - FATECS.

_____

Prof. Abiezer Amarilia Fernandez

Coordenador do Curso

Banca Examinadora:

_____

Prof. José Julimá Bezerra - Mestre em Engenharia Elétrica.

Instituto Militar de Engenharia - RJ

Orientador

_____

Prof. João Marcos Souza Costa. Mestre em Matemática

_____

Prof. Giu Renato Ribeiro Gonçalves. Doutor em Física

_____

Prof. Antônio Barbosa Júnior. Especialista em Tecnologia da Informação

Dedico este trabalho ao meu pai Marcos Pessoa, à minha mãe Maria Farias e ao meu irmão Marco Antônio, por sempre me ajudarem na vida, me ensinando e guiando sempre para os melhores caminhos.

AGRADECIMENTOS

Agradeço primeiro aos meus pais por terem me dado toda educação que tenho hoje. Aos meus amigos por sempre me acompanharem. Ao meu irmão, Marco Antônio, que me ajudou bastante e me ajudou no desenvolvimento do projeto. Ao meu professor orientador José Julimá Bezerra Júnior. E a minha namorada, Luciana Fraga.

RESUMO

Este projeto apresenta uma proposta de um sistema de automação residencial para iluminação utilizando a tecnologia ZigBee. Para isso, um protótipo foi construído com o objetivo de simular o acionamento de uma lâmpada à distância. Os dispositivos CON-USBBEE e o RCON-HOMEBEE são os principais componentes do protótipo. A tecnologia ZigBee é utilizada para a comunicação entre essas duas placas, via radio freqüência. O CON-USBBEE enviará através de um software de gerenciamento um comando para a placa RCON-HOMEBE, para acionar as lâmpadas. O computador é responsável pelo controle e monitoramento do sistema.

Palavras Chave: comunicação por rádio freqüência, ZigBee.

ABSTRACT

This project presents a proposal for a home automation system for lighting using ZigBee technology. For this, a prototype was built to simulate the firing of a lamp from a distance. The devices CON- USBBEE and RCON-HOMEBEE are the main components of the prototype. The ZigBee technology is used for communication between these two boards, via radio frequency. The CON-USBBEE send through the software a command to a management board RCON-HOMEBE, to power the lamps. The computer is responsible for controlling and monitoring system.

Key-words: Radio frequency communication, ZigBee.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE ABREVIATURAS E SIGLAS

IEEE - Instituto de Engenheiros Eletricistas e Eletrônicos

WPAN - Wireless Personal Area Network

RF – Rádio Freqüência

RFD - Reduced Function Device

FFD - Full Function Device

PHY – Camada física

MAC – Media Access Control

ZC – ZigBee Coordenator

ZR – ZigBee Router

ZED – ZigBee End Device

USB - Universal Serial Bus

RSSI - received signal strength indicator

CAPÍTULO 1 - INTRODUÇÃO

1.1 – Motivação

O principal fator motivacional para a realização desde projeto foi o crescente aumento no consumo de energia do país. No Brasil, o consumo de energia tem aumentado visivelmente, tendo uma pequena diminuição quando são tomadas algumas atitudes forçadas, como o racionamento de energia. Em 2008, um estudo realizado pelo IBGE, mostra que, entre 1995 e 2006, o consumo de energia cresceu em 37,37%. Esse número foi superior ao crescimento do PIB, 31,46%. Um dos motivos para esse grande consumo de energia pode ser a falha humana, que sempre esquecem uma luz acesa durante longo tempo. *(Estadão, 2008)*

Tendo em vista esses dados e a consciência do enorme consumo no país, e com intuito de reduzir esse elevado consumo de energia, esse projeto tem como objetivo, através de um protótipo, criar um sistema de automação residencial para iluminação. Nesse sistema, o usuário poderá acender e apagar lâmpadas de sua residência pelo computador, com um simples "clique". Além disso, o usuário terá um arquivo com hora e data de todas as ações executadas no sistema.

1.2 – Visão Geral

A Figura 1.1 ilustra o posicionamento dos dispositivos eletrônicos que serão controlados por um sistema de gerenciamento, sendo esquematizada a comunicação entre eles.

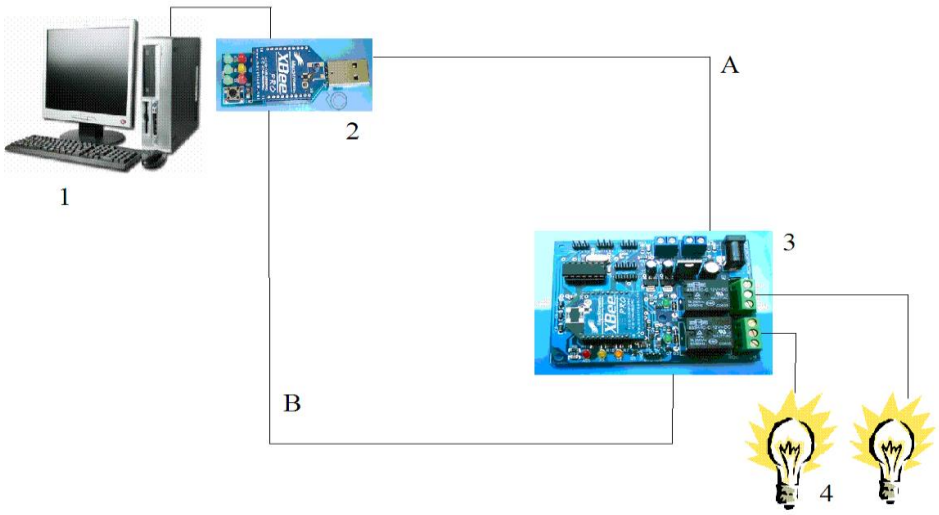


Figura 1.1 – Posicionamento e comunicação de dispositivos

Na Figura 1.1 pode-se ver a placa COON-USBBEE (2) ligada ao computador (1). Ele enviará o comando para a placa RCON-HOMEBEE (3), via ZigBee (A). Com o comando as lâmpadas (4) serão acionadas para ligar/desligarem. E depois será enviada uma confirmação(B) de que a ação foi executada.

1.3 – Objetivos do Trabalho

A finalidade do trabalho é apresentar um sistema de automação residencial para iluminação utilizando tecnologia de comunicação ZigBee. O sistema trabalhará com a função de ligar e desligar remotamente lâmpadas de uma residência. Para isso será executada algumas etapas, como:

A) Conectar as partes físicas do Sistema;
   - Computador;
   - Placa CON-USBBEE;
   - Placa RCON-HOMEBEE;
   - 2 Lâmpadas.
B) Criar um software para se comunicar com os hardwares;
C) Criar uma interface moderna, porém simples para o Sistema.
   - Será implementado em C#(CSharp).
D) Executar testes no projeto.
   - Fazer testes para garantir total êxito na apresentação.

1.3 – Estruturas da Monografia

Além deste capítulo introdutório, esta monografia está estruturada em mais três capítulos e organizada da seguinte maneira:

No capítulo 2 é apresentado o referencial teórico e tecnológico, que embasa o projeto. Primeiramente trata de detalhes sobre automação residencial. Em seguida apresenta detalhes sobre a tecnologia ZigBee utilizada no projeto.

O capítulo 3, capítulo do desenvolvimento do projeto, possui a visão e a topologia do projeto. Além de especificar os hardwares e softwares utilizados no protótipo. Esse capítulo também mostra os testes realizados e a simulação do projeto.

O capítulo 4, conclusão, marca o final da monografia concluindo-a e apresentando propostas para futuros trabalhos.

CAPÍTULO 2 – REFERENCIAL TEÓRICO E TECNOLÓGICO

2.1 - Comunicação ZigBee

2.1.1 - Introdução

Atualmente o foco das redes wireless comerciais se encontra no contexto das redes locais (WLAN's), tanto em soluções proprietárias como nos padrões desenvolvidos pelo IEEE, por exemplo. Com a evolução natural das tecnologias das redes sem fio, estas passaram a atender não só as aplicações corporativas mais sofisticadas como também aquelas envolvendo pequenos volumes de dados que exigem baixas taxas de transmissão como, por exemplo, o controle de equipamentos eletroeletrônicos. Além disso, outras tecnologias sem fio têm sido utilizadas também com o objetivo de proporcionar a comunicação pessoal e o controle de dispositivos diversos, são as chamadas redes pessoais (WPAN's). *(Passarela, 2006)*

Basicamente, essas tecnologias têm o propósito de permitir o controle remoto de equipamentos domésticos (TV's, videocassetes, geladeiras, etc) e periféricos (teclados, mouse, impressoras, etc), eliminando os cabos e tornando mais prática a operação desses equipamentos pelos usuários.

Uma das tecnologias mais recentes dentro desse grupo de redes para aplicações pessoais e que permite o gerenciamento e controle desses dispositivos é o padrão ZigBee, também conhecido como HomeRF Lite e que corresponde ao IEEE 802.15.4, homologado em maio de 2003. A Figura 2.1 apresenta algumas aplicações do padrão ZigBee, em automação residencial.

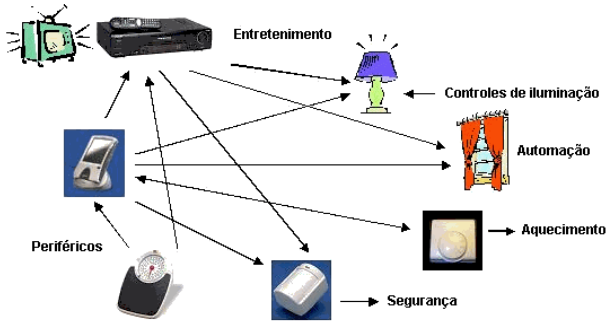


Figura 2.1 - Aplicações do ZigBee

Fonte: (Pinheiro, 2006)

O padrão ZigBee foi desenvolvido para se tornar uma alternativa de comunicação em redes que não necessitem de soluções mais complexas para seu controle, barateando assim os custos com a aquisição, instalação de equipamentos, manutenção e mão de obra. Trata-se de uma tecnologia relativamente simples, que utiliza um protocolo de pacotes de dados com características específicas, sendo projetado para oferecer flexibilidade quanto aos tipos de dispositivos que pode controlar.

2.1.2 - Comparação com outras tecnologias

O Quadro 2.1 relaciona alguns dados das tecnologias sem fio (wireless), Bluetooth e ZigBee onde é possível observar como definir as melhores aplicações para cada tipo de dispositivo sem fio. O padrão ZigBee está como líder absoluto na transmissão de textos.

Quadro 2.1 – Comparação com outras tecnologias

Característica	IEEE 802.11B	Bluetooth	ZigBee
Bateria	Horas	Dias	Anos
Complexidade	Muito complexo	Complexo	Simples
Dispositivos	32	7	64000
Distância	100 metros	10 metros	70-300 metros
Taxa Tranferência	11 Mbps	1 Mbps	250 Kbps

Fonte: *(*Pinheiro, 2006)

A denominação ZigBee é mais comummente utilizada para controles remoto, produtos a bateria e sensores em geral. Já as tecnologias bluetooth e WI-FI são utilizadas para porta USB sem fio e fone de ouvido e transferência de arquivos, redes de computadores e navegação à internet respectivamente.

2.1.3 - ZigBee Alliance

A ZigBee Alliance, que desenvolve o padrão ZigBee junto ao IEEE, é uma associação que conta com mais de 45 empresas, que trabalham em conjunto para desenvolver um padrão capaz de possibilitar um controle seguro, de baixo custo e de baixa potência em redes sem fio para o controle de diversos equipamentos, incluindo soluções para a automação predial, aplicações em telemedicina e entretenimento (jogos).

2.1.4 - Faixa de Frequência

Os dispositivos baseados na tecnologia ZigBee operam na faixa ISM que não requer licença para funcionamento, incluindo as faixas de 2,4GHz (Global), 915Mhz (América) e 868Mhz (Europa) e com taxas de transferência de dados de 250kbps em 2,4GHz, 40kbps em 915Mhz e 20kbps em 868Mhz. A figura 2.2. ilustra as bandas de operação do ZigBee. *(Pinheiro, 2006)*

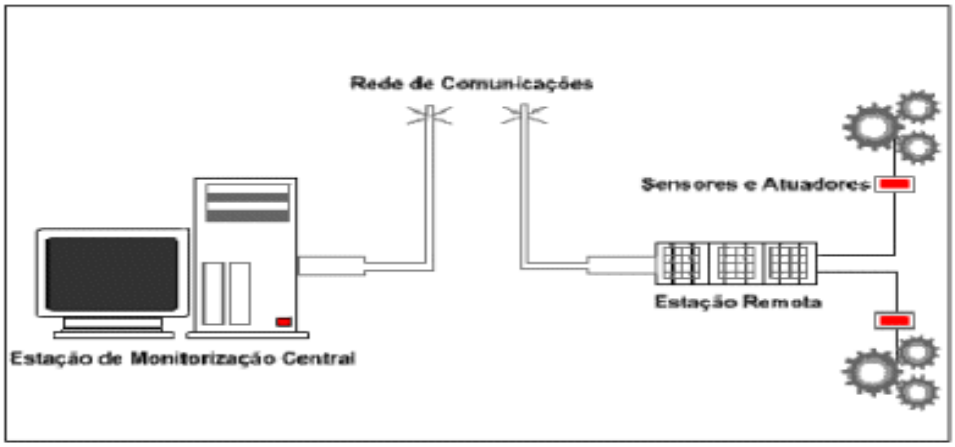


Figura 2.2 - Bandas de operação do ZigBee

Fonte: (Pinheiro, 2006)

O padrão oferece atualmente interfaces com velocidades de conexão compreendidas entre 115kbps e 250kbps e com um alcance de transmissão entre 70m e 300m, dependendo diretamente da potência dos equipamentos e de características ambientais (obstáculos físicos, interferência eletromagnética, etc).

Quanto ao problema de alimentação dos dispositivos, os módulos de controle dotados com esta nova tecnologia podem ser alimentados até mesmo por baterias (pilhas) comuns, sendo que sua vida útil está relacionada diretamente com a capacidade da bateria e a aplicação a que se destina. Nesse aspecto, o protocolo ZigBee foi projetado para suportar aplicações com o mínimo de consumo (com pilhas comuns, um dispositivo pode funcionar até 6 meses).

2.1.5 - Estrutura

Podemos identificar dois tipos de dispositivos em uma rede ZigBee, definidos pelo IEEE:

- Full Function Device (FFD) - pode funcionar em toda a topologia do padrão, desempenhando a função de coordenador da rede e conseqüentemente ter acesso a todos os outros dispositivos. Trata-se de dispositivos de construção mais complexa;

- Reduced Function Device (RFD) – é limitado a uma configuração com topologia em estrela, não podendo atuar como um coordenador da rede. Pode comunicar-se apenas com um coordenador de rede. São dispositivos de construção mais simples.

Devemos observar que em topologias com configuração estrela, uma rede ZigBee requer pelo menos um dispositivo FFD atuando como coordenador da rede e os demais dispositivos podem ser do tipo RFD para reduzir o custo do sistema. Para topologias ponto-a-ponto (Peer-to-Peer) e em árvore, todos os dispositivos devem ser FFD. *(Pinheiro, 2006)*

A Figura 2.3 mostra as três topologias de redes para ZigBee, a topologia de Estrela, Árvore e ponto à ponto.

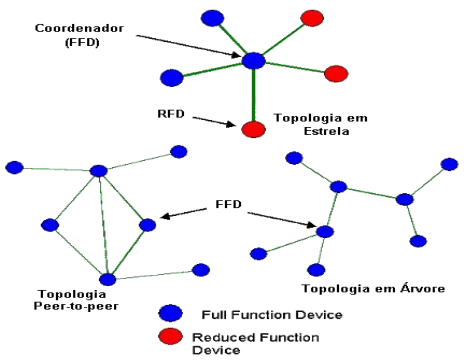


Figura 2.3 - Topologias para redes ZigBee

Fonte: (Pinheiro, 2006)

O Quadro 2.2 apresenta uma comparação entre os dispositivos de uma rede ZigBee com suas principais características:

Quadro 2.2 - Tabela de funcionalidades dos dispositivos ZigBee

Coordenador da Rede – FFD	Nó da Rede – RFD
Ajustes de parâmetros da rede	Função passiva na rede
Transmite informações pela rede	Efetua buscas por redes disponíveis
Gerencia os nós da rede	Transferência de dados da aplicação
Armazena informações dos nós de rede	Determina o status dos dados
Distribui mensagens entre nós de rede	Solicitam dados ao coordenador da rede
Opera tipicamente no estado "active"	Pode permanecer no estado "sleep" por longos períodos

2.1.6 - Características do Padrão

O padrão ZigBee (IEEE 802.15.4) foi projetado objetivando apresentar algumas características.O consumo de potência baixo e implementação simples, com interfaces de baixo custo. Possui dois estados principais de funcionamento: "active" para transmissão e recepção e "sleep", quando não está transmitindo. A simplicidade de configuração e redundância de dispositivos (operação segura), também é uma característica do padrão. Assim como a densidade elevada dos nós por a rede. As camadas PHY e MAC permitem que as redes funcionem com grande número de dispositivos ativos. Este atributo é crítico para aplicações com sensores e redes de controle. *(Pinheiro, 2006)*

2.1.7 - Camadas de Protocolos

A publicação do padrão IEEE 802.15.4, definiu interfaces com baixas taxas de transmissão (menores que 250Kbps) e estabeleceu uma estrutura de rede que incorpora

os conceitos de redes ad hoc, características de conexão em malha e em multi-hop (múltiplos saltos). Adicionalmente, novos algoritmos de segurança e perfis de aplicação foram definidos objetivando garantir a segurança e a perfeita interação entre os diversos equipamentos. Abaixo, na Figura 2.4, temos as camadas de protocolos do Zigbee. *(Pinheiro, 2006)*

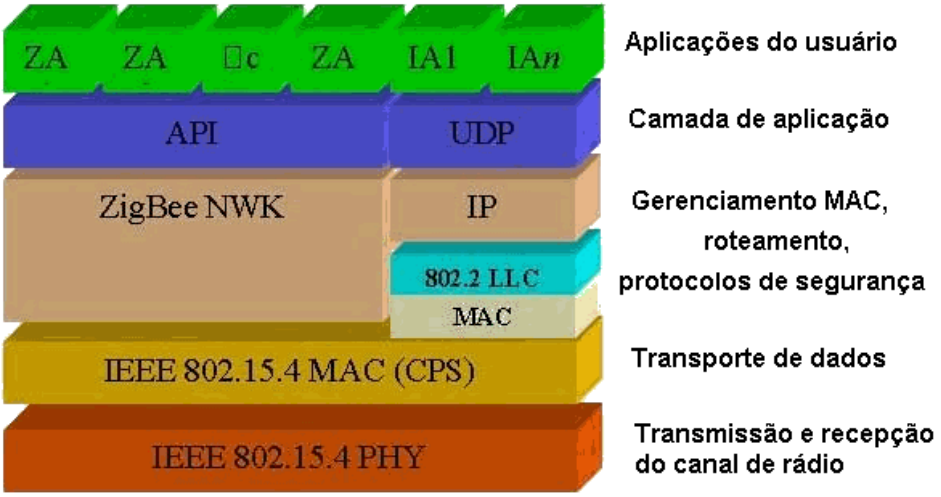


Figura 2.4 - Camadas de protocolos ZigBee

Fonte: (Pinheiro, 2006)

A **camada física (PHY)** foi projetada para acomodar as necessidades de interfaces de baixo custo, permitindo níveis elevados de integração. O uso da técnica de transmissão de Seqüência Direta (DSS) permite que os equipamentos sejam muito simples, possibilitando implementações mais baratas.

