RAFAEL AUGUSTO MORENO GONÇALVES

DISPOSITIVO PARA CONTROLE DE ELETRÔNICOS POR MEIO DE INFRAVERMELHO A PARTIR DE UMA INTERFACE DE REDE SEM FIO

São Paulo
2009

RAFAEL AUGUSTO MORENO GONÇALVES

DISPOSITIVO PARA CONTROLE DE ELETRÔNICOS POR MEIO DE INFRAVERMELHO A PARTIR DE UMA INTERFACE DE REDE SEM FIO

Dissertação apresentada à Escola Politécnica da Universidade de São Paulo para obtenção do título de Bacharel em Engenharia.

Área de Concentração:
Engenharia Mecatrônica

Orientador: Prof. Doutor Marcos Ribeiro Pereira Barretto

São Paulo
2009

FICHA CATALOGRÁFICA

Gonçalves, Rafael Augusto Moreno
Dispositivo para controle de eletrônicos por meio de infravermelho a partir de uma interface de rede sem fio / R.A.M. Gonçalves. -- São Paulo, 2009.
65 p.

Trabalho de Formatura - Escola Politécnica da Universidade de São Paulo. Departamento de Engenharia Mecatrônica e de Sistemas Mecânicos.

1. Infravermelho (Protótipo) 2. Redes de computadores 3. Programação orientada a objetos I. Universidade de São Paulo. Escola Politécnica. Departamento de Engenharia Mecatrônica e de Sistemas Mecânicos II. t.

Aos meus pais e amigos

RESUMO

Neste trabalho é desenvolvido um protótipo capaz de, emulando controles remotos via infravermelho, mapeados com auxílio de um osciloscópio, comandar eletrônicos remotamente por meio de uma interface implementada em Python e Django em um computador pessoal. O dispositivo fabricado utiliza tecnologia Zigbee, por meio de módulos XBee para comunicar-se com o computador, o qual disponibiliza suas funcionalidades na forma de um website. Por fim, mapeia-se o controle de um rádio Philips modelo flat em um arquivo XML estruturado e comenta-se seu desempenho, o qual é perfeitamente condizente com o proposto. Conclui-se que a partir do protótipo desenvolvido é possível incorporar diversas características interessantes ao sistema, resultando em um dispositivo de maior atratividade comercial.

Palavras-chave: Automação residencial. Domótica. Infravermelho. Zigbee. Miwi. Arduino. Controle remoto. Python. Django. XML.

ABSTRACT

Development of a prototype that emulates an infrared remote control (previously mapped using an oscilloscope) and commands electronics remotely trough a web based interface implemented using Python and Django on an ordinary PC. The prototype uses Zigbee technology (XBee modules) to communicate with a PC based server. Finally, it presents the mapping of a Phlips flat radio remote control into a XML file and some comments about the performance of the system. It concludes that with some minor software changes it's possible to add a series of interesting features to the prototype, rendering it into a commercial product.

Keywords: Home automation. Infrared. Zigbee. Miwi. Arduino. Remote control. Python. Django. XML.

Lista de Figuras

Lista de Tabelas

Sumário

1. Introdução

1.1. Motivação

Uma vez que o nível tecnológico e o poder aquisitivo avançam, o número de dispositivos presentes no ambiente, seja em casa ou no trabalho, aumenta sensivelmente. Televisores, projetores, sistemas de áudio e vídeo, computadores, celulares e diversos periféricos estão presentes na vida do homem moderno e o desejo de integrá-los torna-se crescente, dando origem ao promissor campo da domótica, a automação residencial, em especial ao setor ligado ao entretenimento. Embora os serviços de uma empresa especializada sejam ainda muito onerosos, um passo nessa direção já tornou-se bastante comum: a existência de pequenas redes domésticas e comerciais a fim de compartilhar recursos.

Enquanto há poucos anos estas eram dependentes de cabeamento, hoje a possibilidade de acesso a redes sem-fio é uma realidade consolidada. Celulares, computadores portáteis e de mesa da geração atual já vêm preparados para este tipo de integração, fazendo com que a escolha por essa topologia seja imediata para redes residenciais e de pequenos escritórios. A implementação destas dá-se usualmente com a centralização em um roteador (*router*) possibilitando acessar recursos ligados a rede sem que algum computador específico tenha que permanecer ligado. Ficam excluídos porém os dispositivos incapazes de comunicar-se através do protocolo de rede escolhido (usualmente TCP/IP).

Neste trabalho de conclusão serão analisadas arquiteturas para permitir o controle de tais aparelhos – televisores, rádios, projetores e outros – através de um dispositivo capaz de integrar-se a uma rede TCP / IP e emitir sinais via infravermelho. Após a escolha de uma destas arquiteturas, serão apresentadas os requisitos do produto e discutidas as diferenças básicas entre o desenvolvimento de um produto e um protótipo. A última parte é dedicada ao projeto do protótipo e sua fabricação.

1.2. Objetivos

O objetivo principal deste trabalho é o projeto de dispositivos capazes de permitir o controle de aparelhos dotados de controle remoto via infravermelho por meio de uma rede TCP / IP e posterior fabricação de um protótipo. As fases de projeto incluem a especificação dos requisitos do sistema, do hardware e do software necessários.

Os objetivos secundários contemplam o aprendizado da metodologia de um projeto prático em tecnologia aplicado, incluindo a estruturação formal deste documento.

1.3. Organização do Trabalho

O trabalho é estruturado da seguinte forma:

- **Capítulo 2:** Análise das alternativas propostas para solução do problema. Arquiteturas envolvendo caracterização de hardware e topologia de rede, e posterior escolha através de uma matriz de decisão; protocolos de comunicação sem fio e interface homem-máquina.
- **Capítulo 3:** Apresentação dos requisitos funcionais, técnicos e financeiros;
- **Capítulo 4:** Projeto básico envolvendo a escolha dos microcontroladores para o módulo de infravermelho e software para todos os componentes do sistema;
- **Capítulo 5:** Projeto do módulo, fabricação e implementação de software;
- **Capítulo 6:** Projeto e implementação do software do servidor;
- **Capítulo 7:** Instruções de uso e interface implementada;
- **Capítulo 8:** Considerações finais sobre o desempenho e conclusão;
- **Capítulo 9:** Bibliografia;
- **Anexo 1:** Apresentação de considerações básicas sobre infravermelho e o funcionamento dos controles remotos. Detalhamento dos protocolos Philips RC5x e RC6.

2. Análise de Alternativas

2.1. Arquitetura

2.1.1. All-in-One

Proposta imediata, a arquitetura *All-in-One* pode ser entendida como o projeto e fabricação de um módulo especializado, capaz de comunicar-se com a rede sem-fio, dispor as funções de roteador, servir a interface de controle de IR e enviar os sinais. O fluxo de dados pode ser ilustrado como se segue:

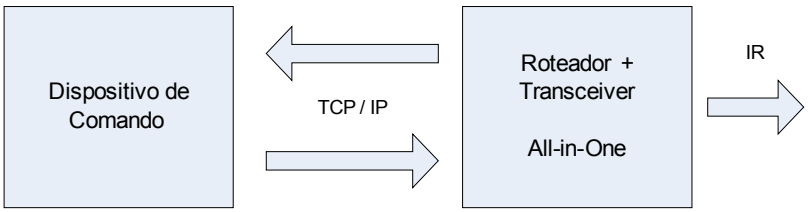

Figura 1: Fluxo de Dados na Arquitetura All-in-One

2.1.2. Módulo Escravo

Uma segunda arquitetura divide o sistema em dois módulos, o roteador e o transceiver infravermelho sendo o fluxo de sinais esquematizado na figura abaixo:

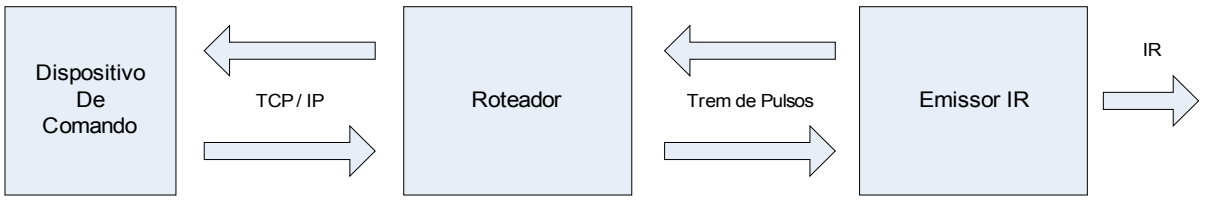

Figura 2: Fluxo de Dados na Arquitetura de Módulo Escravo

Um módulo escravo possui apenas a capacidade de enviar e receber sinais infravermelho, tendo uma porta serial de entrada e outra de saída ligadas diretamente ao roteador, o qual é responsável pela lógica de negócio e processamento do sinal. A escolha desta arquitetura, implica, portanto na reprogramação do firmware – software presente no roteador, responsável por suas funções – para que seja capaz de gerar

e reconhecer o trem de pulsos na comunicação. Além disso deve-se programar a API e a interface de usuário também no firmware, uma vez que a o “protocolo” entendido pelo transceiver é de baixíssimo nível.

Embora tal arquitetura acarrete o menor custo de hardware possível – um LED infravermelho, um resistor e um capacitor, no caso de termos apenas um transmissor – ela incorre em um problema sério ao vincular o projeto a apenas um modelo (ou uma família) de roteadores uma vez que o firmware não será compatível com outras marcas e modelos.

2.1.3. Módulo NA Inteligente

Levando a divisão além do nível de hardware, pode-se implementar uma separação funcional no nível lógico. Nesta arquitetura um microcontrolador com stack TCP/IP compacto é embarcado no módulo transceiver IR, possibilitando a comunicação através deste protocolo com o roteador. A figura abaixo ilustra o fluxo de dados:

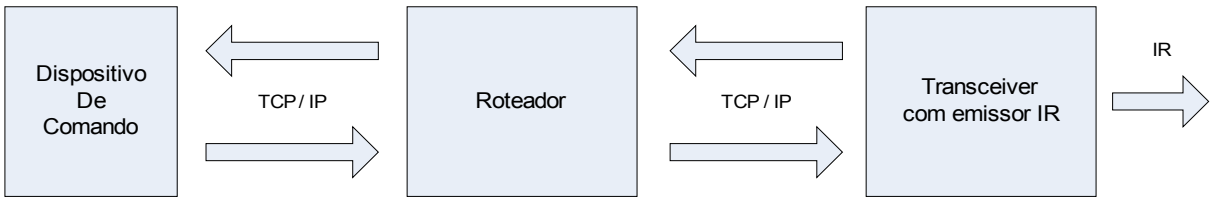

Figura 3: Fluxo de Dados na Arquitetura de Módulo NA Inteligente

Importante observar que ao implementar essa arquitetura nenhuma mudança será necessária no roteador, em verdade qualquer dispositivo capaz de integrar uma rede TCP / IP e conectar-se ao transceiver através de um conector 8P8C (padrão em rede par trançado, também conhecido por RJ45) poderá acessar ou disponibilizar as funções deste.

Outra característica desta arquitetura é possibilitar três produtos diferentes, a saber:

1. **Transmissor de Infravermelho:** Trata-se apenas do transceiver capaz de conectar-se a uma rede TCP / IP. Pode ser ligado diretamente a um computador,

hub, switch, roteador ou access point previamente existente.

2. **Access Point com Infravermelho:** Utilizando-se de um AP capaz de operar em Structure Mode (modo que permite a interligação de uma rede Ethernet com uma WiFi), pode-se adaptar um transmissor na porta disponível e projetar uma nova caixa para os circuitos. Pode-se assim criar produtos interessantes como, por exemplo, um bulbo de teto.
3. **Roteador com Infravermelho:** Um procedimento similar ao anterior pode ser aplicado a um roteador utilizando uma de suas portas para Ethernet. Todas as funcionalidades seriam mantidas.

2.1.4. Módulos em Rede Secundária

Embora a solução anterior demonstre-se razoável em termos de obtenção de produtos, em verdade a utilização do circuito interno de um roteador ou de um access point traz problemas relacionados a manutenção da garantia destes e também agrega um custo elevado aos módulos.

Uma alternativa que proporciona a utilização de diversos módulos, baixo custo e mantém possibilidades interessantes de uma estrutura de rede para a comunicação entre eles é a utilização de uma rede sem fio secundária, implementando um protocolo simples que possa ser implementada sem grandes recursos de hardware. Nesta arquitetura um módulo, denominado base, estabelece essa rede e também conecta-se a rede TCP/IP original, realizando a ponte entre ambas.

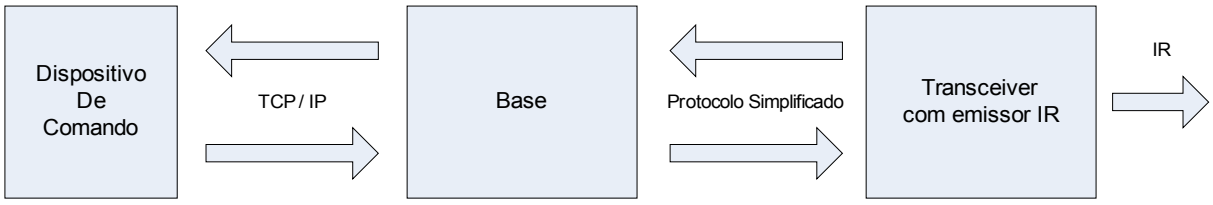

Figura 4: Fluxo de Dados na Arquitetura de Módulos em Rede Secundária

Uma característica desta arquitetura é a comunicação de duas vias entre o módulo e a base que, embora não apresente uso prático no nosso projeto, possibilita a criação de sensores e atuadores capazes de informar seu estado à base.

2.1.5. Módulo Atuador com Lógica de Negócio em Servidor

Tal configuração, demonstrada na figura abaixo, centraliza a lógica de negócio, os controles, processamento de comandos, interfaces e gerenciamento de usuários em um PC utilizado como servidor, permitindo flexibilidade no desenvolvimento do software e facilitando a implementação por possibilitar a escolha por soluções consagradas, como Apache, servidor web, e Python, linguagem de programação interpretada com orientação a objetos.

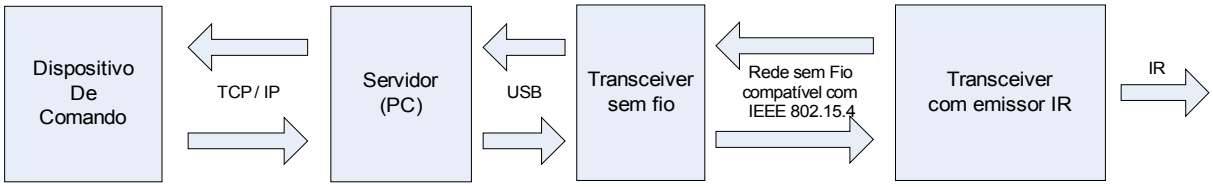

Figura 5: Fluxo de dados na arquitetura de Módulo Atuador com Lógica de Negócio em Servidor

A arquitetura é interessante por permitir também facilidade e capacidade de expansão ao utilizar os recursos computacionais de um PC, ainda que de custo reduzido, as possibilidades de automação são ilimitadas e a interação entre diversos periféricos torna-se simples desenvolvimento de software.