A **camada do Media Access Control (MAC)** foi projetada para permitir topologias múltiplas com baixa complexidade, onde o gerenciamento de energia, por exemplo, não requer modos de operação complexos. O MAC também permite que um dispositivo com funcionalidade reduzida (RFD) opere na rede sem a necessidade de grandes quantidades de memória disponíveis, podendo controlar também um grande número de dispositivos sem a necessidade de colocá-los "em espera", como ocorre em algumas tecnologias sem fio.

A **camada de rede** foi projetada para possibilitar o crescimento da rede sem a necessidade de equipamentos de transmissão de potência mais elevada. A camada de

rede também pode operar quantidades grandes de nós de rede com latências relativamente baixas.

A camada NWK utiliza um algoritmo que permite implementações da pilha de protocolos visando balancear os custos das unidades em aplicações específicas, o consumo das baterias, buscando produzir soluções com o perfil específico de custo-desempenho para a aplicação. *(Pinheiro, 2006)*

2.1.8 - Dispositivos ZigBee

A especificação ZigBee define três tipos diferentes de dispositivos, com funções variadas, que são o ZigBee Coordinator, Router e o End Device, que são especificados na Figura 2.5.

O **ZigBee coordinator (Coordenador ZigBee - ZC) e**xiste em cada rede. Este dispositivo agrega o maior número de funções, por exemplo, o coordenador é capaz de criar uma rede tornando-se a raiz da árvore dessa rede, sendo, portanto, o único dispositivo capaz de comutar dados entre redes.

O **ZigBee Router (Roteador ZigBee - ZR)** age como um roteador intermediário, roteando dados para outros dispositivos.

O **ZigBee End Device (Dispositivo final ZigBee - ZED)** contêm funções apenas para trocar informações com seu nó pai (ou um roteador ou um coordenador); não podendo encaminhar dados para outros dispositivos. Como requer menos memória, pois não precisa armazenar informações de roteamento, é mais barato que um roteador ou um coordenador ZigBee. *(Azevedo, 2006)*

ZigBee Coordinator (ZC)	ZigBee Router (ZR)	ZigBee End Device (ZED)	ZigBee Network Layer Function
•			Establish a new ZigBee network
•	•		Assign logical network addresses
•	•		Permit other devices to join/leave
•	•		Maintain lists of neighbors and routes
•	•		Route network-layer packets
•	•	•	Transfer network-layer packets
•	•	•	Join/leave a ZigBee network

Figura 2.5 - Tipos de dispositivos e suas Funções

Fonte: (Azevedo, 2006)

2.1.9 - Topologia ZigBee

O protocolo define uma estrutura de rede que incorpora os conceitos de redes ad hoc, as características de conexão em malha e multi-hop. As topologias que podemos encontrar em uma rede Zigbee são a topologia em estrela (star), em árvore (tree) e em malha (mesh). Os principais componentes integrantes dessa rede são o coordenador (coordinator), os roteadores (routers) e os clientes. *(ZigBee, 2009)*

A topologia Estrela é a mais simples, onde tem um coordenador e os elementos clientes. Ela é indicada para ambientes que ofereçam poucos obstáculos para a transmissão dos sinais. Sua principal vantagem é a facilidade de implementação e coordenação. A desvantagem está na presença de um único coordenador, o que gera a dependência de todos os clientes para este e o alcance do sinal de RF que nesse tipo de rede não é muito grande.

A topologia Árvore é formada por sub-redes que se comunicam entre si através de elementos roteadores. Aqui temos mais de um elemento coordenador, cada um gerenciando uma rede diferente. Sua aplicação principal está na comunicação de

dispositivos situados em andares diferentes e ou entre salas distantes entre si, separadas por paredes. O alcance do sinal de RF deste tipo de rede é maior, tendo em vista que um coordenador pode controlar um cliente ligado em outra rede através de uma solicitação ao outro coordenador.

Já a topologia Malha permite que, com a entrada de novos dispositivos, a rede se ajuste automaticamente durante sua inicialização, otimizando o tráfego de dados. Com essa topologia é possível construir redes mais extensas e de maior complexidade, possibilitando o controle e monitoração de grandes áreas. *(Pinheiro, 2006)*

A Figura 2.6 exemplifica um modelo de rede ZigBee. Pode-se observar dois tipos de rede na imagem, o tipo malha e o tipo estrela.

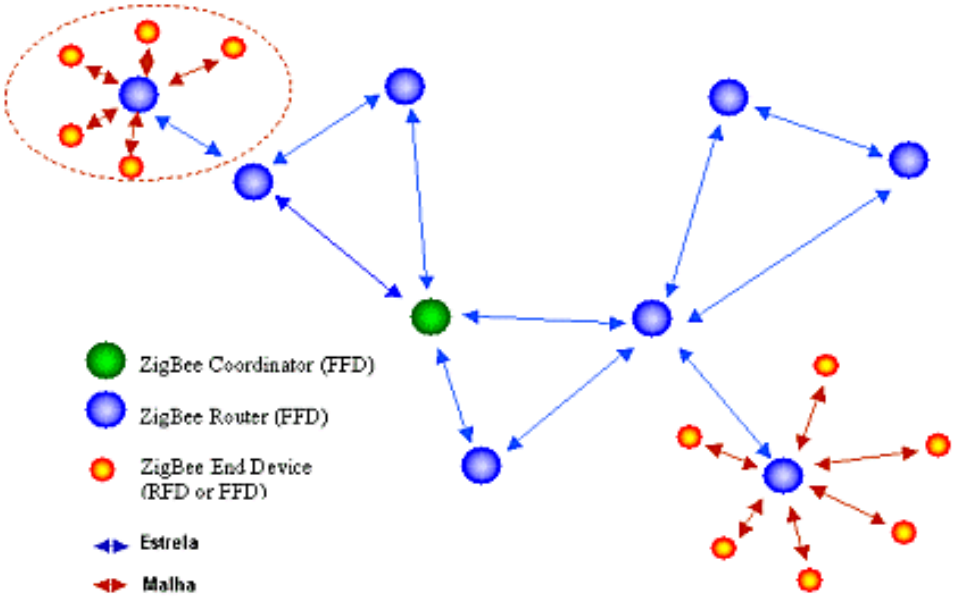


Figura 2.6 – Modelo de rede Zigbee

Fonte: (Pinheiro, 2006)

2.1.10- Vantagens e aplicações do ZigBee

O protocolo ZigBee foi projetado para transmitir dados mesmo em ambientes hostis à rádio freqüência, normalmente encontrados em aplicações industriais e comerciais. *(McBride, 2008)*

O protocolo ZigBee oferece:

- Ciclo do “Standby” configurável - Aumenta a vida útil da bateria;

- Baixa latência;
- Suporte para múltiplas topologias de rede: estática, dinâmica, estrela e mesh;
- Modulação DSSS (Direct Sequence Spread Spectrum);
- Até 65.000 unidades na mesma rede;
- Criptografia AES de 128-bit para assegurar um canal seguro entre os dispositivos;
- Redução na colisão de pacotes;
- Indicador da qualidade do link;

O protocolo ZigBee oferece conjunto de ferramentas de segurança para garantir redes seguras e confiáveis. Lista de controle de acesso, temporizadores de pacotes e criptografia AES (Advanced Encryption Standard) certificado pelo Instituto Nacional de Padrões e Tecnologia dos Estados Unidos (NIST – National Institute of Standard and Tecnology) auxiliam na proteção dos dados transmitidos. *(McBride, 2008)*

Com o ZigBee é possível implantar amplas soluções de redes sem fio com baixo custo e baixo consumo de energia. Ele é capaz de funcionar durante anos com baterias comuns em diversas aplicações de monitoramento como controle de iluminação, medidores, sensores de fumaça ou gás carbônico, telemetria sem fio, controle de aquecedores, ventiladores e condicionadores de ar (HVAC), controle de aquecedores de fluídos, segurança doméstica, controle de perímetro, controle de cortinas e claridade e etc.

“São quatro horas da manhã em uma fazenda do interior. Sensores distribuídos pela plantação informam a quantidade de água presente no solo e a umidade do ar. Os funcionários da fazenda analisam os dados para decidir onde e quando irrigar para otimizar a produção. A informação também pode ser útil como um sistema de alarme para situações adversas como a condensação da água. A produtividade aumenta e recursos preciosos são utilizados de forma mais eficiente.”

Os sensores distribuídos pela lavoura estão interconectados em uma rede “mesh”. Se algum ficar indisponível a rede é reparada automaticamente, cada unidade é

capaz de conectar-se com outra dinamicamente para encontrar uma nova rota e assim restabelecer conexão de rede.

Um componente chave para protocolo ZigBee é a capacidade de suportar redes do tipo mesh. Em uma rede mesh, os pontos estão interconectados com outros de forma que pelo menos dois caminhos conectam cada um deles a rede. Conexões entre os pontos são atualizadas dinamicamente e otimizada em condições adversas. Em alguns casos, pontos conectados a apenas um, são utilizados para restabelecer uma rede mesh parcialmente funcionando.

Redes mesh são descentralizadas por natureza, cada ponto é auto roteavel a capaz de conectar-se a outros pontos conforme a necessidade. As características de uma topologia mesh e roteamento ad-hoc fornecem maior estabilidade em condições instáveis ou falhas em um ponto único.

2.1.11 - Módulos XBee & XBee-Pro

Digi é um membro da ZigBee Alliance e tem desenvolvido soluções baseadas na arquitetura ZigBee. Os módulos XBee e XBee-PRO oferecem uma solução simples de implementar e um grande impulso tanto em variedade quanto em confiabilidade para empresas interessadas em oferecer ZigBee. Tais módulos têm as seguintes características:

• Compacto;

• Pronto para conectar e comunicar sem fio;

• Otimizado para aplicações de baixo custo e baixa taxa de transferência;

• Bateria com tempo de vida estendido;

• Segurança robusta;

• Alta confiabilidade na transmissão de dados;

• Compatível pino-a-pino com outros módulos;

• O alcance dos módulos XBee-PRO é 2 a 3 vezes maior do que um módulo ZigBee comum.

2.2 - Placa CON-USBBEE

2.2.1 - Introdução

A ROGERCOM desenvolveu a CON-USBBEE, com facilidade de conexão estilo PENDRIVE, para facilitar a conexão do módulo base XBEE/XBEE-PRO ao computador, seja para atualização de firmware ou mesmo para fazer coleta de dados ou controle. A Figura 2.7 ilustra a placa CON-USBBEE. *(RogerCom, 2008)*

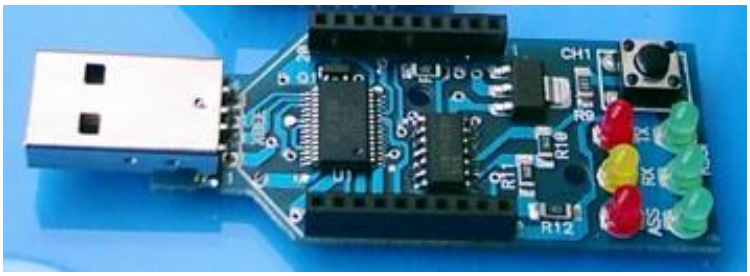


Figura 2.7 – Placa CON-USBBEE

Fonte: (RogerCom, 2008)

A placa CON-USBEE aceita tanto o módulo XBee™ como o XBee-Pro™, como são totalmente compatíveis, Redes ZigBee podem ser construídas com ambos os módulos, simultaneamente.

A placa CON-USBBE usa um chip conversor USB/Serial; regulador de tensão LDO (baixa queda de tensão), comparador de tensão conectado aos LEDs (RSSI) que simulam a força do sinal de RF; LEDs indicadores de TX, RX, módulo ligado (ASS), e um micro-botão para "resetar" o módulo XBee/XBee-Pro™, como é mostrado na Figura 2.8.

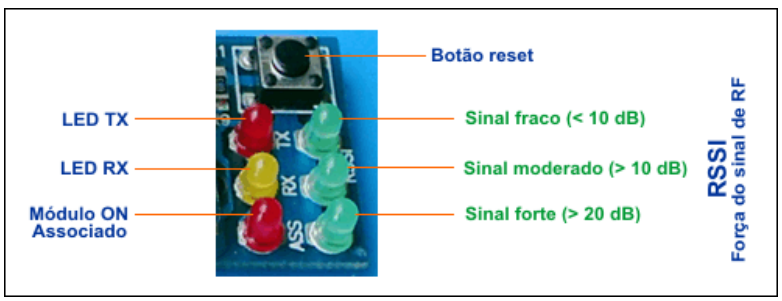


Figura 2.8 - Botão Reset e LEDs indicadores da placa CON-USBBEE

Fonte: (RogerCom, 2008)

Quando instalamos no computador o driver USB para (Windows 98, ME, 2000, XP,Vista, x64 e também para Linux e Mac) que acompanha a placa, é criada uma porta COMx virtual quando a placa CON-USBBEE é plugada. Assim, é possível através de um programa (escrito em C/C++Builder, Delphi, VB, Java, C#, etc), se comunicar com a placa como se fosse uma comunicação serial padrão RS232. Também é possível acessar a placa através de uma DLL, que oferece mais recursos na programação. A Figura 2.9 ilustra a placa CON-USBBEE numa visão inferior.

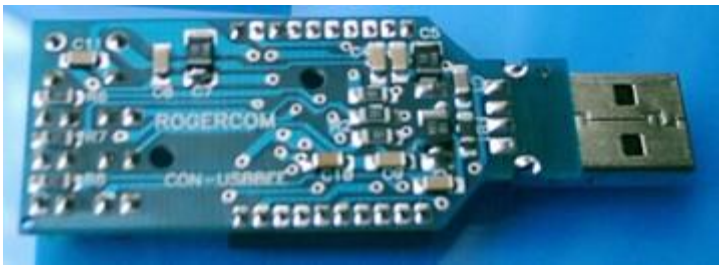


Figura 2.9 - Placa CON-USBBEE (visão inferior)

Fonte: (RogerCom, 2008)

2.2.2 - Característica

Abaixo seguem as principais características do CON-USBBEE:

A) Converte a interface Serial 3.3v do módulo XBee/XBee-Pro para USB;

B) Não precisa fonte de alimentação, a corrente é fornecida pelo próprio Bus USB;

C) Tem a mesma facilidade de conexão que um Pendrive;

D) LEDs indicadores de transmissão (TX), recepção (RX), Ligado e sinal RSSI;

E) Freqüência de operação: ISM 2.4 GHz;

F) Taxa de dados de RF: 250.000 bps;

G) Taxa de dados da Interface (Data Rate): 115.200 bps;

H) Alcance em áreas internas ou urbanas: 90m/100m;

I) Alcance em linha de visão (em campo aberto): 1,6Km;

J) Encriptação de 128-bit AES;

K) Comanda todos os módulos remotos XBee/XBee-Pro que estejam na Rede;

L) Troca de dados entre PCs e laptops;

M) Ideal para automação residencial, industrial, etc;

N) Totalmente compatível com o aplicativo X-CTU da DIGI™ para configuração de parâmetros e atualização do firmware nos módulos XBee/XBee-Pro. *(RogerCom, 2008)*

2.3 - Placa RCON-HOMEBEE

2.3.1 - Introdução

A função da placa HOMEBEE é a de automatizar determinados ambientes numa residência, escritório ou empresa. Ela pode trabalhar com ou sem fio para se comunicar com um computador ou outro dispositivo como CLP, microcontrolador, etc. Com fio opcionalmente a comunicação pode ser feita via RS232/TTL ou a partir de um conversor USB/Serial. Usando transmissão serial ou ZigBee no modo transparente, o controle da placa HOMEBEE é feita através do envio de 2 bytes, sendo o primeiro o identificador e o segundo o comando. Sem fio, a comunicação se faz através do protocolo ZigBee/IEEE 802.15.4, usando dois módulos transceivers Xbee ou Xbee-Pro. É possível usar encriptação AES de 128 bits, endereçamento de 16 bits, definir numero do canal e rede, entre outras possibilidades. *(RogerCom, 2008)*

A Placa HOMEBEE possui duas saídas a relés, que podem ser usadas para ligar ou desligar dispositivos com tensão até 220v e corrente de 10A; duas saídas TTL 5v, duas entradas digitais para conectar interruptores ou sensores de contato seco.