2.1.6. Matriz de Decisão

Considerando as quatro arquiteturas propostas para o dispositivo é imperativo fazer a opção por uma delas para a continuidade do processo. Tal situação é extremamente comum nos projetos de engenharia sendo que dispõe-se de diversos métodos para tal decisão, dentre eles o mais comum talvez seja a matriz de decisão. A apostila de PNV2100, Introdução à Engenharia, a define no seguinte formato:

> *Uma forma relativamente simples de implementar o procedimento acima esboçado é o que se chama de Matriz de Decisão. Consiste em selecionar a melhor alternativa pela determinação da maior média ponderada das notas. Assim, colocamos em uma tabela, de um lado, as soluções propostas, por exemplo ao longo da primeira linha. Na primeira coluna listamos os critérios de avaliação. Pode-se reservar a 2a. coluna para atribuir os pesos atribuídos aos critérios. Os demais espaços da tabela são*

utilizados para a atribuição de notas a cada uma das alternativas segundo os critérios adotados.

Tomando as arquiteturas pelas soluções propostas resta determinarmos os critérios de avaliação e respectivos pesos, tal processo dá-se, contudo, de maneira majoritariamente subjetiva. Abaixo os escolhidos são apresentados.

1. Custos de Produção
2. Disponibilidade Software, Sistema de Desenvolvimento e Suporte
3. Facilidade de Produção
4. Fornecedores Estabelecidos
5. Suporte a Módulos Adicionais
6. Possibilidade de expansão

		Arquitetura				
Critérios	**Pesos**	**I**	**II**	**III**	**IV**	**V**
I	9	1	10	8	6	4
II	7	4	4	6	7	9
III	6	2	5	6	5	9
IV	9	6	2	6	9	10
V	8	0	4	8	10	10
IV	9	0	0	6	7	10
	Total	103	198	322	357	413

Tabela 1: Matriz de Decisão para Arquiteturas

Como esperado, dados os prós e contras apresentados e ponderados neste processo, a escolha de engenharia é pelo *Módulo Atuador com Lógica de Negócio em Servidor*. Uma segunda escolha, por módulos em rede secundária ainda seria possível, dado que uma pontuação 13,6% inferior é aceitável, porém dadas as complicações no desenvolvimento (refletida no critério II), produção (III) e as dificuldades de expansão (IV) manter-se-á a primeira escolha.

2.2. Comunicação sem Fio

Uma vez que a arquitetura escolhida contempla a utilização de uma rede sem fio simplificada para conectar o PC-Servidor aos módulos, é necessário analisar, dentre as duas opções líder de mercado, qual é mais interessante ao desenvolvimento.

2.2.1. ZigBee

ZigBee é um protocolo de rede sem fio desenvolvido para sensores, atuadores e redes de controle que necessitem de baixa taxa de transmissão de dados e baixo custo de implementação. Aplicações como redes de automação predial, sistemas de segurança domésticos, controle industrial, sensoriamento remoto e periféricos.

Comparado a outros protocolos, o ZigBee se caracteriza por sua baixa complexidade, facilidade de implementação e hardware simples. Oferece três frequências de operação – 868 MHz na Europa, 915 MHz nos USA e Austrália, e 2.4 GHz em todo o mundo – além de uma série de capacidades de segurança e suporte a diversas topologias.

O protocolo ZigBee baseia-se na IEEE 802.15.4-2003, sendo sua especificação mantida e publicada pela ZigBee Aliance. Para a distribuição comercial de um produto utilizando o protocolo, deve-se ser um associado de tal órgão, certificando-se como ZigBee Adopter, pagando anualidade no valor de US$ 3500,00 quando da data de redação deste documento.

Stacks capazes de implementar tal protocolo são fornecidos pelos fabricantes de microcontroladores: Atmel e Microchip. Atualmente os seguintes aspectos são suportados:

- Operação a 2.4 GHz (ambos) e demais frequências do protocolo (Atmel);
- Diferentes tipos de dispositivos: roteadores, coordenadores e clientes;
- Biblioteca de funções padrão implementando rotinas definidas na norma;

Adicionalmente a Microchip oferece:

- Portabilidade em toda a família PIC18;

- Suporte *out-of-box* para o compilador MPLAB® C18

Algumas limitações também são associadas a este stack:

- Não há suporte a redes com passagem de *token* ou *beacon*;
- Endereços de rede de dispositivos que desconectem não podem ser realocados;
- Remoção automática de nós da tabela de vizinhos não implementada;
- Resolução de conflitos para PAN ID não disponível.

A taxa máxima de transferência é ditada pela frequência de operação: 2,4 GHz resulta em 250 kbps, 915 Mhz em 40 kbps e 868 Mhz em 20 kbps.

A especificação IEEE 802.15.4 define que em uma rede pode-se ter dispositivos de funções completas (FFD) – os quais podem desempenhar qualquer papel: roteadores, coordenadores ou clientes – e de funções reduzidas (RFD), os quais só podem desempenhar o papel de clientes. As topologias suportadas são: estrela, cluster e mesh.

- **Estrela:** Há apenas um roteador/servidor responsável pela manutenção da rede e diversos clientes conectados a ele. Os clientes não comunicam-se entre si, todas as mensagens passam pelo centralizador;

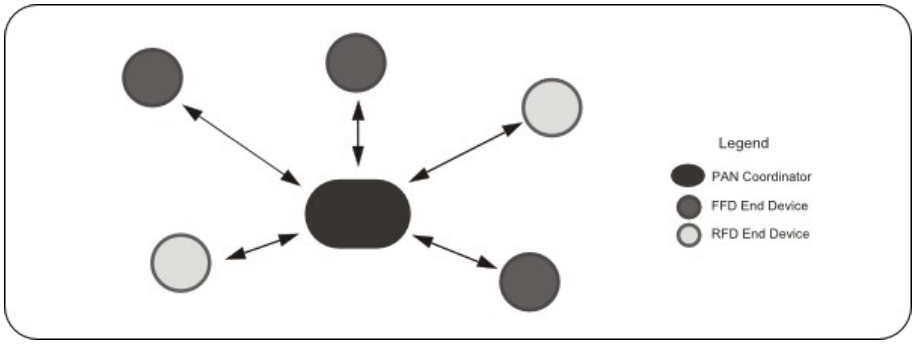

Figura 6: Topologia estrela

- **Cluster:** A rede possui além do roteador e dos clientes um ou mais coordenadores, aos quais os clientes podem conectar-se e que são responsáveis pela árvore abaixo de si. Os clientes não podem comunicar-se entre si;