No PC, através da criação de um software especifico pelo usuário, é possível gerenciar uma rede de placas HOMEBEE controladas por uma única placa CON-USBBEE.

2.3.2 - Hardware

A Figura 2.10 traz o componentes da placa numerados, para serem explicados em seguida.

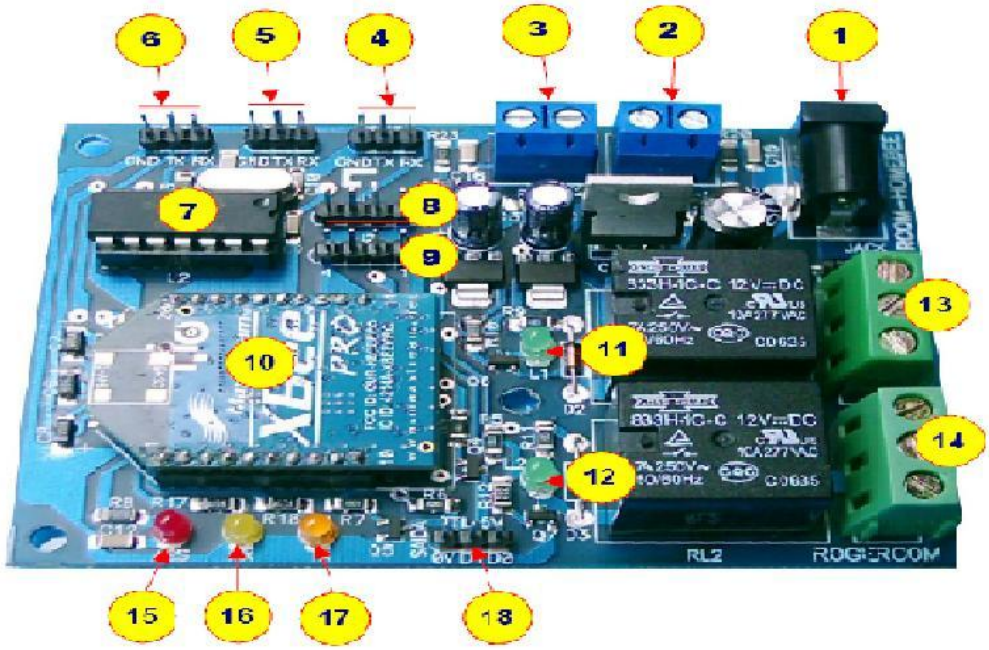


Figura 2.10 – Placa RCON-HOMEBEE

Fonte: (RogerCom, 2008)

1 – Alimentação:

A placa HOMEBEE deve ser alimentada por uma fonte externa capaz de fornecer entre 12-24v/600mA.

2 e 3 – Entradas Digitais E1 e E2:

As entradas Digitiais E1 e E2, como ilustradas na Figura 2.11, são entradas de contato seco. Um simples curto entre seus terminais, gera um pulso que é detectado pela placa, e enviado pela Serial ou via módulo XBee/XBee-Pro, conforme as configurações dos Jumps. Através de um software de configuração da placa é possível associar as entradas E1 e E2 aos relés R1 e R2.

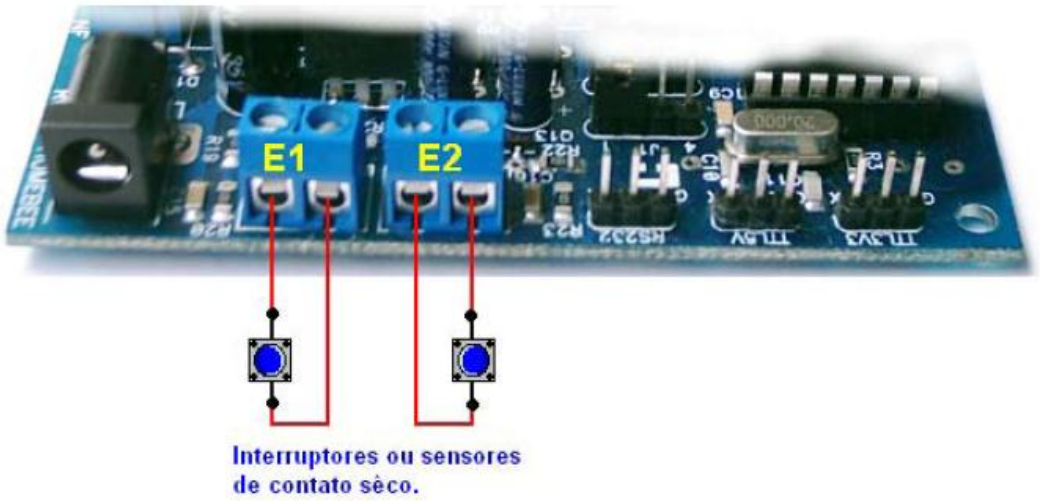


Figura 2.11 – Entradas E1 e E2

Fonte: (RogerCom, 2008)

4,5 e 6 – Interfaces Seriais RS232, TTL5v e TTL3v3:

A placa HOMEBEE dispõe de três opções para comunicação serial via cabo (RS232, TTL5v, TTL3v3), como pode-se ver na Figura 2.12. Após a escolha através dos jumps, somente um canal estará disponível.

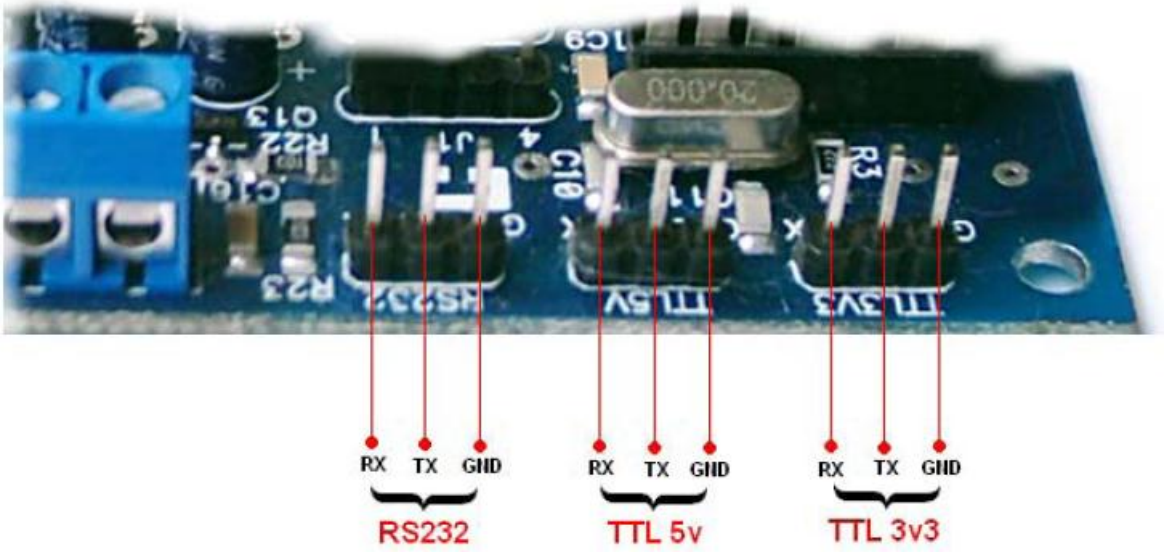


Figura 2.12 – Interfaces Seriais

Fonte: (RogerCom, 2008)

7 – Microcontrolador;

Controla todas as funções da placa HOMEBEE. O microcontrolador usado é o PIC16F688 com tecnologia nanowatt alimentado com 3v3v.

8 e 9 – Configuração de Jumps:

A Figura 2.13 ilustra os Jumps, J1 e J2, que são configurados seguindo as informações citadas a seguir.

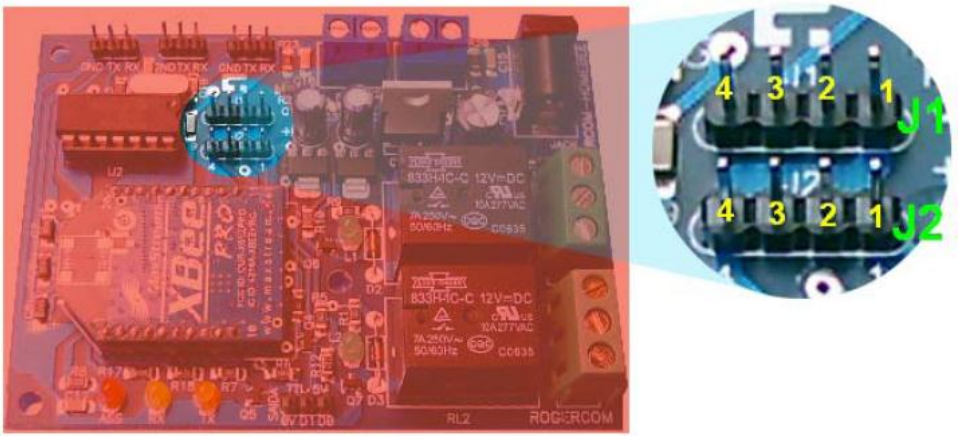


Figura 2.13 – Jumps, J1 e J2

Fonte: (RogerCom, 2008)

Configuração 1: XBee/XBee-Pro ←→ PIC

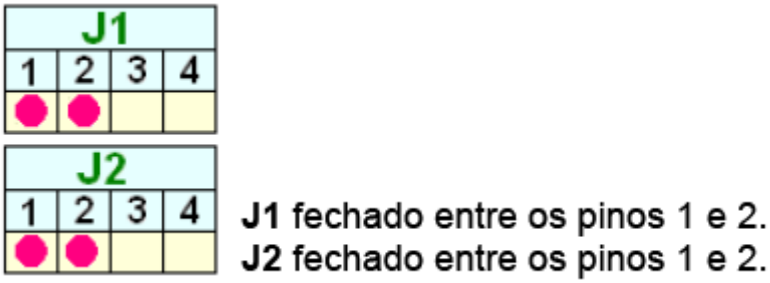


Figura 2.14 – Configuração de Jumps

Fonte: (RogerCom, 2008)

Configuração 2: PIC ←→ RS232, TTL5v, TTL3v3

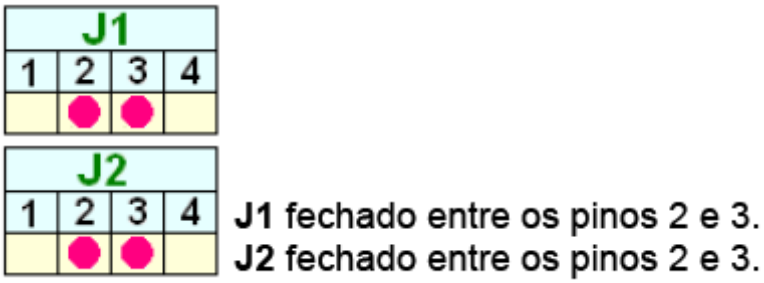


Figura 2.15 – Configuração de Jumps

Fonte: (RogerCom, 2008)

Configuração 3: XBee/XBee-Pro←→ RS232, TTL5v, TTL3v3

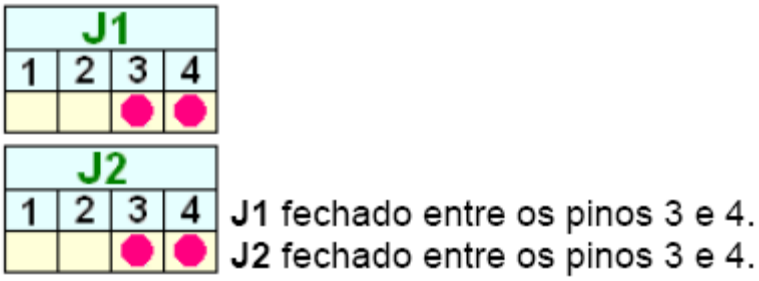


Figura 2.16 – Configuração de Jumps

Fonte: (RogerCom, 2008)

10 – Conector para modulo XBee/XBee-Pro:

Para que a placa HOMEBEE estabeleça comunicação sem fio, é necessário incluir um módulo Xbee ou XBee-Pro.

11 e 12 – LED's verdes L1 e L2:

Quando estão ligados indicam que o relé 1 e/ou 2 estão ligados. Quando apagados os relés 1 e/ou 2 estão desligados.

13 e 14 – Saidas a Relés (R1 e R2):

Atraves das saídas a relés, mostradas na Figura 2.17, é possível ligas/desligar dispositivos conectados a rede eletrica 110 ou 220v, ou mesmo aqueles alimentados com corrente continua.

NA – Interruptor **N**ormalmente **A**berto;
C – Comum
NF – Interruptor **N**ormalmente **F**echado.

Figura 15 – Conectores das saídas dos Relês

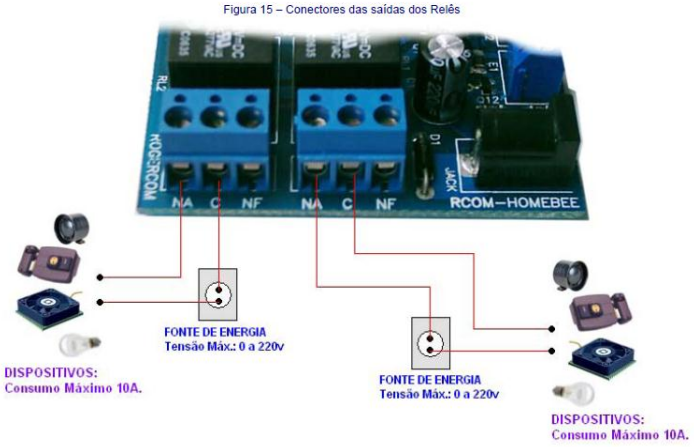


Figura 2.17 – Saída a Relés

Fonte: (RogerCom, 2008)

15 – LED vermelho (Ass):

Quando aceso/piscando, indica que o modulo XBee/XBee-Pro da placa está ligado/operando.

Quando aceso sem piscar, indica que ele está dormindo.

16 – LED laranja (TX):

Quando aceso/piscando, indica que o modulo XBee/XBee-Pro da placa está transmitindo dados.

17 – LED amarelo (RX):

Quando aceso/piscando, indica que o modulo XBee/XBee-Pro da placa está recebendo dados.

18 – Saidas TTL digitais (D0 e D1):

São saídas TTL 5v, que pode ser usadas para controlar um driver de relés externo, ou mesmo, enviar sinais para um microcontrolador. Veja a Figura 2.18.

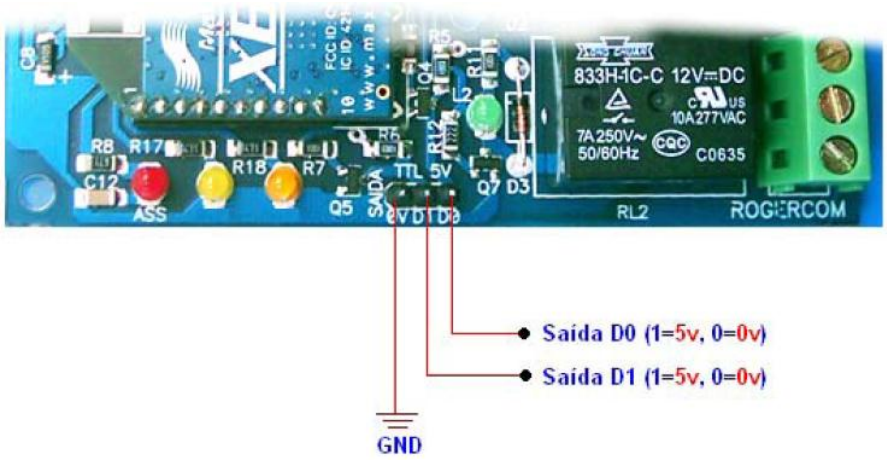


Figura 2.18 – Saídas TTL

Fonte: (RogerCom, 2008)

2.3.3 - Características

- Compatível com módulos XBee e XBee-Pro ZB ou IEEE 802.15.4. Placa para ligar/desligar lâmpadas, aparelhos eletro-eletrônicos, fechaduras elétricas, irrigação de jardins, abrir/fechar portas, portões, cancelas, etc;
- Dimensões: 9,0 cm x 6,5 cm.
- Características:

  • Segurança com encriptação de 128-bit AES (nos XBee/XBee-Pro ZB ou IEEE 802.15.4);

  • 2 Saídas tipo contato seco a Relés 110/220v / 10A;

  • 2 Entradas digitais contato seco;

  • 2 Saídas TTL 5v;

  •Fonte de alimentação 12v/500mA (não inclusa);

  • Interface serial opcional:

  - **RS232** (TX, RX, GND);

  - **TTL 5v** (TX, RX, GND);

  - **3,3v** (TX, RX, GND);

  (RogerCom, 2008)

2.4 - Linguagem de Programação

2.4.1 - Conceito

Podemos imaginar o computador como uma super calculadora, capaz de fazer cálculos muito mais rápido que nós, mas para isso devemos dizer para o computador o que deve ser calculado e como deve ser calculado. A função das linguagens de programação é exatamente essa, ou seja, servir de um meio de comunicação entre computadores e humanos. *(Andrade, 2007)*

Existem dois tipos de linguagens de programação: as de baixo nível e as de alto nível. Os computadores interpretam tudo como números em base binária, ou seja, só entendem zero e um. As linguagens de baixo nível são interpretadas diretamente pelo computador, tendo um resultado rápido, porém é muito difícil e incômodo se trabalhar com elas. Exemplos de linguagens de baixo nível são a linguagem binária e a linguagem Assembly.