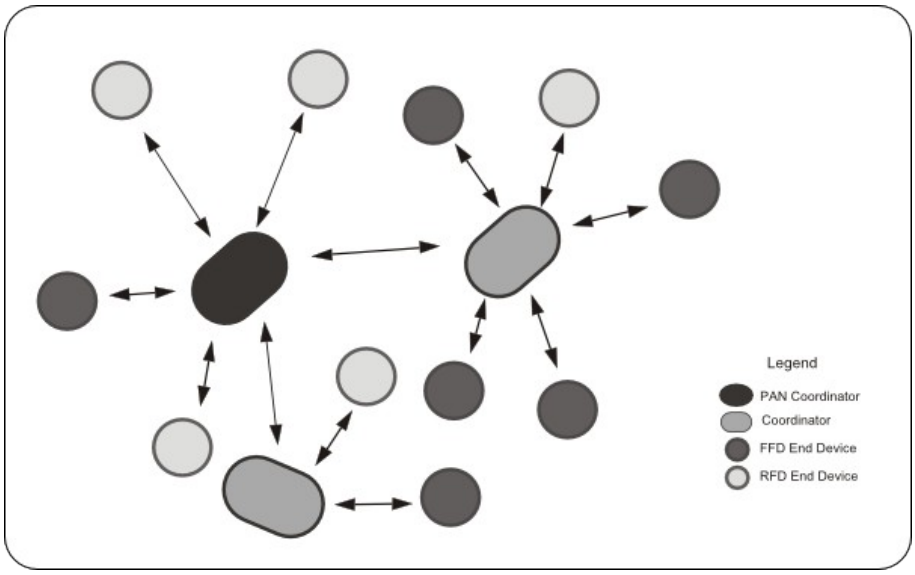

Figura 7: Topologia cluster

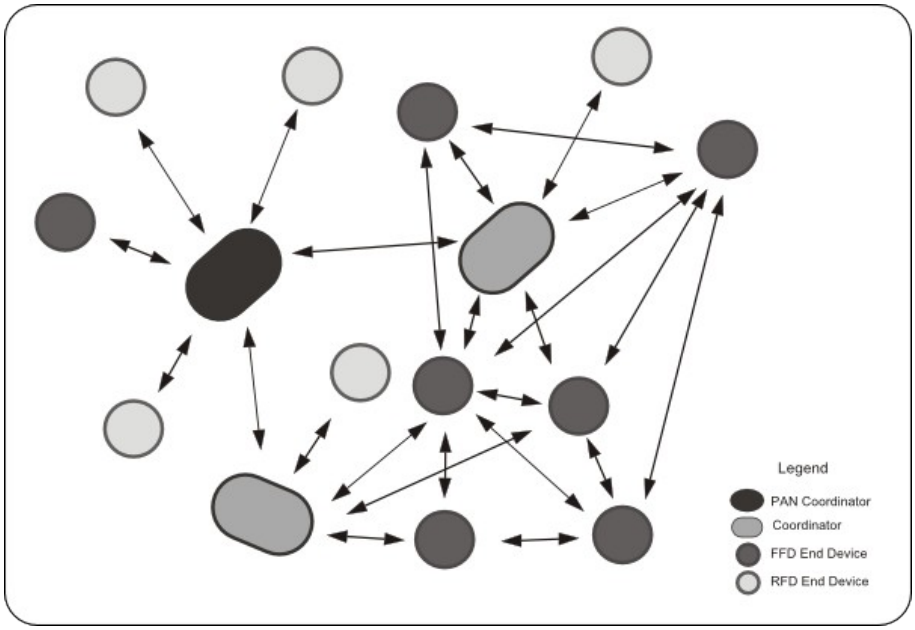

Figura 8: Topologia mesh

- **Mesh:** Similar ao cluster, mas nessa topologia tanto os coordenadores quando os clientes podem comunicar-se entre si.

2.2.2. MiWi

O protocolo MiWi foi desenvolvido pela Microchip em para implementação de

comunicação sem fio em dispositivos com poder de processamento e memória reduzidos, sendo portável em toda a série de famílias PIC16, PIC18 e PIC20, compatível com a especificação IEEE 802.15.4.

As taxas de transmissão mantém-se dependentes da frequência de operação, apresentando os mesmos valores dos apresentados para o ZigBee – 2,4 GHz resulta em 250 kbps, 915 Mhz em 40 kbps e 868 Mhz em 20 kbps – assim como as topologias possíveis, deve-se notar porém que o número máximo de dispositivos por coordenador é 127 e o número máximo de coordenadores é 8, totalizando um máximo de 1024 dispositivos na rede.

2.2.3. Comparativo e Decisão

Uma tabela comparativa entre os dois protocolos é apresentada a seguir:

	MiWi	**ZigBee**
Tamanho do Código	Roteador < 16 kbyte Coordenador < 16 kbyte Cliente 2 – 8 kbyte	Roteador 37 – 96 kbyte Coordenador 30 – 64 kbyte Cliente 18 – 40 kbyte
Licença	Padrão proprietário, mas de uso livre em microcontroladores da família PIC	Para uso ao aderir a ZigBee Aliance (ZBA) como ZigBee Adopter
Documentação	*Application Notes* da Microchip, traduzidas a partir do original em chinês, defasada	Diversas documentações da ZBA e de fabricantes de módulos, como a Digi e a Atmel.

	MiWi	**ZigBee**
Comunidade	Poucos desenvolvedores ocidentais, maior parte das informações disponíveis em sites chineses	Muitos desenvolvedores publicando em inglês e alemão, suporte abrangente em sites e listas de discussão
Hardware Suportado	PIC16, PIC18, PIC20	PIC18, PIC20, Atmel, Parallax
Rede	1024 nós, 4 níveis (máximo)	65536 nós, infinitos níveis
Certificação Obrigatória	Certificação de dispositivos wireless segundo leis locais	Certificação de dispositivos wireless segundo leis locais; certificação da ZigBee Aliance
Custos Associados	Uso de microcontroladores da Microchip é obrigatório	$3,500 por ano + taxas de teste + taxa de certificação
Mercado de Automação	Sem aplicações comerciais	Série Home Automation com diversos fabricantes operando em um padrão compatível definido pela ZBA.
		Escolha de Projeto

Tabela 2: Comparativo entre MiWi e ZigBee

Uma vez que os custos de desenvolvimento de um protótipo são independentes de certificação ou associação e que, em caso de produção em massa, tais custos tornam-se insignificantes, a escolha por Zigbee é natural, especialmente por tratar-se de um protocolo com maior comunidade de usuários, melhor documentação e aceitação no mercado de automação.

3. Requisitos

3.1. Requisitos do Protótipo

3.1.1. Requisitos Funcionais

Os requisitos funcionais podem ser descritos em função de casos de uso acionados pelo usuário. Tais casos de uso são reconsiderados nos capítulos 5 e 6 quando do projeto detalhado do módulo e do servidor onde apresentam-se diagramas formais. Por hora segue a seguinte descrição dos casos necessário ao funcionamento básico do sistema:

- **Executar Comando:** O usuário deve ser capaz de selecionar um comando previamente cadastrado e este deve ser reproduzido pelo emissor de infravermelho, controlando assim o dispositivo alvo.
- **Cadastrar Controle:** O usuário deve ser capaz de cadastrar novos comandos por meio da interface fornecida, tais comandos serão apresentados na forma de controles virtuais, cuja semântica será abordada adiante.

3.1.2. Requisitos Técnicos

Embora a proposta de projeto parta claramente do ponto de que há um nível tecnológico suficientemente avançado para a existência de tal produto, ainda é necessário estudar e definir os diversos elementos de tecnologia candidatos a solução e escolhê-los segundo critérios como:

- Boas bibliotecas de código com bom suporte;
- Capacidade de operar com onda portadora de 36 KHz.

Outros requisitos técnicos são definidos aos limites adicionais impostos dado o caráter unitário da prototipagem. O custo mais elevado de alguns componentes, as dificuldades de manufatura de um protótipo sem máquinas especializadas e as limitações de tempo devem ser considerados ao decidir entre a fabricação fiel ao produto, o uso de plataformas de desenvolvimento industriais ou uma solução

híbrida.

3.1.3. Requisitos Financeiros

Embora este trabalho objetive o desenvolvimento de um protótipo e não de um produto – ver tabela adiante –, para o qual seria necessário realizar um estudo de mercado a fim de estabelecer um custo máximo de desenvolvimento e produção, espera-se que, para viabilidade de fabricação do protótipo, os custos sejam estimados em, no máximo, mil reais.

4. Projeto Básico do Protótipo

O protótipo como definido é constituído de duas partes:

1. Software do servidor-pc, responsável pela implementação da interface e da lógica de negócio;
2. Módulo de atuação infravermelho, com recepção dos comandos enviados por meio da rede ZigBee.

O projeto básico do protótipo seleciona componentes chave a serem utilizados nessas três partes, são eles:

- Transceiver de rádio;
- Adaptador USB – ZigBee;
- Microcontrolador do módulo;
- Placa de desenvolvimento para o módulo.

4.1. Seleção de Transceiver de Rádio

4.1.1. Microchip MRF24J40MA

A microchip fornece o circuito integrado MRF24J40, capaz de receber e transmitir dados segundo a especificação IEEE 802.15.4, todavia é necessário o uso de uma série de componentes passivos adicionais, incluindo a antena e componentes passivos para a realização do casamento das impedâncias. Para o desenvolvimento de protótipos, porém, há a possibilidade de utilizar o módulo MRF24J40MA que já inclui todos os componentes necessários ao funcionamento, comunicando-se com o microcontrolador através da interface SPI com três fios. Todo o processamento, e o stack Zigbee, deve ser implementado no microcontolador.

Figura 9: Transceiver de Rádio MA24J40MA

4.1.2. Digi/Maxstream Pro

Solução microprocessada e encapsulada em módulo de 20 pinos que permite a utilização de rede ZigBee em três modos diferentes:

- **Virtual Wire:** Sincroniza dois ou mais módulos ZigBee fazendo com que o sinal lógico fornecido em uma das sete entradas analógicas/digitais do XBee seja reproduzida nos demais. Implementação direta para aplicações de switch e dispositivos booleanos de detecção.
- **Transparente:** Utilização dos pinos 2 e 3 do módulo XBee como substitutos transparentes de uma interface de comunicação serial.
- **Full API:** Acesso a API do módulo, descrita no User Manual para implementação das camadas superiores do modelo OSI.

Os módulos são alimentados com 3.3V, CC, e consomem pouca energia. Todo o stack ZigBee é implementado nos próprios módulos, os quais podem ser configurados através de comantos AT. Estão disponíveis nas seguintes variantes: transceiver 2.4 Ghz compatível com mesh e antena no chip, transceiver 2.4GHz compatível com mesh e antena externa, transceiver de longo alcance (900m a 64 km) e 900MHz compatível apenas em P2P. Todos possuem possibilidade de Update de firmware pela internet através do software Digi X-CTU.

Figura 10: Digi/Maxstream XBee e XBee Pro

Uma vez que este módulo encapsula todo o stack Zigbee, deixando as complicações de software relacionadas a ele, em especial a necessidade de programação em arquitetura de multitarefa, é escolha imediata para o desenvolvimento do protótipo.

4.2. Adaptador USB – ZigBee

Embora haja diversos adaptadores, modems, Zigbee para computador, assim como para refes WiFi 802.11.a/b/g/n, é interessante utilizar um módulo XBee também no computador, aproveitando o encapsulamento da rede. Para tanto, deve-se realizar comunicação serial com este, sendo a escolha padrão o uso de um chip USB-Serial da FTDI, de código 0830-D. A empresa Droids produz um módulo para XBees utilizando este chip, todavia nenhum software é fornecido.

A implementação da comunicação entre o computador e o módulo XBee ligado à porta serial será tratada mais adiante, no projeto de software do protótipo, seção servidor.

Figura 11: DROIDS USB - Serial para ZigBee

4.3. Microcontrolador

A escolha preferencial por microcontrolador é o ATMEGA 328 presente na placa de desenvolvimento Arduino Duemilanove, com a qual, dada sua utilização em outros projetos também o vasto suporte na web – seja por meio do site do fabricante do microchip ou da comunidade que dá suporte ao Arduino. Todavia, antes que seja possível afirmar a viabilidade do uso desse no protótipo deve-se aferir se este é capaz de gerar uma saída de PWM na frequência necessária para a transmissão dos comandos IR.

Segundo o *datasheet* do fabricante, o ATMEGA 328 apresenta 3 timers que podem ser configurados para a saída PWM, sendo eles:

- *Timer 0*: 8 bits
- *Timer 1*: 16 bits
- *Timer 2*: 8 bits

Uma vez que o *Timer 0* e o *Timer 1* são utilizados pelo bootloader do Arduino para configuração do gerenciamento de interrupções, a melhor opção é utilizar o *Timer 2*, cujos canais A e B estão ligados respectivamente aos pinos 11 e 3 da placa de desenvolvimento para gerenciar o PWM. Assim sendo, a documentação informa que o PWM pode funcionar em dois modos:

4.3.1. Fast PWM (PWM Rápido)

A figura a seguir ilustra o modo de funcionamento conhecido como *Fast PWM*, neste modo um contador parte de zero até um valor limite, cujo máximo é 255, dois valores limites A e B, denotados por OCRnA e OCRnB onde n é o número do *Timer*. Enquanto o valor do contador é inferior, a saída (A ou B) é mantida em *HIGH*, sendo esta condição falsa, o valor vai para *LOW*.

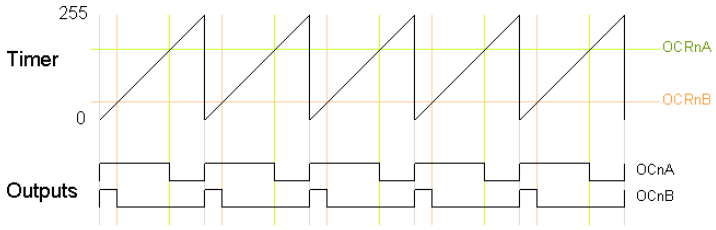

Figura 12: Fast PWM

No modo *Fast PWM*, pode-se regular a frequência de duas maneiras:

1. **Alteração do *Prescale* do Contador**

   O contador pode ser incrementado em diferentes períodos, altera-se estes períodos através da escolha de um divisor denominado *prescale*. No ATMEGA328, os valores possíveis são 1, 8, 32, 64, 128, 256 e 1024.

2. **Alteração do Valor Máximo de Contagem**

   Nesta opção inutiliza-se o OCnA, setando-o como valor máximo para a contagem. Ao atingir o valor especificado neste registrador, o contador é reiniciado. A saída A passa a ter o comportamento conhecido como PMW de meia frequência:

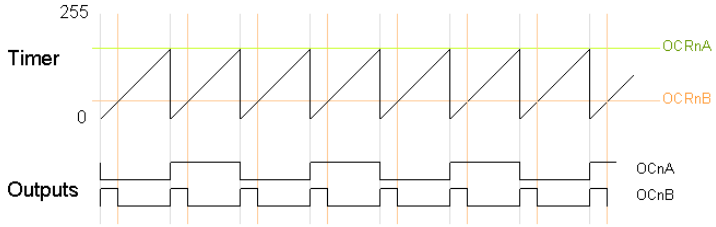

Figura 13: Fast PWM com regulagem por OCnA

Sendo assim a frequência resultante pode ser calculada, aplicando-se os dois métodos utilizando a fórmula:

$$f_{\text{PWM}} = \frac{F_{\text{Oscilador}}}{N_{\text{Prescale}} * N_{\text{OCnA}}}$$

Figura 14: Frequência do Fast PWM

4.3.2. Phase-correct PWM (PWM de Fase Correta)

O funcionamento do *Phase-correct PWM* é tão simples quanto o *Fast PWM*. Neste modo, o contador é incrementado até um valor máximo (255 ou OCnA) e logo depois decrementado, as saídas tem comportamento semelhante, mantendo-se em *HIGH* enquanto o contador for inferior ao valor configurado para as saídas A e B. O controle de frequência pode ser realizado usando as duas técnicas mencionadas no item 4.3.1. As figuras abaixo ilustram o comportamento.