Já as linguagens de alto nível são mais fáceis de trabalhar e de entender, as ações são representadas por palavras de ordem (exemplo faça, imprima, etc) geralmente em inglês, foram feitos assim para facilitar a memorização e a lógica. Elas não são interpretadas diretamente pelo computador, sendo necessário traduzí-las para linguagem binária utilizando-se de um programa chamado compilador.

2.4.2 – C#(Csharp)

C# (CSharp) é uma linguagem de programação orientada a objetos criada pela Microsoft, faz parte da sua plataforma .Net. A companhia baseou C# na linguagem C++ e Java. *(França, 2007)*

A linguagem C# foi criada junto com a arquitetura .NET. Embora existam várias outras linguagens que suportam essa tecnologia (como VB.NET, C++, J#), C# é considerada a linguagem símbolo do .NET por três motivos. Foi criada praticamente do zero para funcionar na nova plataforma, sem preocupações de compatibilidade com código de legado. O compilador C# foi o primeiro a ser desenvolvido. A maior parte das classes do .NET Framework foram desenvolvidas em C#.

A criação da linguagem, embora tenha sido feita por vários desenvolvedores, é atribuída principalmente a Anders Hejlsberg, hoje um Distinguished Engineer na

Microsoft. Anders Hejlsberg era desenvolvedor de compiladores na Borland, e entre suas criações mais conhecidas estão o Turbo Pascal e o Delphi.

2.4.3. – Características

C# (CSharp) é, de certa forma, a linguagem de programação que mais diretamente reflete a plataforma .NET sobre a qual todos os programas .NET executam. C# está de tal forma ligado a esta plataforma que não existe o conceito de código não-gerenciado (unmanaged code) em C#. Suas estruturas de dados primitivas são objetos que correspondem a tipos em .NET. A desalocação automática de memória por garbage colletor além de várias de suas abstrações tais como classes, interfaces, delegados e exceções são nada mais que a exposição explicita recursos do ambiente .NET.

Quando comparada com C e C++, a linguagem é restrita e melhorada de várias formas. Os ponteiros e aritmética sem checagem só podem ser utilizados em uma modalidade especial chamada modo inseguro (unsafe mode). Normalmente os acessos a objetos são realizados através de referências seguras, as quais não podem ser invalidadas e normalmente as operações aritméticas são checadas contra sobrecarga (overflow). Os objetos não são liberados explicitamente, mas através de um processo de coleta de lixo (garbage collector) quando não há referências aos mesmos, previnindo assim referências inválidas. Os destrutores não existem. O equivalente mais próximo é a interface Disposable, que juntamente com a construção using block permitem que recursos alocados por um objeto sejam liberados prontamente. Também existem finalizadores, mas como em Java sua execução não é imediata. Como no Java, não é permitida herança múltipla, mas uma classe pode implementar várias interfaces abstratas. O objetivo principal é simplificar a implementação do ambiente de execução. O C# é mais seguro com tipos que C++. As únicas conversões implícitas por default são conversões seguras, tais como ampliação de inteiros e conversões de um tipo derivado para um tipo base. Não existem conversões implícitas entre inteiros e variáveis lógicas ou enumerações. Não existem ponteiros nulos (void pointers) (apesar de referências para Object serem parecidas). E qualquer conversão implícita definida pelo usuário deve ser marcada explicitamente, diferentemente dos construtores de cópia de C++. A sintaxe para a declaração de vetores é diferente ("int[] a = new int[5]" ao invés de "int a[5]"). Membros de enumeração são colocados em seu próprio espaço de nomes (namespace). O C# não possui modelos (templates), mas o C# 2.0 possui genéricos (generics).

Propriedades estão disponíveis, as quais permitem que métodos sejam chamados com a mesma sintaxe de acesso a membros de dados. E Recursos de reflexão completos estão disponíveis.

Apesar de C# ser freqüentemente tido como similar a Java, existem uma série de diferenças importantes. O Java não implementa propriedades nem sobrecarga de operadores. Ele não implementa um modo inseguro que permita a manipulação de ponteiros e aritmética sem checagem. Possui exceções checadas, enquanto exceções em C# são não checadas como em C++. Java não implementa o goto como estrutura de controle, mas C# sim. Java utiliza-se de comentários Javadoc para gerar documentação automática a partir de arquivos fonte. C# utiliza comentários baseados em XML para este propósito. E o C# suporta indexadores e delegados.

2.5 – Banco de Dados

2.5.1 – Microsoft Access

O Microsoft Access (nome completo Microsoft Office Access), também conhecido por MSAccess, é um Sistema de gerenciamento de banco de dados da Microsoft, incluído no pacote do Microsoft Office Professional, que combina o Microsoft Jet Database Engine com uma interface gráfica do utilizador (graphical user interface). Ele permite o desenvolvimento rápido de aplicações que envolvem tanto a modelagem e estrutura de dados como também a interface a ser utilizada pelos usuários.

Microsoft Access é capaz de usar dados guardados em Access/Jet, Microsoft SQL Server, Oracle, ou qualquer recipiente de dados compatível com ODBC.

O desenvolvimento da estrutura de dados se dá de forma muito intuitiva, bastando que o desenvolvedor possua conhecimetos básicos em modelagem de dados e lógica de programação. *(Microsoft, 2008)*

Programadores relativamente inexperientes e usuários determinados podem usá-lo para construir aplicações simples, sem a necessidade de utilizar ferramentas desconhecidas.

Primeiramente foi o nome de um programa de comunicação da Microsoft, destinado à competir com ProComm e outros programas. Esse produto fracassou e foi

abandonado. No segundo semestre de 1992 a Microsoft lançou seu primeiro Sistema de Gerenciamento de Banco de Dados e reusou o nome: o Microsoft Access (MS Access).

Para a Microsoft havia uma grande vantagem quanto ao mercado, como era a dominadora do seu próprio mercado, foi a primeira a lançar um software executável em plataforma Windows, enquanto que os outros programas deste segmento, liderados pela dBa-se, eram voltados para o ambiente DOS. Ao ser lançado por um preço bastante atrativo o MS Access 1.0 para Windows foi logo tomando conta do seu espaço. Um dos fatores que muito contribuiu pra isto foi o seu preço de apenas noventa e nove dólares e também contamos com os investimentos que os administradores da Microsoft fizeram como a compra da Fox Software por cento e setenta milhões de dólares.

Embora as vantagens de lançamentos, faltava de um pacote de ferramenta para desenvolvedores, o que foi princípio para muitas reclamações, assim como a limitação do software de 128MB de memória. Já em 1993 é lançado no mercado MS Access 1.1 com a ampliação para 1GB de memória e junto a ele veio Distribution Kit e os incentivos da para que os desenvolvedores criassem seus aplicativos e os vendessem sem a necessidade do cliente ter uma versão do Access em seu computador, bastava apenas utilizar o Access Runtime.

No entanto o programa ainda precisava de reajustes e dois anos depois modificações apareceram com a versão 2.0. Agora era possível trabalhar com 254 tabelas ao mesmo tempo e no mesmo arquivo de dados. O novo ambiente de programação dava facilidades ao desenvolvedor e agora também não era mais necessário digitar imensas linhas de códigos. Neste mesmo ano chega ao Brasil a versão em português do Access.

A nova versão do MS Access , em 1995, pulou para 7.0 acompanhando o pacote de programas Microsoft Office 7.0. Introduzida na era de 32 bits a nova versão do Access vinha acompanhada com a linguagem Visual Basic for Applications (VBA), com o seu próprio ambiente de programação. Além do recurso de replicar banco de dados, dando a possibilidade de trabalhar off-line em cópias de banco de dados, podendo ser sincronizada com a base central.

Em março de 1997 foi lançada a versão 8.0, preparada para internet, capaz de gravar hiperlinks e salvar arquivos em formato HTML. E em 1999 lança a versão 9.0, também conhecido como MS Access 2000, com suporte a OLE DB e um produto independente até hoje. Com isso, em 2002, a nova versão com suporte a uma linguagem mais comum, o XML (Extensible Markup Language), que é uma linguagem de marcação de dados e também dá uma maior integralidade com o browser, podendo criar páginas em formato HTML de acesso as dados do banco.

O Access 2003 na sua versão 11 traz maior integração com browser, além da linguagem VBA, marcação em XML e incorporação da linguagem SQL nas consultas de tabelas do banco.

2.5.2 – Características

Um dos benefícios do Access do ponto de vista do programador é sua relativa compatibilidade com o SQL – buscas podem ser visualizadas e editadas como sendo indicações de SQL, e estes por sua parte podem ser usados diretamente em Macros e Módulos VBA para manipular tabelas do próprio Access. Usuários podem misturar e usar ao mesmo tempo VBA e Macros para formulários de programação e lógica, além de serem oferecidos possibilidades com técnicas de orientação-objeto. *(FunctionX, 2007)*

O escritor de relatórios do Access, mesmo sendo capaz da criação sofisticada de relatórios, não é tão potente quanto outro escritor de relatórios de dados – Crystal Reports. MSDE (Microsoft SQL Server Desktop Engine) 2000, a mini-versão do MS SQL Server 2000, é incluído com a edição de desenvolvedor do Office XP e pode ser usado ao lado do Access como uma alternativa ao Jet Database Engine.

Versões anteriores do MSDE e do Microsoft Exchange Server de fato usam o motor do Jet para lidar com volumes elevados de dados e inserir uma camada de aplicação "falsa" para as aplicações que se encontram acima dela. A falta de conhecimento a respeito deste fato têm contribuído à uma desmerecida falta de respeito quanto à família de produtos Access/Jet, particularmente quando se diz respeito a projetos de "grande porte".)

A função "cortar e colar"(cut and paste) do Access pode torná-lo uma ferramenta útil para a conexão entre diversos bancos de dados (por exemplo, entre o Oracle e o Microsoft SQL Server durante conversões de dados ou bancos de dados).

Access disponibiliza várias funções de exportação e importação (import and export), que permitem a integração entre o Windows e outras aplicações de plataforma, muitas das quais podem ser executadas dentro das próprias aplicações ou manualmente pelo usuário. Serve, como exemplo, o bastante compacto formato SNP para a compartilhação de relatórios perfeitamente formatados entre usuários que não dispõem de uma versão completa do Access.

Diferentemente dos RDBMS completos, o motor de banco de dados Jet da Microsoft não dispõe de gatilhos de bancos de dados e procedimentos armazenados (stored procedures).

A partir do MS Access 2000 (Jet 4.0), há um sintaxe que permite a criação de pedidos de busca com parâmetros, semelhante à criação de procedimentos armazenados, porém tais procedimentos são limitados à uma declaração por procedimento.

O Microsoft Access permite que formulários contenham códigos que são ativados à medida que mudanças são feitas à uma tabela subjacente (contanto que as modificações são feitas apenas com aquele formulário), e é comum usar buscas "pass-through" e outras técnicas no Access para ativar procedimentos armazenados em RDBMSs que suportam este tipo de sistema.

Em arquivos de projeto de banco de dados do Access ADP (suportado em MS Access 2000 e adiante), as ferramentas relacionadas aos bancos de dados são completamente diferentes, uma vez que este tipo de arquivo é conectado ao MSDE ou ao Microsoft SQL Server, ao invés de usar o motor Jet. Deste modo, ele suporta a criação de quase qualquer objeto no servidor subjacente (tabelas com restrições e gatilhos, visualizações, procedimentos armazenados e UDF's).

Contudo apenas formulários, relatórios, macros e módulos são armazenados no arquivo ADP (outros objetos são armazenados no banco de dados "back end").

CAPÍTULO 3 – DESENVOLVIMENTO DO PROJETO

Neste capítulo serão apresentados a implementação, os testes e os resultados obtidos no desenvolvimento do projeto e, também, a simulação do funcionamento do protótipo. No item 3.1 será abordado o entendimento geral do projeto. O item 3.2 mostra a descrição dos dispositivos eletrônicos utilizados. O item 3.3 apresenta o software responsável pelo funcionamento do protótipo. No item 3.4 são mostrados testes e resultados, e por último é mostrada a simulação do protótipo, no item 3.5.

3.1 - Desenvolvimento do projeto

O projeto foi dividido em quatro etapas. A primeira etapa foi o planejamento geral do projeto, pensando no que fazer e por que fazer. Já na segunda etapa, a mais extensa, foi feito a redação da monografia, o seu estudo bibliográfico e a compra de materiais para o protótipo. Na terceira etapa foi executada a montagem do protótipo e a realização de testes, para verificar onde havia problemas e como resolvê-los. Na última etapa foi realizada a última verificação da monografia, atento a alguns erros que podiam existir. Na Figura 3.1 podemos ver as quatro etapas.

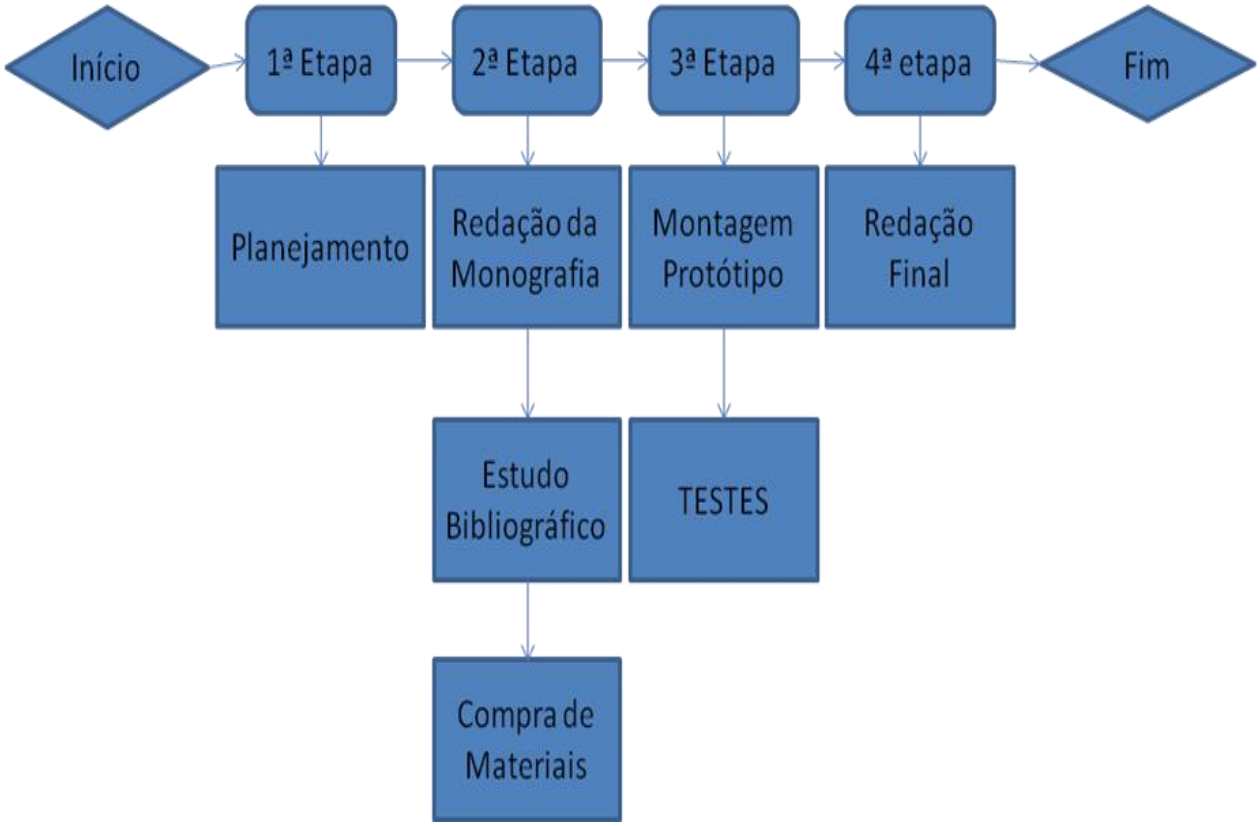


Figura 3.1 – Etapas do desenvolvimento do projeto

3.1.1 - Estrutura Geral do Projeto

Este projeto apresenta um protótipo de um sistema de automação residencial para iluminação. O objetivo desse sistema de automação é promover uma economia no gasto de energia e, também, comodidade ao usuário. Neste protótipo encontram-se os seguintes dispositivos eletrônicos:

-Placa CON-USBBEE com módulo XBee-Pro;
-Placa RCON-HOMEBEE com módulo XBee-Pro.

Esses dispositivos são monitorados e controlados por um sistema de gerenciamento implementado em C# (CSharp). Quando for executado algum comando, a placa receberá uma conifrmação. Com isso, o usuário poderá fazer consultas dos horários em que alguma ação foi efetuada. A comunicação entre o software e o hardware é feita via Zigbee - radiofreqüência.

A Figura 3.2 ilustra a comunicação entre o software e o dispositivo.