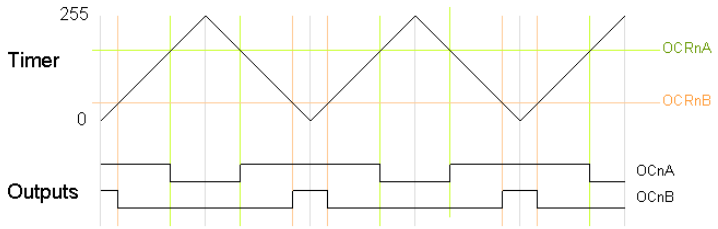

Figura 15: Phase-correct PWM

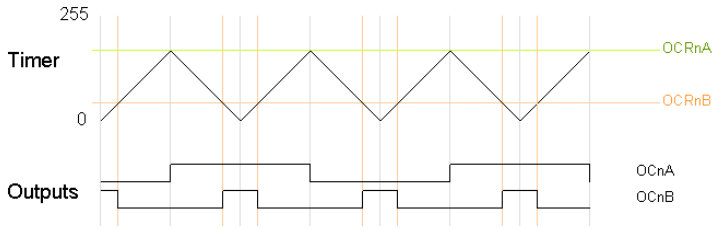

Figura 16: Phase-correct PWM com frequência regulada por OCnA

Este modo apresenta frequências menores, como denotado pelo fator 2 no denominador da fórmula, todavia sua saída é mais simétrica.

$$f_{\text{PWM}} = \frac{F_{\text{Oscilador}}}{2 * N_{\text{Prescale}} * N_{\text{OCnA}}}$$

Figura 17: Frequência do Phase-correct PWM

4.3.3. Duty Cicle

Para um PWM, o *duty cicle* pode ser facilmente entendido como a razão entre o tempo durante o qual o sinal permanece em *HIGH* e o período da onda. A figura abaixo demonstra diferentes valores de DC:

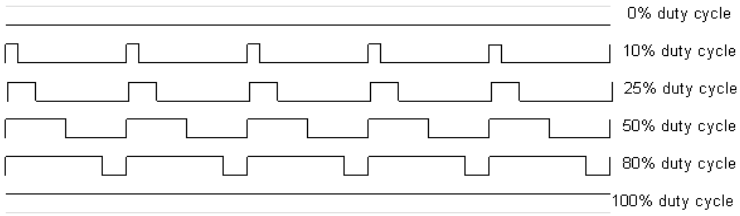

Figura 18: Duty Cicle, diferentes valores de DC para onda quadrada

4.3.4. Viabilidade

A tabela a seguir demonstra as frequências máxima e mínima para cada *prescale* possível do ATMEGA328, para ambos os modos de PWM, tomando como frequência do oscilador os 16 Mhz do cristal presente no Arduino Duemilanove. As frequências são apresentadas em Hertz.

	Fast PWM		**Phase-correct PWM**	
Prescale	**OCnA = 1**	**OCnA = 255**	**OCnA = 1**	**OCnA = 255**
1	16000000	62745	8000000	31373
8	2000000	7843	1000000	3922
32	500000	1961	250000	980
64	250000	980	125000	490
128	125000	490	625000	245
256	625000	245	31250	123
512	31250	123	15625	61
1024	15625	61	7813	31

Tabela 3: Viabilidade do PWM do Arduino

Uma vez que os sinais de infravermelho para controle remoto são modulados na faixa de 30 KHz a 40 KHz, pode-se notar que não só é viável o uso do ATMEGA328 no Arduino Duemilanove como existem várias possibilidades para OcnA e Prescale.

Seja o denominador da frequência final uma constante composta pelo fator multiplicador e o *prescale*, é claro, dadas as fórmulas 16 e 17, que a utilização do *Fast PWM* com o menor *prescale* possível permite melhor ajuste da frequência final em decorrência da variação de OCnA.

O Duty Cicle pode ser ajustado segundo a fórmula:

$$\frac{N_{\mathrm{OCnB}}}{N_{\mathrm{OCnA}}} = DC$$

Figura 19: DC

4.3.5. Arduino

Demonstrada a viabilidade, do uso do microcontrolador Atmel ATMEGA328 na placa de desenvolvimento Arduino Duemilanove deve-se agora apresentá-la.

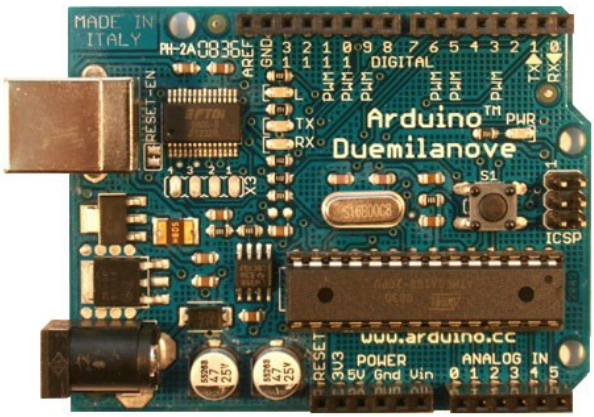

Figura 20: Arduino Duemilanove

A tabela abaixo apresenta um sumário das características da placa, tópicos de interesse são discutidos a seguir:

Microcontrolador	ATMEGA328
Tensão de Operação	5V
Tensão de Alimentação (recomendada)	7-12V
Tensão de Alimentação (limites)	6-20V
Entradas e Saídas Digitais	14 (seis dispõem de PWM)
Entradas Analógicas	6
Corrente (CC) em Pino Digital	40 mA
Corrente no Pino de 3.3 V	50 mA
Memória Flash	32 KB (2 para o *bootloader*)
EEPROM	1 KB
Clock	16 MHz

Tabela 4: Características do Arduino Duemilanove

4.3.5.1. Energia

O Arduino Duemilanove pode ser alimentado por meio da própria conexão USB ou de uma alimentação externa. A placa seleciona a fonte automaticamente. A alimentação externa deve ser provida por uma bateria ou um adaptador AC – DC, podendo serem ligados a um conector de 2.1mm e centro positivo presente na placa ou diretamente aos pinos de Vin e POWER.

A placa pode ser alimentada com qualquer tensão entre 6 e 20 V, todavia a recomendação é para que se use valores entre 7 e 12V, para evitar, por um lado, flutuações que causariam instabilidades na placa e, por outro, superaquecimento do regulador de tensão.

Os pinos de energia são:

- **Vin:** Ligado ao terminal positivo do regulador de voltagem, este pino pode ser utilizado para alimentação do circuito ou ainda para acessar a tensão de alimentação diretamente;
- **5V:** A tensão regulada utilizada para alimentar o microcontrolador e demais componentes da placa;
- **3V3:** Saída de 3.3V gerada pelo adaptador USB – Serial. Dreno máximo de

corrente é 50 mA.