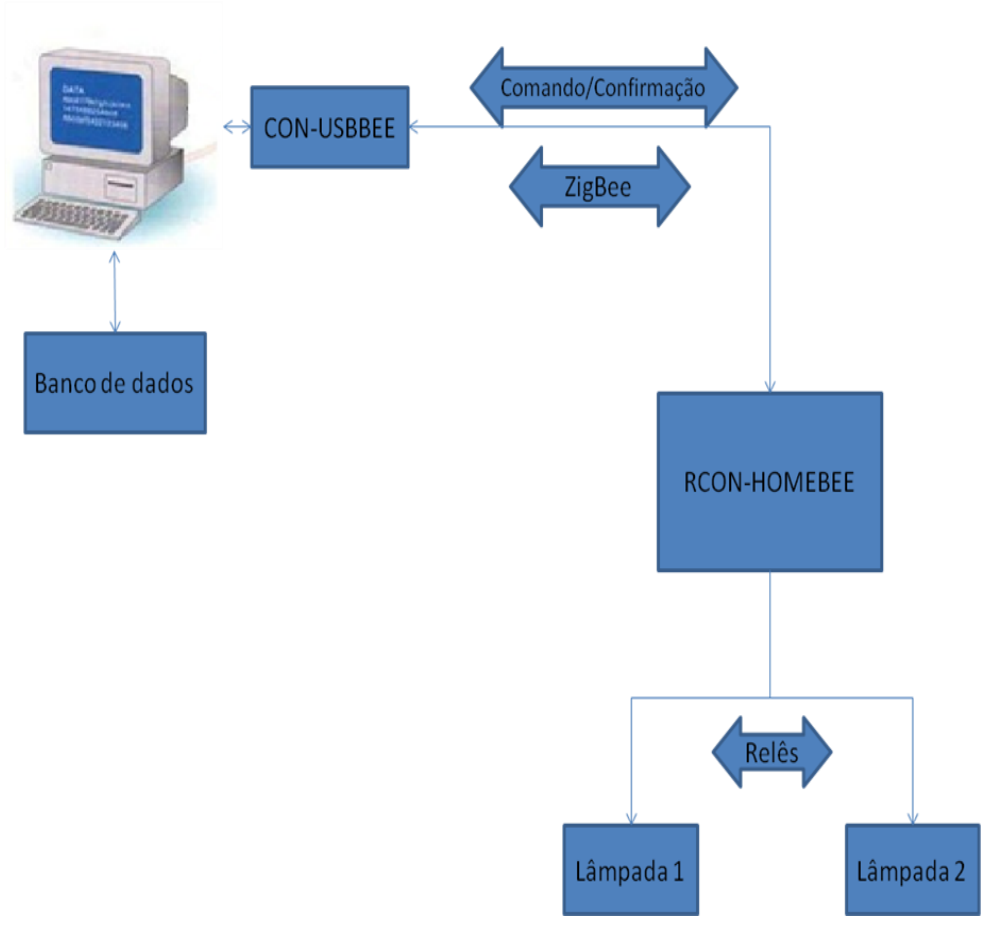


Figura 3.2 – Comunicação entre Software e hardware

O objetivo do projeto é ligar e desligar lâmpadas remotamente, através do computador. Para isso serão utilizadas as placas citadas acima.

3.1.2 – Funcionamento Básico do Projeto

O funcionamento básico do projeto é explicado na Figura 3.3.

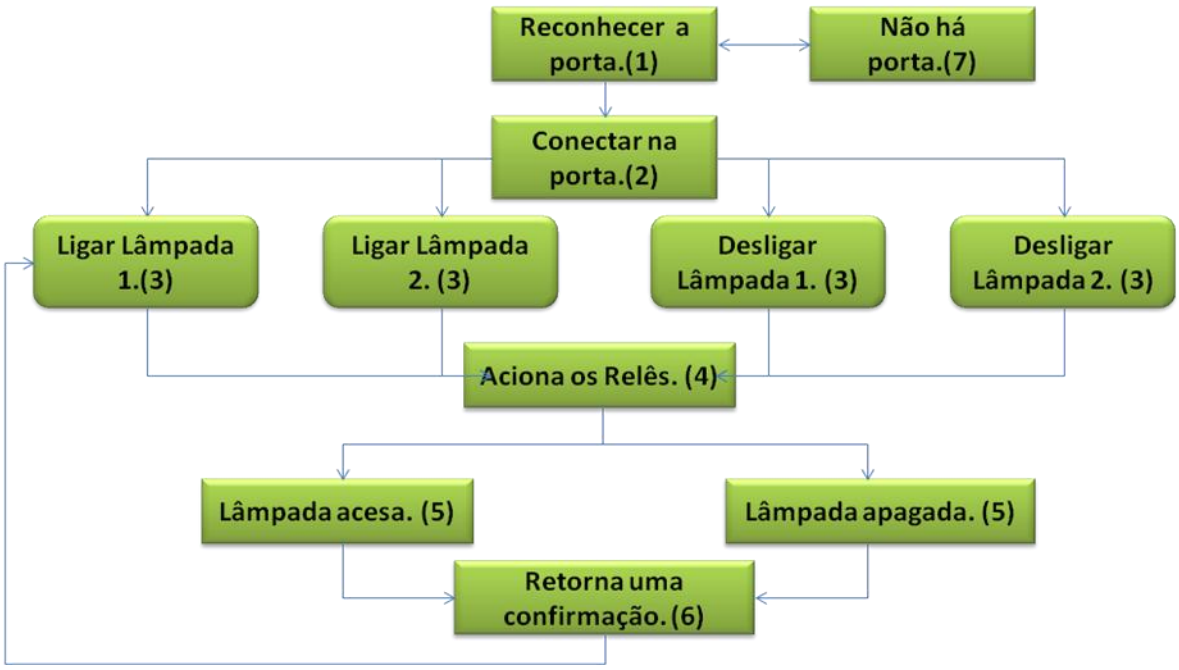


Figura 3.3 – Funcionamento do projeto

Os números na Figura 3.3 servem para seqüenciar as etapas do funcionamento do projeto. Primeiramente o sistema reconhecerá a porta (1) onde está conectada a placa CON-USBBEE, se não encontrá-la ele exibirá uma mensagem avisando que não foi encontrada nenhuma porta (7). Se encontrar a porta, basta conectá-la (2) e a placa já fica ativa. Depois de conectada o usuário escolherá o que deseja fazer, ligar ou desligar (3) as lâmpadas. Com a ação escolhida, o relé será acionado (4) e a lâmpada será acesa ou apagada (5). Então o sistema mostrará uma mensagem avisando o que foi executado (6). Depois o software trabalhará em um loop, entre as etapas (3), (4), (5) e (6), até a porta ser desconectada.

3.2 - Dispositivos Eletrônicos do Projeto

Os dispositivos eletrônicos utilizados no projeto foram os seguintes:

- Módulo XBee-Pro;

- Placa CON-USBBEE;
- Placa RCON-HOMEBEE;
- Dispositivos complementares.

3.2.1 - Especificações dos Dispositivos Utilizados

3.2.1.1 - Módulo XBee-Pro

Para a comunicação ZigBee foi utilizado o módulo XBee-Pro. No projeto serão utilizados dois módulos XBee-Pro, sendo um para a placa CON-USBBEE e um para a placa RCON-HOMEBEE, como mostrado nas Figuras 3.4 e 3.5.

Figura 3.4 – CON-USBBEE

Fonte: (RogerCom, 2008)

Figura 3.5 – RCON-HOMEBEE

Fonte: (RogerCom, 2008)

Os dois módulos serão configurados pelo software X-CTU. Esse software é desenvolvido pela MaxStream e é utilizado para configuração dos módulos XBee-Pro, como mostra a figura 3.6.

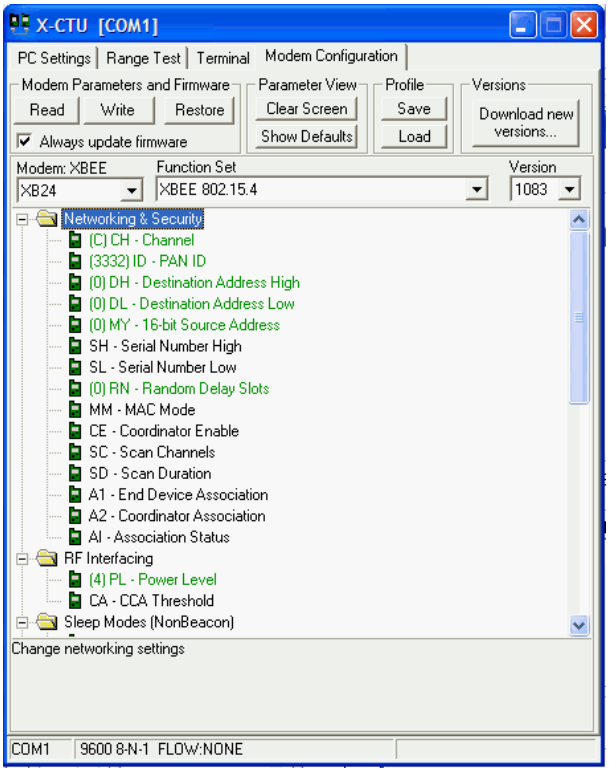


Figura 3.6 – Software X-CTU

3.2.1.2 - Placa CON-USBBEE

A placa CON-USBBEE tem a função de comunicar um dos módulos XBee-Pro com o computador através de uma porta USB. Através dessa placa que serão enviados os comandos para os relés.Ela também receberá uma confirmação de que a ação foi executada. Para utilizar a placa basta instalar o driver CDM 2.02.04. Com isso, será criada uma porta COMx para a placa, automaticamente. Na USBBEE, o módulo XBee-Pro será configurado com os seguintes padrões:

Endereço (DL): 5000

Endereço (MY): 5001

AP: 0

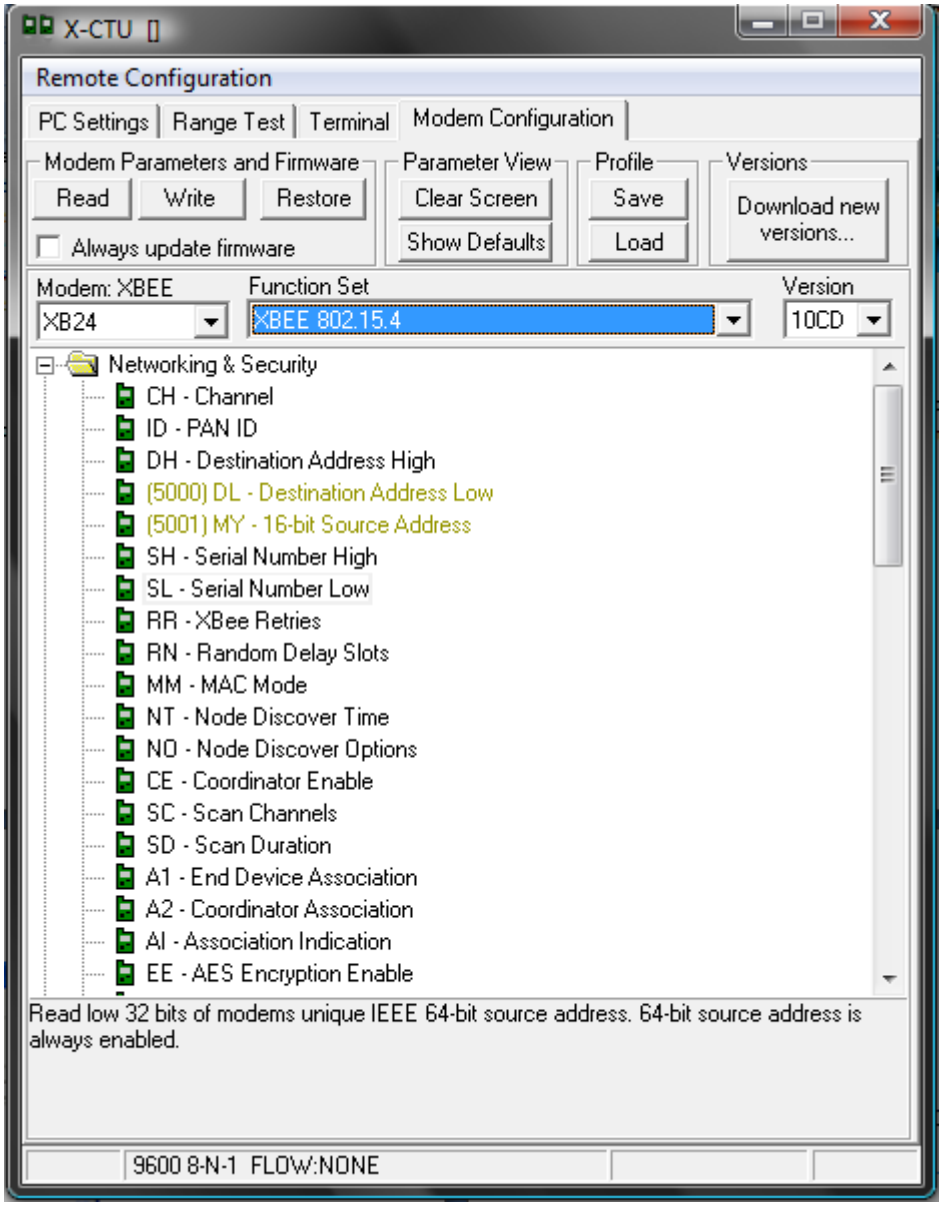


Figura 3.7 – Configuração CON-USBBE

3.2.1.3 - Placa RCON-HOMEBEE

A placa RCON-HOMEBEE receberá os comandos, enviados pelo software de gerenciamento, através do módulo XBee-Pro. Esses comandos acionaram os Relés 1 e/ou 2 para ligar/desligar. E enviará, também, a confirmação de que foi ligada ou desligada a lâmpada. A placa é alimentada por uma fonte externa capaz de fornecer 12-24 v/600mA. Ela vai controlar dois relés com lâmpadas de 220 v. Na HOMEBEE, o módulo XBee-Pro será configurado com os seguintes padrões:

Endereço (MY): 5001

Endereço (DL): 5000

AP: 0

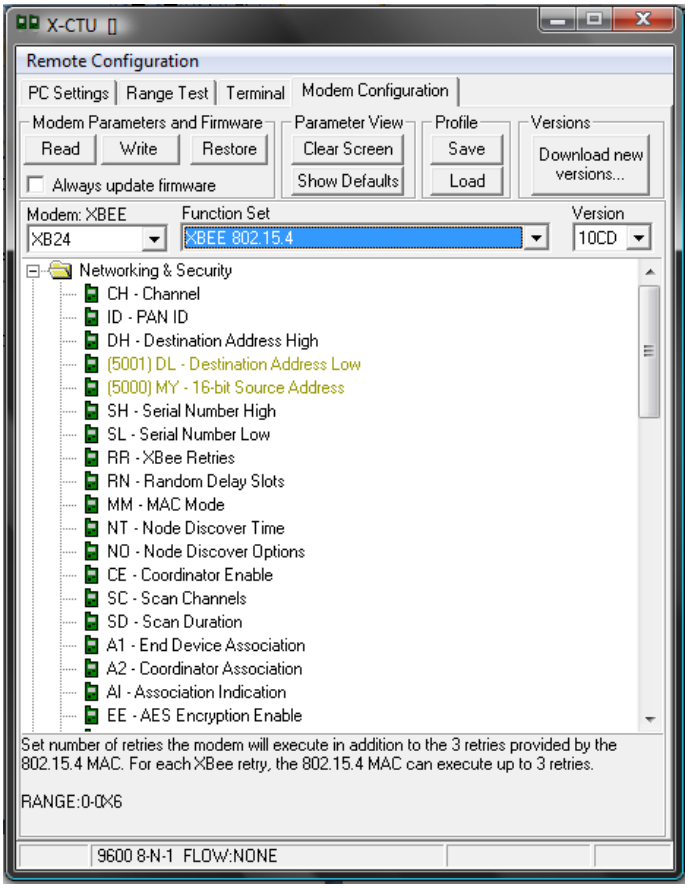


Figura 3.8 – Configuração RCON-HOMEBEE

Quando Algum relé é acionado, o seu LED é aceso.

Figura 3.9 – Relé aceso.

3.2.1.4 - Demais dispositivos

Serão utilizados também um computador, uma fonte de 18v e 2 lâmpadas. O computador será responsável pela execução dos comandos do software de gerenciamento e onde ficará o banco de dados com as informações para pesquisa. A fonte alimentará a placa RCON-HOMEBEE, lembrando que a placa CON-USBBEE é alimentada com a corrente fornecida pelo próprio Bus USB. Já as lâmpadas serão utilizadas para os testes do protótipo.

3.3 – Software

Para a comunicação entre os dispositivos foi criado um software de gerenciamento, que foi implementado em C# (CSharp). Toda informação enviada ou recebida pela placa é em forma de pacotes. A Fig ura 3.10 mostra o mapa de bits do byte de controle de escrita na placa.

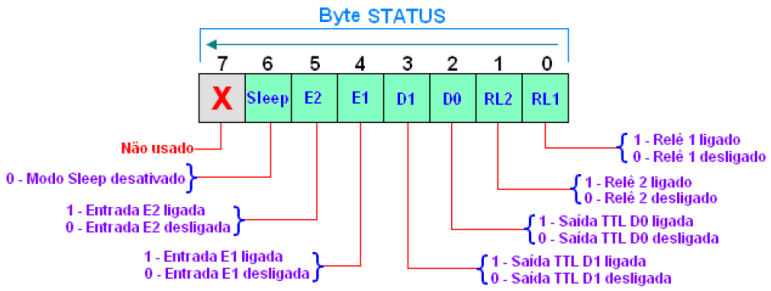


Figura 3.10 – Mapa de bits

Fonte: (RogerCom, 2008)

O Quadro 3.1 mostra o formato de um pacote XBee-pro para enviar dados de controle para a placa HOMEBEE.

Quadro 3.1 – Pacote de dados para envio de comando

Bytes	Descrição	
1	Delimitador Inicial.	7E
2	Tamanho dos bytes.	00 07
1	Identificador da API.	01
1	API Frame ID.	01
2	Parte baixa do endereço destino (DL).	50 01

1	Byte de opção.	7B
2	Pacote de dados.	01
1	Checksum.	30

Fonte: RogerCom, 2008

Exemplo de um pacote para ligar o relé 1;

7E 00 07 01 01 50 01 00 7B 01 30

3.3.1 – Funções principais do software

O software terá uma interface muito fácil de ser utilizada. Para isso, ele possuirá algumas funções como, conectar e desconectar a porta, ligar e desligar os relés 1 e 2 separadamente e os dois relés ao mesmo tempo.

Para reconhecer e conectar a porta COMx, serão utilizadas as seguintes linhas de código exibidas na Figura 3.11.

```
private void button1_Click(object sender, EventArgs e)
{
    try
    {
        if (cmbPortas.Text != "")
        {
            portaCom = new SerialPort(cmbPortas.SelectedItem.ToString(), 9600, Parity.None, 8, StopBits.One);

            portaCom.Open();
            MessageBox.Show("Porta Conectada.");
        }
        else
        {
            MessageBox.Show("É necessário escolher uma porta");
        }

        button1.Enabled = false;
        btnDesconectar.Enabled = true;
    }
    catch (Exception ex)
    {

        MessageBox.Show(ex.Message + "\r\n" + "Porta " + cmbPortas.SelectedItem.ToString() + " esta aberta");
    }

}
```

Figura 3.11 – Conectar Porta COMx.

Para ligar os relés 1 e 2, separadamente, e depois desligá-los um por vez, foram utilizadas as linhas de código exibidas nas Figuras 3.12, 3.13, 3.14 e 3.15. No comando foi utilizada a função “Write”.

```
private void btnOnRele1_Click(object sender, EventArgs e)
{
    try
    {
        byte[] rele1 = new byte[11];
        //7E 00 07 81 50 01 24 00 57 01 B1
        //7E 00 07 81 50 01 28 00 57 01 AD
        rele1[0] = 0x7E;
        rele1[1] = 0x00;
        rele1[2] = 0x07;
        rele1[3] = 0x01;
        rele1[4] = 0x01;
        rele1[5] = 0x50;
        rele1[6] = 0x01;
        rele1[7] = 0x00;
        rele1[8] = 0x7B;
        rele1[9] = 0x01;
        rele1[10] = 0x30;

        textBox1.Text = rele1.ToString() + "\r\n";
        portaCom.Write(rele1, 0, rele1.Length);

        lerPlaca();
        MessageBox.Show("Lâmpada 1 ligada.");
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message);
    }
}
```

Figura 3.12 – Ligar Relé 1.

```
private void btnOffRele1_Click(object sender, EventArgs e)
{
    try
    {
        byte[] rele1 = new byte[11];
        rele1[0] = 0x7E;
        rele1[1] = 0x00;
        rele1[2] = 0x07;
        rele1[3] = 0x01;
        rele1[4] = 0x01;
        rele1[5] = 0x50;
        rele1[6] = 0x01;
        rele1[7] = 0x00;
        rele1[8] = 0x7B;
        rele1[9] = 0x00;
        rele1[10] = 0x30;

        portaCom.Write(rele1, 0,rele1.Length);
        lerPlaca();
        MessageBox.Show("Lâmpada 1 desligada.");
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message);
    }
}
```

Figura 3.13 – Desligar Relé 1.

```
private void btnOnRele2_Click(object sender, EventArgs e)
{
    try
    {
        byte[] rele2 = new byte[11];
        //7E 00 07 01 01 50 01 00 7B 02 1F
        rele2[0] = 0x7B;
        rele2[1] = 0x00;
        rele2[2] = 0x07;
        rele2[3] = 0x01;
        rele2[4] = 0x01;
        rele2[5] = 0x50;
        rele2[6] = 0x01;
        rele2[7] = 0x00;
        rele2[8] = 0x7b;
        rele2[9] = 0x02;
        rele2[10] = 0x1f;

        portaCom.Write(rele2, 0, rele2.Length);
        MessageBox.Show("Lâmpada 2 ligada.");
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message);
    }
}
```

Figura 3.14 – Ligar Relé 2.

```
private void btnOffRele2_Click(object sender, EventArgs e)
{
    try
    {
        byte[] rele2 = new byte[11];
        rele2[0] = 0x7B;
        rele2[1] = 0x00;
        rele2[2] = 0x07;
        rele2[3] = 0x01;
        rele2[4] = 0x01;
        rele2[5] = 0x50;
        rele2[6] = 0x01;
        rele2[7] = 0x00;
        rele2[8] = 0x7b;
        rele2[9] = 0x00;
        rele2[10] = 0x1f;

        portaCom.Write(rele2, 0, rele2.Length);
        lerPlaca();
        MessageBox.Show("Lâmpada 2 desligada.");
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message);
    }
}
```

Figura 3.15 – Desligar Relé 2

Para ligar os dois relés ao mesmo tempo foram utilizadas as linhas de código mostradas na Figura 3.16, utilizando também a função “Write”.

```
private void button3_Click(object sender, EventArgs e)
{
    {
        try
        {
            byte[] rele12 = new byte[11];
            rele12[0] = 0x7B;
            rele12[1] = 0x00;
            rele12[2] = 0x07;
            rele12[3] = 0x01;
            rele12[4] = 0x01;
            rele12[5] = 0x50;
            rele12[6] = 0x01;
            rele12[7] = 0x00;
            rele12[8] = 0x7b;
            rele12[9] = 0x03;
            rele12[10] = 0x2e;

            portaCom.Write(rele12, 0, rele12.Length);
            lerPlaca();
            MessageBox.Show("Lâmpada 1 e 2 ligadas.");
        }
        catch (Exception ex)
        {
            MessageBox.Show(ex.Message);
        }
    }
}
```

Figura 3.16 – Ligar Relés 1 e 2.

Quando a porta não for mais utilizada, basta desconectá-la. As linhas de código utilizadas para essa função então exibidas na Figura 3.17.

```
private void btnDesconectar_Click(object sender, EventArgs e)
{
    portaCom.Close();
    button1.Enabled = true;
    btnDesconectar.Enabled = false;
}
```

Figura 3.17 – Desconectar Porta COMx.

3.4 - Testes e resultados

Para testar a placa HOMEBEE serão executados alguns testes com o software RS232HomeBee. Selecionamos a porta onde está conectada a CON-USBBEE, escolhemos a velocidade, Bit de dados, a paridade e o bit de parada. A Figura 3.18 mostra a interface do software e quais as configurações para o teste.

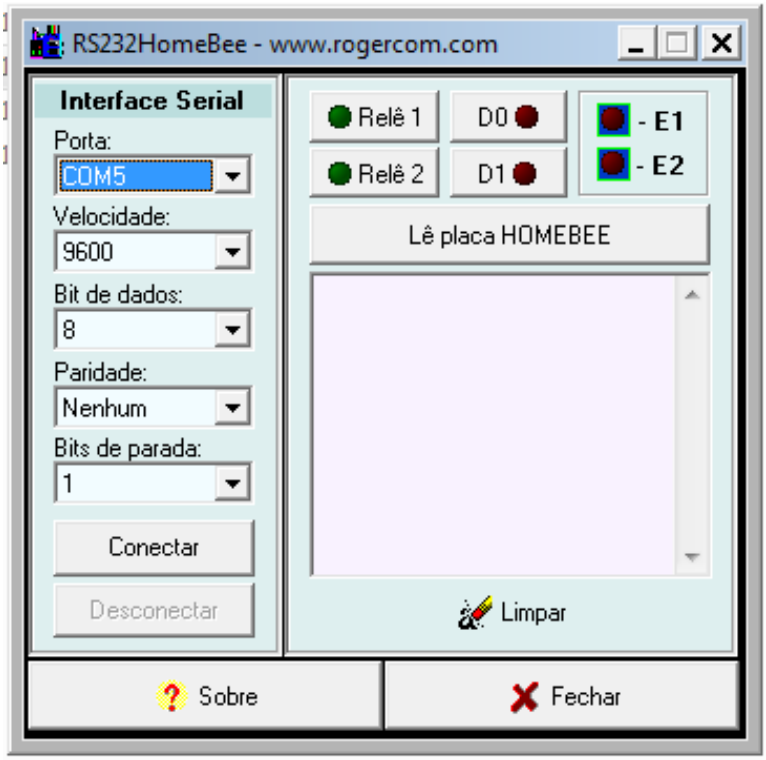


Figura 3.18 – RS232HomeBee

Para testar a HOMEBEE serão executadas algumas ações, que são passadas a seguir.

Conecte a porta e selecione qual relé quer testar.

Figura 3.19 – selecionando relé 1

Feito isso, o LED correspondente ao relé 1 será aceso.

Figura 3.20 – LED 1 aceso

Faça o mesmo para o relé 2.

Figura 3.21 – selecionando relé 2

Figura 3.22 – LED 2 aceso

Pronto, a placa HOMEBEE está funcionando.

Com a placa funcionando, pode-se montar o restante do protótipo, que são as lâmpadas e os fios que as ligaram na carga. As imagens abaixo mostram as lâmpadas prontas.

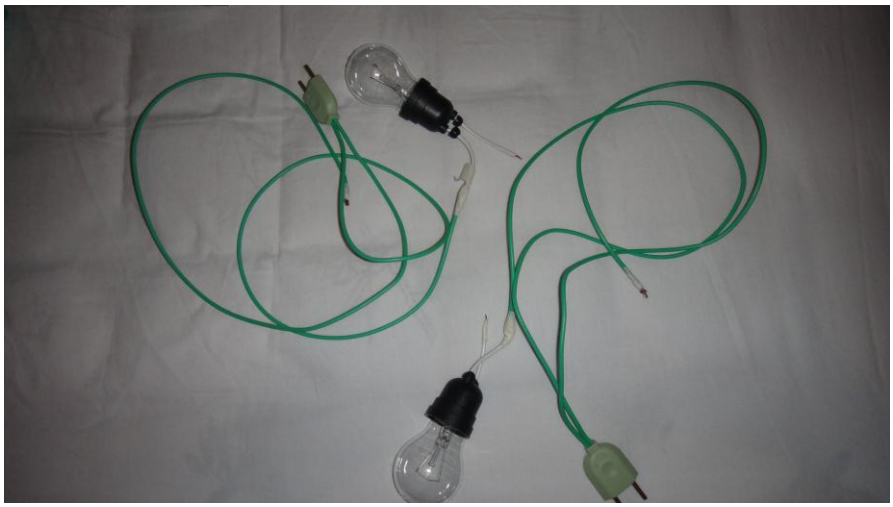


Figura 3.23 – Lâmpadas prontas.

Depois de prontas, pode-se testá-las. Para isso usaremos novamente o software RS232HomeBee. As imagens abaixo mostram que foi executado com êxito.

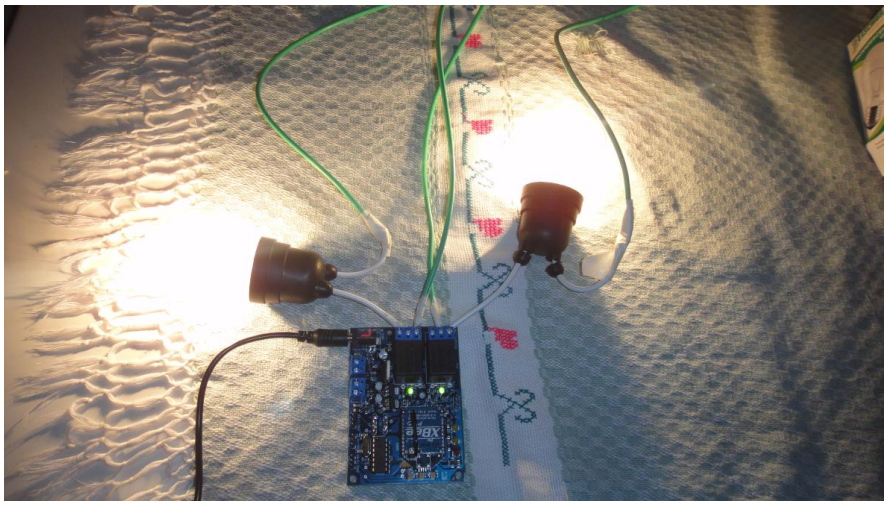


Figura 3.24 – Lâmpadas acesas.

3.5 - Simulação

A simulação do projeto é com um protótipo real. No protótipo, tudo que foi testado será aplicado.

Depois da fase de testes e com todos os dispositivos funcionando, pode-se simular o funcionamento do protótipo. A Figura 3.25 ilustra a interface do software.

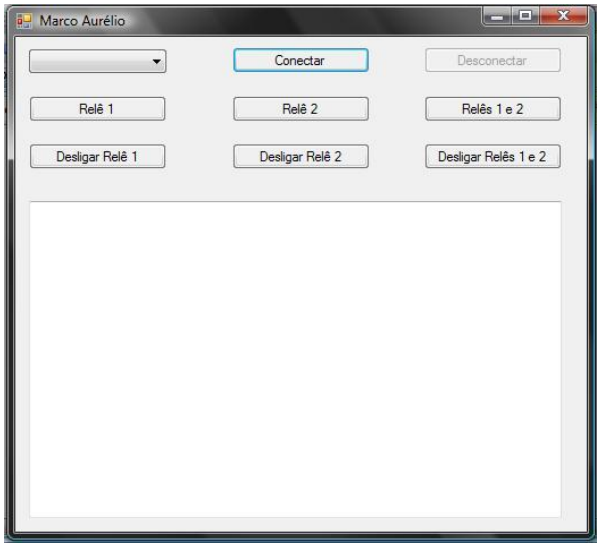


Figura 3.25 – Interface do software

Agora utilizando o software, serão executadas as funções do protótipo.

Com o programa em execução selecione a porta onde está conectada a placa CON-USBBEE. Quando feito isso, receberá uma mensagem de confirmação da ação.

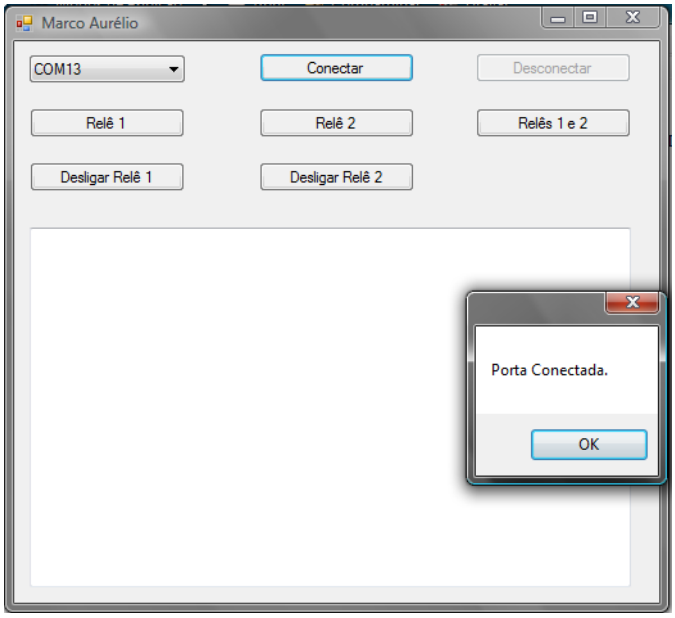


Figura 3.26 – Porta conectada.

Depois de conectar a porta, selecione a ação que deseja executar. O primeiro teste será ligar a lâmpada 1 e desligá-la posteriormente.

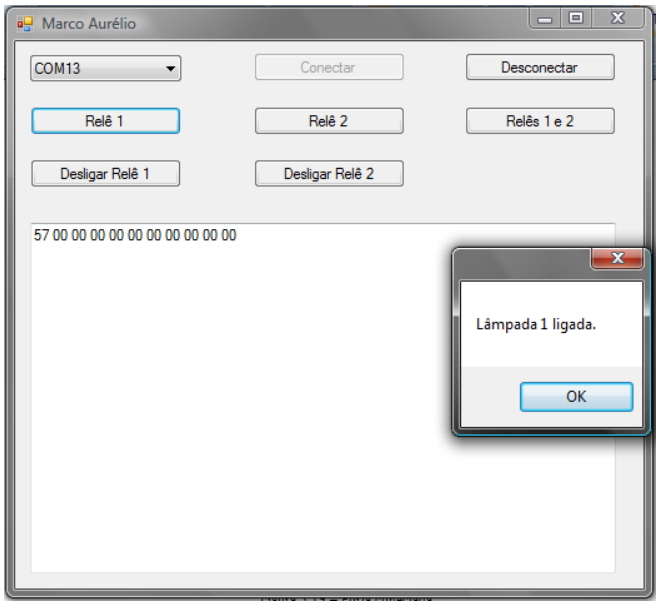


Figura 3.27 – Ligar lâmpada 1.

Na Figura 3.28 é ilustrada a lâmpada 1 acesa. O comando foi enviado pelo software de gerenciamento, como se pode ver na Figura 3.27.

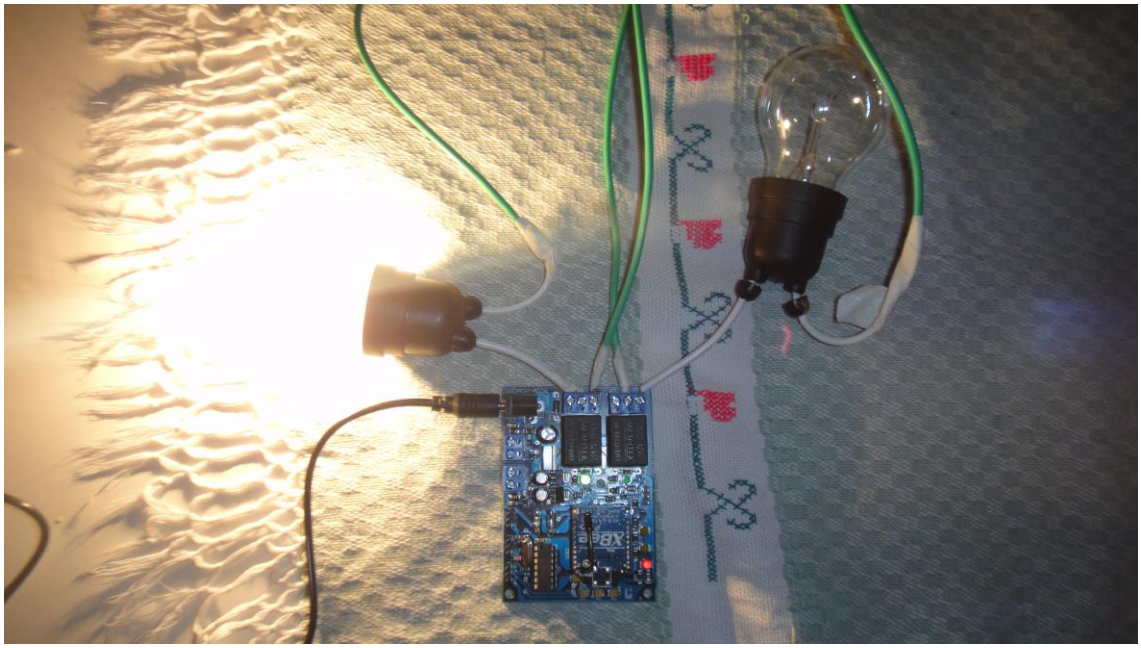


Figura 3.28 – Lâmpada 1 acesa.

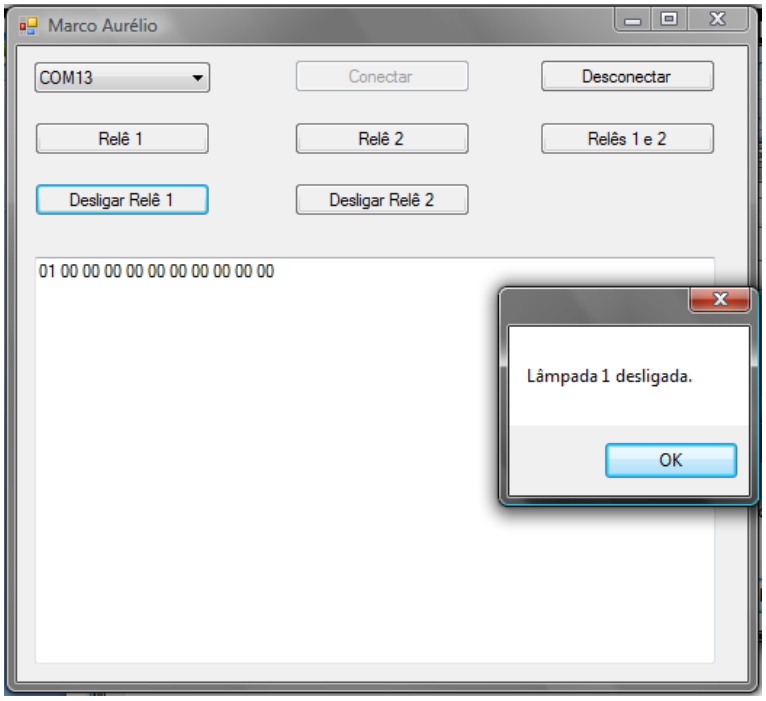


Figura 3.29 – Desligar lâmpada 1..

As Figuras 3.30 e 3.31 ilustram, respectivamente, a ação executada para acender a lâmpada 2 e a lâmpada acesa.

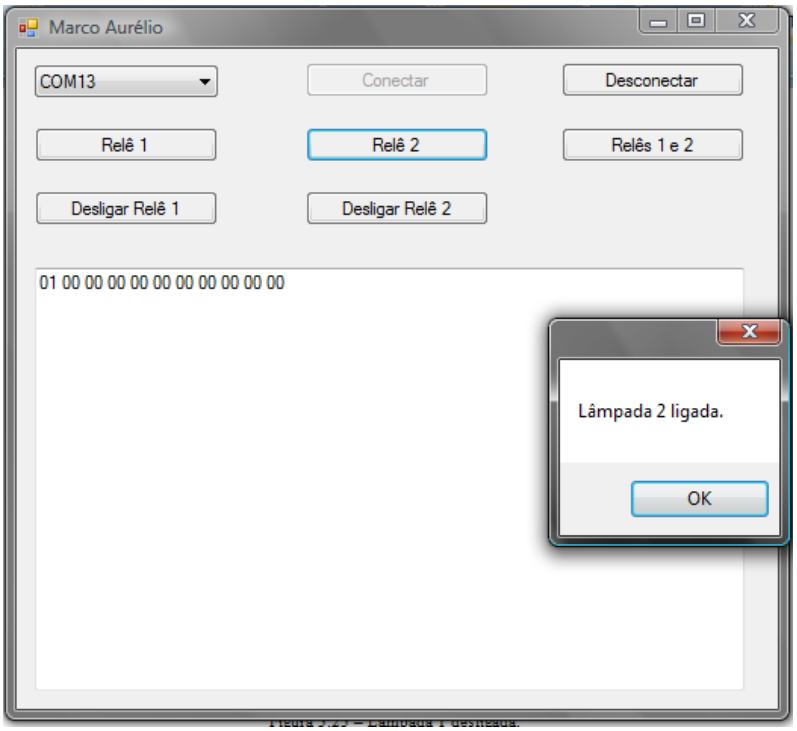


Figura 3.30 – Ligar lâmpada 2.

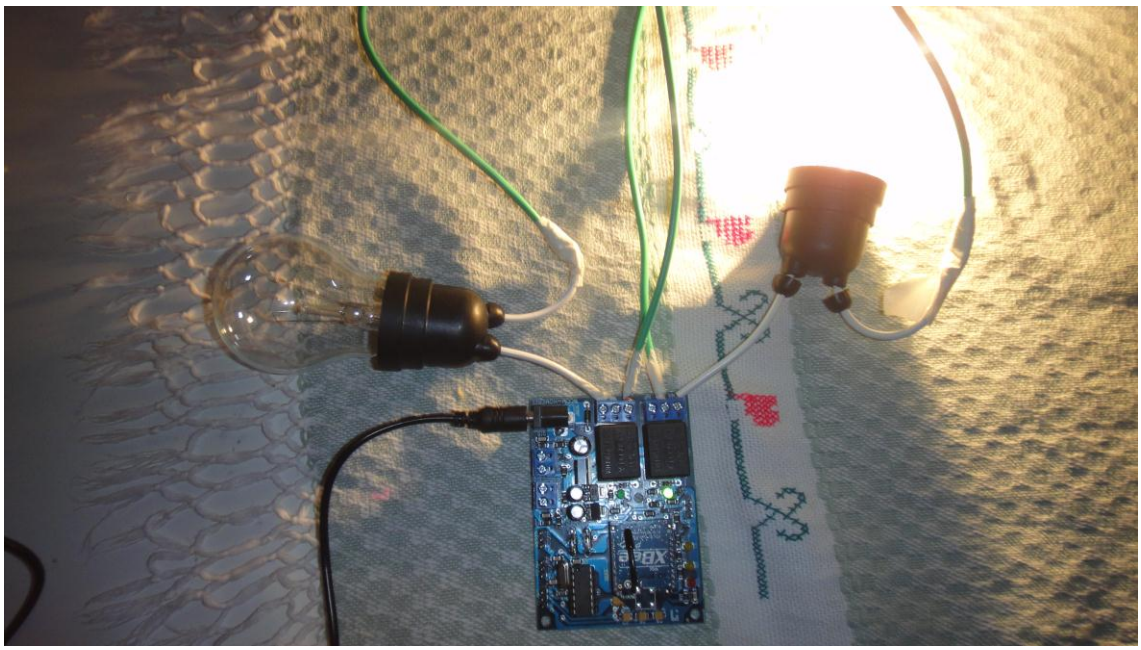


Figura 3.31 – Lâmpada 2 acesa.

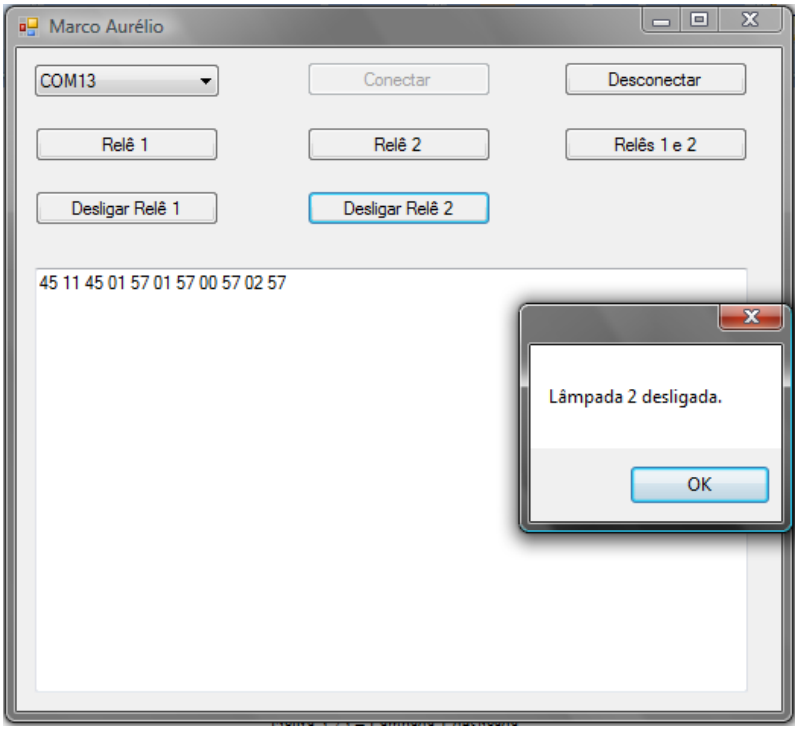


Figura 3.32 – Desligar lâmpada 2.

O próximo teste foi ligar as 2 lâmpadas juntas. As figuras 3.33 e 3.34 ilustram essa ação.

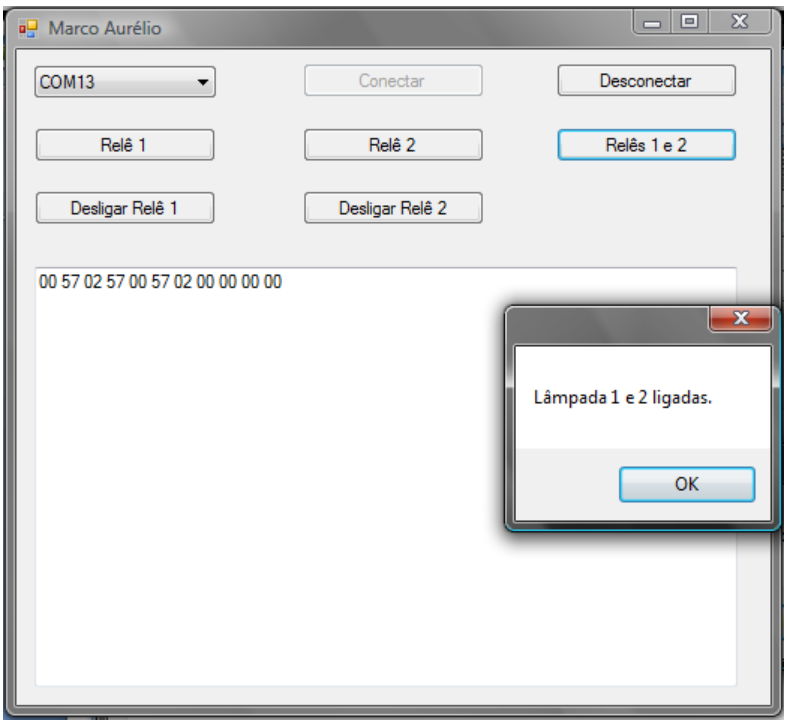


Figura 3.33 – Ligar lâmpadas 1 e 2.

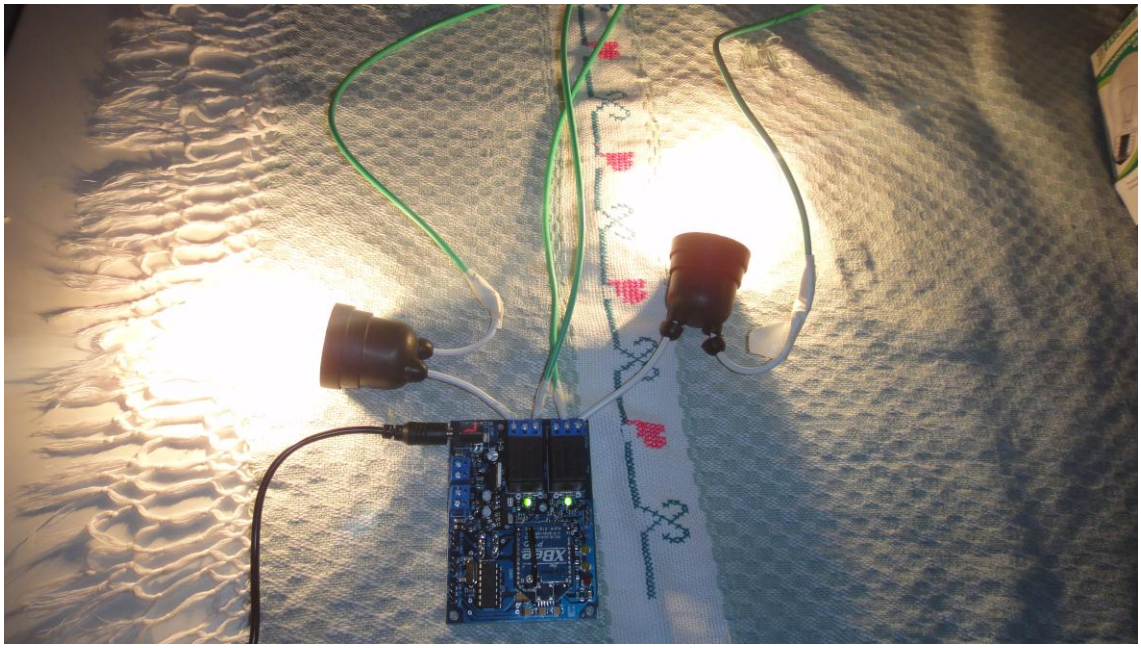


Figura 3.34 – Lâmpadas 1 e 2 acesas.

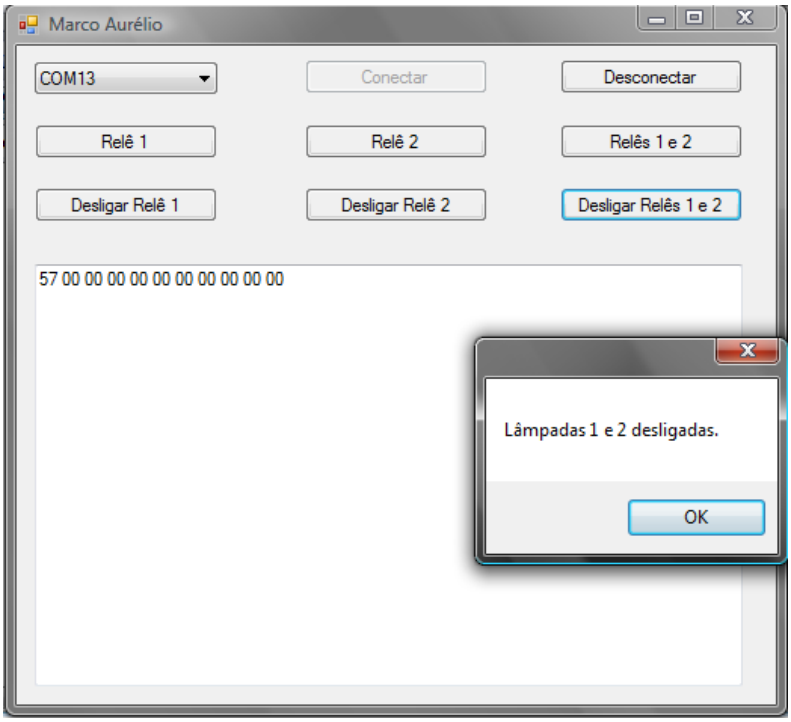
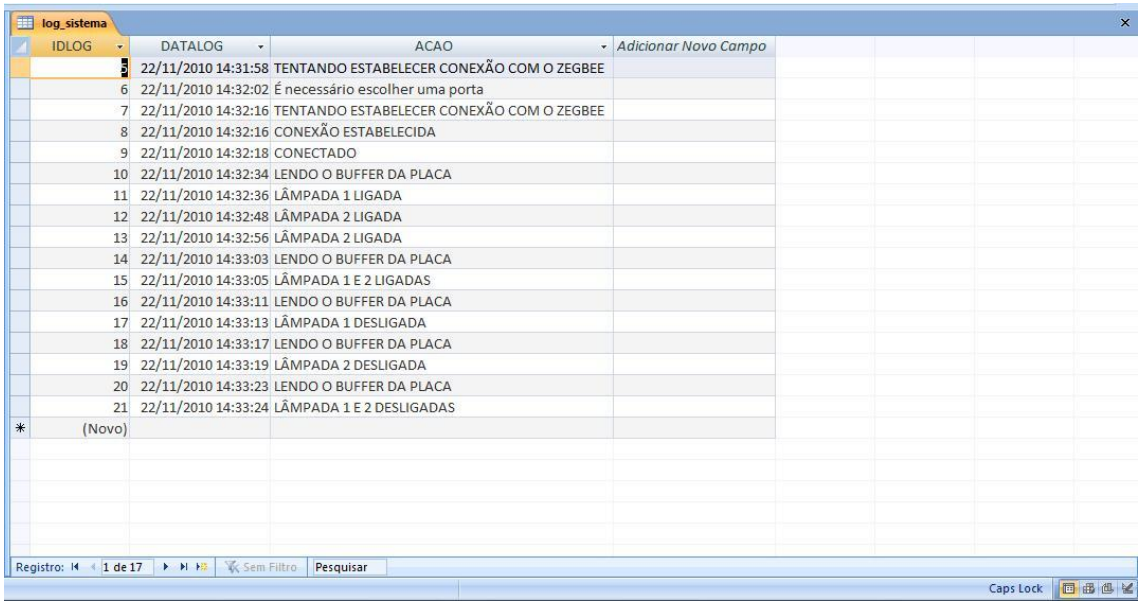


Figura 3.35 – Desligar lâmpadas 1 e 2.

Com os testes executados, pode-se ver o Log do sistema, com as informações das ações executadas, na Figura 3.36.



log_sistema

IDLOG	DATALOG	ACAO	Adicionar Novo Campo
5	22/11/2010 14:31:58	TENTANDO ESTABELECER CONEXÃO COM O ZEGBEE	
6	22/11/2010 14:32:02	É necessário escolher uma porta	
7	22/11/2010 14:32:16	TENTANDO ESTABELECER CONEXÃO COM O ZEGBEE	
8	22/11/2010 14:32:16	CONEXÃO ESTABELECIDA	
9	22/11/2010 14:32:18	CONECTADO	
10	22/11/2010 14:32:34	LENDO O BUFFER DA PLACA	
11	22/11/2010 14:32:36	LÂMPADA 1 LIGADA	
12	22/11/2010 14:32:48	LÂMPADA 2 LIGADA	
13	22/11/2010 14:32:56	LÂMPADA 2 LIGADA	
14	22/11/2010 14:33:03	LENDO O BUFFER DA PLACA	
15	22/11/2010 14:33:05	LÂMPADA 1 E 2 LIGADAS	
16	22/11/2010 14:33:11	LENDO O BUFFER DA PLACA	
17	22/11/2010 14:33:13	LÂMPADA 1 DESLIGADA	
18	22/11/2010 14:33:17	LENDO O BUFFER DA PLACA	
19	22/11/2010 14:33:19	LÂMPADA 2 DESLIGADA	
20	22/11/2010 14:33:23	LENDO O BUFFER DA PLACA	
21	22/11/2010 14:33:24	LÂMPADA 1 E 2 DESLIGADAS	
(Novo)			

Registro: 1 de 17 Sem Filtro Pesquisar

Caps Lock

Figura 3.36 – Log do Sistema

Após a simulação, o protótipo está pronto. Foi testado o software juntamente com as placas e as lâmpadas. Todos funcionam perfeitamente

CAPÍTULO 4 – CONCLUSÃO

4.1 - Conclusões

Este projeto apresenta uma proposta de automação residencial para iluminação utilizando a tecnologia ZigBee. Para isso foi construído um protótipo para simular o seu funcionamento.

O objetivo geral do projeto é auxiliar em uma redução do excessivo consumo de energia no país. Tendo em vista que só existe uma redução desse índice quando são tomadas medidas mais pesadas, como o racionamento de energia. E sabe-se também, que uma parte grande desse consumo é desperdiçada. Com isso, o projeto visa dar ao usuário uma comodidade de acender ou apagar uma lâmpada a distância, evitando um pouco de desperdício.

Todas as tarefas propostas para a realização do trabalho foram executar com sucesso. Primeiro a configuração e os testes das placas USBBEE e HOMEBEE, foi realizado com êxito. A porta foi conectada no software com sucesso, podendo assim executar as funções do protótipo. Por fim, os comandos de ligar e desligar as lâmpadas funcionou perfeitamente.

Os resultados obtidos com o protótipo foram satisfatórios para os objetivos iniciais. A principal vantagem deste proposto é evitar o desperdício de energia e trazer comodidade a usuário. Porém este trabalho se limita a isto não considerando assim outros aspectos, como por exemplo, o controle de outros dispositivos.

O modelo proposto neste trabalho trás varias vantagens e não tem um custo alto. A comunicação ZigBee poderá ser incrementado no futuro trazendo novas soluções sem que para isso precise mudar de tecnologia ou aparelho. Outra vantagem é que podem ser somados novos dispositivos ao protótipo, fazendo apenas algumas configurações. Mostrando assim que seu custo/benefício é considerável valendo o investimento.

4.2 - Sugestões de Trabalhos Futuros

Com os estudos e pesquisas realizados para a execução do projeto, surgiram novas idéias para trabalhos futuros. A comunicação ZigBee está começando a ser bastante estudada e utilizada, tendo em vista as suas vantagens em relação as outras.

Utilizando a comunicação ZigBee, pode-se sugerir alguns trabalhos futuros citados a seguir:

- Utilizar o ZigBee para controlar cortinas, persianas e toldos;
- Utilizar o ZigBee para controle de presença através de sensores;
- Implantar o software de gerenciamento em uma tela de LCD;
- Colocar login e senha para logar no software de gerenciamento, e melhorar sua interface.

REFERÊNCIAS BIBLIOGRÁFICAS

INSTITUTO DE INGENIERÍA ELÉCTRICA (IIE), Automação residencial

Em: iie.fing.edu.uy

Acesso em: 17/08/2010

DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO - UnB, ZigBee

Em: www.cic.unb.br/~bordim/TD/Arquivos/G15_Monografia.pdf

Acesso em: 05/09/2010

PASSARELA, Lucas. ZigBee. Universidade de Brasília, 2006.

INFOESCOLA, Linguagens de Programação

Em: http://www.infoescola.com/informatica/o-que-sao-linguagens-de-programacao/

Acesso em: 27/09/2010

OFICINA DA NET, C# (CSharp)

Em:
http://www.oficinadanet.com.br/artigo/526/c_sharp_csharp_o_que_e_esta_linguagem

Acesso em: 03/10/2010

FRANÇA, Lucas Luiz, CSharp, O que é esta linguagem. Oficina Da Net, 2007.

ROGERCOM, ZigBee

Em: http://www.rogercom.com/ZigBee/ZigBee.htm

Acesso em: 15/09/2010

MCBRIDE, Jan. Padrão ZigBee de comunicação sem fio. DIGI, 2008.

AZEVEDO, Tiago. Roteamento ZigBee. Universidade Federal do Rio de Janeiro, 2006.

ZIGBEE, Automação Residencial com ZigBee

Em: http://www.zigbee.org/Markets/ZigBeeHomeAutomation/FAQ.aspx

Acesso em: 12/09/2010

ZIGBEE, Automação Residencial com ZigBee - Benefícios

Em: http://www.zigbee.org/Markets/ZigBeeHomeAutomation/Benefits.aspx

Acesso em: 12/09/2010

ROGERCOM, Placa CON-USBBEE

Em: www.rogercom.com

Acesso em: 27/07/2010

ROGERCOM, Placa RCON-HOMEBEE

Em: www.rogercom.com

Acesso em: 27/07/2010

MICROSOFT, Microsoft Access

Em: office.microsoft.com/pt-br/training/CR006182940.aspx

Acesso em: 25/09/2010

FUNCTIONX, Microsoft Access

Em: www.functionx.com/access/

Acesso em: 26/09/2010

ESTADÃO, Consumo de Energia

Em: http://www.estadao.com.br/noticias/vidae,consumo-de-energia-no-pais-cresceu-3737-entre-1995-e-2006,183713,0.htm

Acesso em: 22/06/2010

ANEXO A – PROGRAMA PRINCIPAL

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.IO.Ports;
using System.Data.OleDb;
namespace xBeeTeste
{
   public partial class frmPrincipal : Form
   {
      private OleDbConnection _conexao = new
OleDbConnection("Provider=Microsoft.ACE.OLEDB.12.0;Data Source="+
Application.StartupPath + @"\log.accdb;Persist Security Info=False;");
      private SerialPort portaCom = new SerialPort();
      private StringBuilder query = new StringBuilder("INSERT INTO
LOG_SISTEMA(DATALOG, ACAO) VALUES(#{0}#, '{1}')");
      private OleDbCommand comando = new OleDbCommand();
      public frmPrincipal()
      {
         InitializeComponent();
      }

      private void button1_Click(object sender, EventArgs e)
      {
         try
         {
            gravarLog("TENTANDO ESTABELECER CONEXÃO COM O ZEGBEE");
            if (cmbPortas.Text != "")
            {
               portaCom = new SerialPort(cmbPortas.SelectedItem.ToString(), 9600,
Parity.None, 8, StopBits.One);

               portaCom.Open();
               gravarLog("CONEXÃO ESTABELECIDA");
               MessageBox.Show("Porta Conectada.");
            }
            else
            {
               MessageBox.Show("É necessário escolher uma porta");
               gravarLog("É necessário escolher uma porta");
               return;
            }
```

```
            button1.Enabled = false;
            btnDesconectar.Enabled = true;
            gravarLog("CONECTADO");
        }
        catch (Exception ex)
        {
            gravarLog(ex.Message);
            MessageBox.Show(ex.Message + "\r\n" + "Porta " +
cmbPortas.SelectedItem.ToString() + " esta aberta");
        }

    }

    private void gravarLog(string p)
    {
        try
        {
            if(_conexao.State == ConnectionState.Open){
                _conexao.Close();
            }
            _conexao.Open();

            comando.CommandText = String.Format(query.ToString(),
DateTime.Now.ToString(), p);
            comando.CommandType = CommandType.Text;
            comando.Connection = _conexao;

            if (comando.ExecuteNonQuery() > 0)
            {
                lblInfo.Text = "LOG GRAVADO.";
            }
        }
        catch (Exception ex)
        {
            gravarLog(ex.Message);
            MessageBox.Show(ex.Message);
        }
    }

    private void carregarPortas()
    {
        try
        {
            cmbPortas.Items.Clear();

            foreach (string s in SerialPort.GetPortNames())
            {
                cmbPortas.Items.Add(s);
            }
```

```
    }
    catch (Exception ex)
    {
        gravarLog(ex.Message);
        MessageBox.Show(ex.Message);
    }
}

private void frmPrincipal_Load(object sender, EventArgs e)
{
    try
    {
        carregarPortas();
        btnDesconectar.Enabled = false;

    }
    catch (Exception ex)
    {
        gravarLog(ex.Message);
        MessageBox.Show(ex.Message);
    }
}

private void btnOnRele1_Click(object sender, EventArgs e)
{
    try
    {
        byte[] rele1 = new byte[11];
        //7E 00 07 81 50 01 24 00 57 01 B1
        //7E 00 07 81 50 01 28 00 57 01 AD
        rele1[0] = 0x7E;
        rele1[1] = 0x00;
        rele1[2] = 0x07;
        rele1[3] = 0x01;
        rele1[4] = 0x01;
        rele1[5] = 0x50;
        rele1[6] = 0x01;
        rele1[7] = 0x00;
        rele1[8] = 0x7B;
        rele1[9] = 0x01;
        rele1[10] = 0x30;

        textBox1.Text = rele1.ToString() + "\r\n";
        portaCom.Write(rele1, 0, rele1.Length);

        lerPlaca();
        MessageBox.Show("Lâmpada 1 ligada.");
        gravarLog("LÂMPADA 1 LIGADA");
    }
    catch (Exception ex)
```

```
    {
        gravarLog(ex.Message);
        MessageBox.Show(ex.Message);
    }
}

private void lerPlaca()
{
    byte[] buffer = new byte[11];
    portaCom.Read(buffer, 0, buffer.Length);

    textBox1.Text = ByteArrayToHexString(buffer);
    gravarLog("LENDO O BUFFER DA PLACA");
}

private void btnOnRele2_Click(object sender, EventArgs e)
{
    try
    {
        byte[] rele2 = new byte[11];
        //7E 00 07 01 01 50 01 00 7B 02 1F
        rele2[0] = 0x7B;
        rele2[1] = 0x00;
        rele2[2] = 0x07;
        rele2[3] = 0x01;
        rele2[4] = 0x01;
        rele2[5] = 0x50;
        rele2[6] = 0x01;
        rele2[7] = 0x00;
        rele2[8] = 0x7b;
        rele2[9] = 0x02;
        rele2[10] = 0x1f;

        portaCom.Write(rele2, 0, rele2.Length);
        MessageBox.Show("Lâmpada 2 ligada.");
        gravarLog("LÂMPADA 2 LIGADA");
    }
    catch (Exception ex)
    {
        gravarLog(ex.Message);
        MessageBox.Show(ex.Message);
    }
}

private void btnOffRele1_Click(object sender, EventArgs e)
{
    try
    {
        byte[] rele1 = new byte[11];
        rele1[0] = 0x7E;
```

```
        rele1[1] = 0x00;
        rele1[2] = 0x07;
        rele1[3] = 0x01;
        rele1[4] = 0x01;
        rele1[5] = 0x50;
        rele1[6] = 0x01;
        rele1[7] = 0x00;
        rele1[8] = 0x7B;
        rele1[9] = 0x00;
        rele1[10] = 0x30;

        portaCom.Write(rele1, 0,rele1.Length);
        lerPlaca();
        MessageBox.Show("Lâmpada 1 desligada.");

        gravarLog("LÂMPADA 1 DESLIGADA");
    }
    catch (Exception ex)
    {
        gravarLog(ex.Message);
        MessageBox.Show(ex.Message);
    }
}

private void frmPrincipal_FormClosed(object sender, FormClosedEventArgs e)
{
    portaCom.Close();
}


private string ByteArrayToHexString(byte[] data)
{

    StringBuilder sb = new StringBuilder(data.Length * 3);

    foreach (byte b in data)

        sb.Append(Convert.ToString(b, 16).PadLeft(2, '0').PadRight(3, ' '));

    return sb.ToString().ToUpper();

}

//Convert Byte Array To Ascii String

private string ByteArrayToAsciiString(byte[] data)
{

    StringBuilder sb = new StringBuilder(data.Length * 3);
```

```
    foreach (byte b in data)

        sb.Append(Convert.ToChar(b));

    return sb.ToString().ToUpper();

}

private void btnDesconectar_Click(object sender, EventArgs e)
{
    portaCom.Close();
    button1.Enabled = true;
    btnDesconectar.Enabled = false;
    gravarLog("CONEXÃO ENCERRADA.");
}

private void btnOffRele2_Click(object sender, EventArgs e)
{
    try
    {
        byte[] rele2 = new byte[11];
        rele2[0] = 0x7B;
        rele2[1] = 0x00;
        rele2[2] = 0x07;
        rele2[3] = 0x01;
        rele2[4] = 0x01;
        rele2[5] = 0x50;
        rele2[6] = 0x01;
        rele2[7] = 0x00;
        rele2[8] = 0x7b;
        rele2[9] = 0x00;
        rele2[10] = 0x1f;

        portaCom.Write(rele2, 0, rele2.Length);
        lerPlaca();
        MessageBox.Show("Lâmpada 2 desligada.");
        gravarLog("LÂMPADA 2 DESLIGADA");
    }
    catch (Exception ex)
    {
        gravarLog(ex.Message);
        MessageBox.Show(ex.Message);
    }
}

private void button3_Click(object sender, EventArgs e)
{
    {
        try
        {
```

```
            byte[] rele12 = new byte[11];
            rele12[0] = 0x7B;
            rele12[1] = 0x00;
            rele12[2] = 0x07;
            rele12[3] = 0x01;
            rele12[4] = 0x01;
            rele12[5] = 0x50;
            rele12[6] = 0x01;
            rele12[7] = 0x00;
            rele12[8] = 0x7b;
            rele12[9] = 0x03;
            rele12[10] = 0x2e;

            portaCom.Write(rele12, 0, rele12.Length);
            lerPlaca();
            MessageBox.Show("Lâmpada 1 e 2 ligadas.");
            gravarLog("LÂMPADA 1 E 2 LIGADAS");
        }
        catch (Exception ex)
        {
            gravarLog(ex.Message);
            MessageBox.Show(ex.Message);
        }
    }
}

private void textBox1_TextChanged(object sender, EventArgs e)
{

}

private void cmbPortas_SelectedIndexChanged(object sender, EventArgs e)
{

}

private void button2_Click(object sender, EventArgs e)
{
    try
    {
        byte[] rele12 = new byte[11];
        rele12[0] = 0x7B;
        rele12[1] = 0x00;
        rele12[2] = 0x07;
        rele12[3] = 0x01;
        rele12[4] = 0x01;
        rele12[5] = 0x50;
        rele12[6] = 0x01;
        rele12[7] = 0x00;
        rele12[8] = 0x7b;
```

```
            rele12[9] = 0x00;
            rele12[10] = 0x1f;

            portaCom.Write(rele12, 0, rele12.Length);
            lerPlaca();
            MessageBox.Show("Lâmpadas 1 e 2 desligadas.");

            gravarLog("LÂMPADA 1 E 2 DESLIGADAS");
         }
         catch (Exception ex)
         {
            gravarLog(ex.Message);
            MessageBox.Show(ex.Message);
         }
      }
   }
}
```