- **GND:** Terra.

4.3.5.2. Comunicação

O Arduino Duemilanove dispõe de diversas facilidades para comunicação. Uma vez que o ATMEGA328 possui dois pinos para comunicação serial via UART TTL (5V), os quais são ligados aos pinos 0 (RX) e 1 (TX) da placa. O chip da FTDI presente na placa utiliza esses canais para a comunicação com a entrada USB e, *drivers* disponíveis no site da FTDI, permitem a criação de uma porta COM virtual no PC, capaz de comunicar-se com o Arduino.

Além disso, o microcontrolador ainda suporta I2C e SPI. Demais interfaces de comunicação serial podem ser criadas utilizando outros dois pinos quaisquer, por meio da biblioteca *SoftwareSerial* inclusa no framework.

4.3.5.3. Programação

O Arduino Duemilanove vem pré-programado com um *bootloader* que permite estabelecer a conexão serial com o computador e realizar a programação, por meio do ambiente de desenvolvimento do Arduino, sem a necessidade de *hardware* externo para *ICSP – In Circuit Serial Programming* –, utilizando apenas a interface USB.

O ambiente disponível para programação é baseado em Wiring e Prototype e pode ser baixado gratuitamente do site www.arduino.cc. A sintaxe é basicamente C++.

4.4. Programação do Servidor

Uma vez que o servidor será implementado em um PC, não é necessário realizar discussão sobre *hardware*. Todavia, dada o leque de opções possíveis para o desenvolvimento de *software*, algumas decisões importantes devem ser tomadas.

A fim de possibilitar um desenvolvimento ágil e multiplataforma, foi escolhida uma linguagem interpretada com bibliotecas bem documentadas e *framework* de desenvolvimento de aplicações Web. A seguir apresentam-se estes elementos. Python (a linguagem), pySerial (biblioteca para comunicação serial, multiplataforma) e Django (*framework*).

4.4.1. Python

Python é uma linguagem de programação de alto nível, interpretada e orientada a objetos, cujo projeto e a filosofia dão ênfase especial à legibilidade do código e a generalidade de aplicação. Possui um sistema de gerenciamento dinâmico de memória, similar aos encontrados em linguagens como Ruby, Tcl, e Perl.

A linguagem é desenvolvida por uma comunidade gerenciada pela *Python Software Foundation*, sem fins lucrativos. A fundação mantém ainda a implementação de referência da linguagem, conhecida como CPython (Python implementado em C). Outras implementações como Jython (em Java) e mesmo a PyPy (Python em Python) estão disponíveis.

CPython e sua biblioteca nativa estão disponíveis para Windows e para a maior parte dos sistemas operacionais baseados em Unix.

O site oficial do projeto é o: http://www.python.org/

4.4.2. pySerial

Uma vez que as bibliotecas padrão do Python não incluem uma interface de comunicação através de porta serial, foi desenvolvida a pySerial, uma biblioteca de código aberto que implementa a funcionalidade de modo multiplataforma – ainda que para Windows sejam necessárias algumas extensões especiais (pyWin32).

O site oficial do projeto é o: http://pyserial.sourceforge.net/

4.4.3. Django

Django é um *framework web* implementado em Python e desenvolvido para produção ágil de websites, trata-se de um projeto iniciado em julho de 2005 que atingiu maturidade, versão 1.0, em 2008 e é mantido por uma fundação própria. Dentre suas características, são relevantes para este projeto a utilização de uma arquitetura MVC e a disponibilidade de um servidor de testes nativo.

O site oficial do projeto é o: http://www.djangoproject.com/

5. Módulo

5.1. Projeto de Hardware

O projeto de *hardware* para o protótipo é razoavelmente simples, integrando o XBee ao Arduino ao conectar suas interfaces seriais e alimentar aquele com a saída de 3.3V deste. O pino 3, correspondente a saída B de PWM do *Timer2*, do Arduino é ligado ao LED infravermelho.

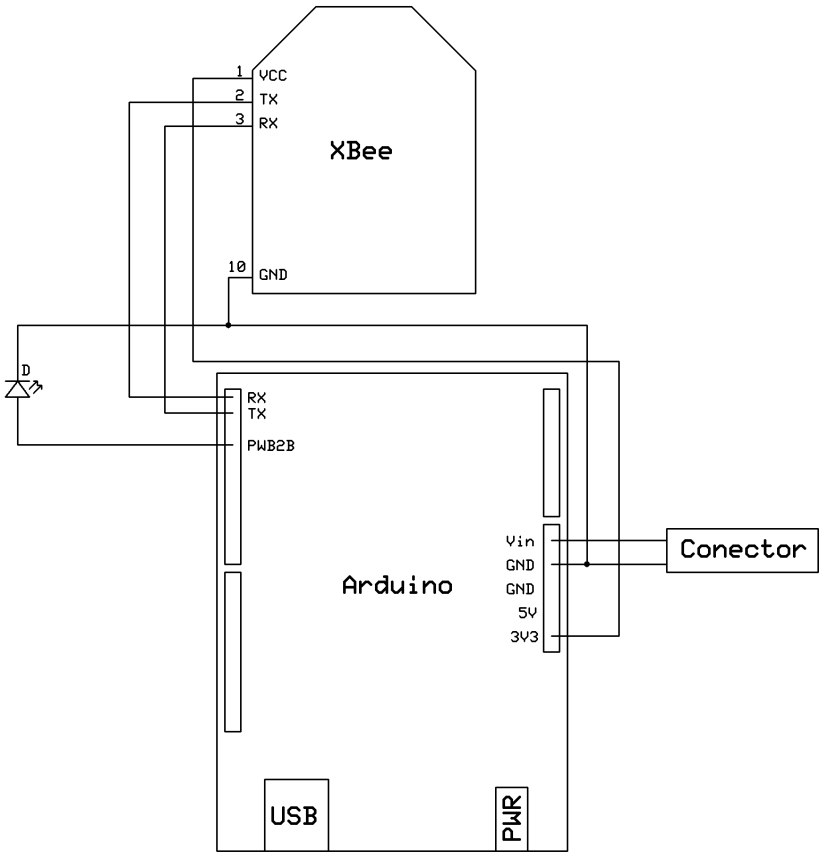

Figura 21: Diagrama de Conexão do Módulo

5.3. Fabricação

Após os testes em *protoboard*, optou-se por fabricar uma placa que possa ser montada facilmente sobre o Arduino. Para isso utilizou-se uma placa de

prototipagem e uma placa de *breakout* para o módulo XBee, em vermelho, uma vez que este não possui espaçamento compatível para os pinos. Pinos extras, em aberto, foram adicionados para firmar a placa quando da montagem sobre o Arduino. As fotos a seguir demonstram os resultados:

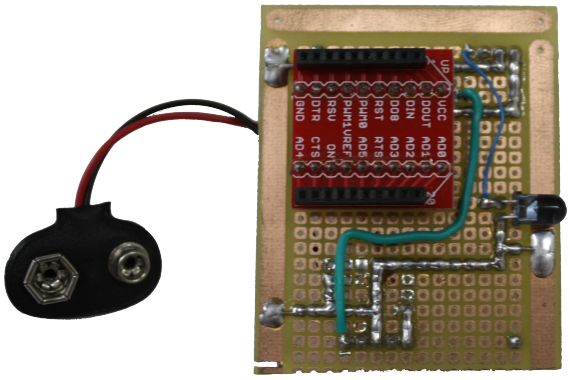

Figura 22: Placa fabricada, Frente

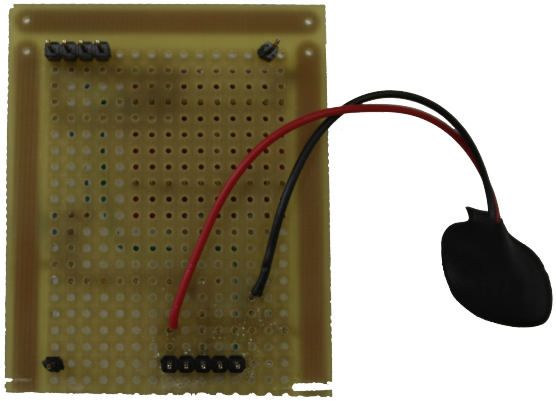

Figura 23: Placa Fabricada, Verso

5.4. Projeto de Software

O software para o módulo apresenta apenas três casos de uso, descritos a seguir:

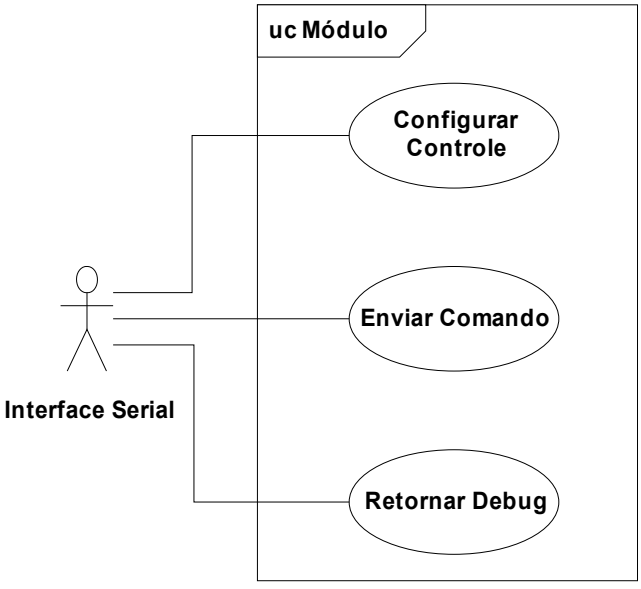

Figura 24: UC: Módulo

Configurar Controle: Altera as variáveis de configuração de transmissão infravermelho utilizando os valores recebidos via *string* na porta serial. As variáveis alteradas devem representar:

1. Ratio (Duty Cicle);
2. Ajuste fino de tempo em HIGH em µs;
3. Ajuste fino de tempo em LOW em µs;
4. Tempo em HIGH em µs;
5. Tempo em LOW em µs;
6. Frequência da onda transportadora (carrier) em Hz;
7. Número de bits do sinal;

Enviar Comando: Recebe uma *string* da porta serial, cuja informação é uma sequencia de zeros e uns que devem ser processados, segundo os parâmetros configurados, acionando o LED infravermelho e transmitindo o comando.

Retornar Debug: A fim de observar o estado das variáveis configuradas, o módulo deve retornar uma listagem de *debug* pela porta serial.

5.5. Implementação

Por fim, uma vez que o código desenvolvido para o microcontrolador é extremamente breve, apresenta-se a listagem:

```
#define MAX 16
#define MSG_TYPE_ID 1
#define DEBUG 0
#define CLOCK 2000000

unsigned int i, j, k, r, ratio = 25, ajusteHigh = 0,
    ajusteLow = 0, tempoLow = 889, tempoHigh = 889,
    carrier = 36000, numBits = 128, divisor = 55, delayHigh = 889,
    delayLow = 889;

char c, sinal[256], buffer[MAX];
boolean msgPronta;
```

```
void setup(){
  pinMode(3, OUTPUT);
  pinMode(11, OUTPUT);
  Serial.begin(9600);
}
void loop(){
  if(!msgPronta){
     // Leitura da serial
     while(Serial.available()){
       c = Serial.read();
       // COMEÇO DE UMA MENSAGEM
       if(c == ':'){
         msgPronta = false;
         j = 0;
         sinal[j++] = c;
       // FINAL DE UMA MENSAGEM
       }else if(c == ';'){
         sinal[j] = c;
         msgPronta = true;
         break;  // PARA SAIR DESTE WHILE
       // CARACTER DE UMA MENSAGEM
       }else if(j > 0){
         sinal[j++] = c;
         if(j > 256)
           j = 0;
       }
     }
  }else{
    // ENVIA ECHO PARA CONFIRMACAO CASO ESTEJA EM MODO DEBUG
    #if DEBUG == 1
        if(msgPronta){
          for(i=0; i<=j; i++){
            Serial.write(sinal[i]);
          }
          Serial.println("OK?");
        }
    #endif

    // Processa cada caso
```

```
    // C -> Command : Usa as confs atuais para enviar a msg por IR
    // S -> Setup : Seta novas configurações
    // D -> Debug : Responde via Serial

    switch(sinal[MSG_TYPE_ID]){
      // NOVO SETUP
      case 'S':
      #if DEBUG == 1
          Serial.println("ENVIOU SETUP");
      #endif
        /*
            FLAGS
            R : Ratio
            h : Ajuste high (us)
            l : Ajuste low (us)
            H : Tempo high (us)
            L : Tempo low (us)
            C : carrier (freq em Hz)
            B : numBits
        */
        // SETA TODOS OS PARAMETROS
        sscanf(sinal, ":SR%dh%dl%dH%dL%dC%dB%d;", &ratio, &ajusteHigh,
&ajusteLow, &tempoHigh, &tempoLow, &carrier, &numBits);
        divisor = CLOCK / carrier;
        ratio = ratio*divisor/100;
        delayHigh = tempoHigh - ajusteHigh;
        delayLow = tempoLow - ajusteLow;

      break;
      // ENVIO DE COMANDO
      case 'C':
      #if DEBUG == 1
          Serial.println("ENVIOU UM COMANDO");
      #endif
        // Processamento do Sinal
        for(i=MSG_TYPE_ID+1; i<j; i++){
          switch(sinal[i]){
            case '1':
              TCNT2 = 0;
```

```
        TCCR2A = 0x63;
        TCCR2B = 0x0A;
        OCR2A = divisor;
        OCR2B = ratio;
        delayMicroseconds(delayHigh);
        break;
      default:
        digitalWrite(3, LOW);
        delayMicroseconds(delayLow);
        break;
    }
  }// FIM PROCESSAMENTO
  // APAGA O LED E ESPERA O FRAME RESTANTE
  digitalWrite(3, LOW);
  delayMicroseconds((numBits-j)*tempoLow);

break;

// ECHO DA CONFIG ATUAL NA SERIAL
case 'D':
  Serial.println("CONFIGURACAO ATUAL");
  Serial.println("------------------------");
  Serial.write("ratio:");
  Serial.println(ratio);
  Serial.write("ajusteHigh:");
  Serial.println(ajusteHigh);
  Serial.write("ajusteLow:");
  Serial.println(ajusteLow);
  Serial.write("tempoHigh:");
  Serial.println(tempoHigh);
  Serial.write("tempoLow:");
  Serial.println(tempoLow);
  Serial.write("carrier:");
  Serial.println(carrier);
  Serial.write("divisor:");
  Serial.println(divisor);
  Serial.write("j:");
  Serial.println(j);
  Serial.write("array sinal:");
```

```
        Serial.println(sinal);
      break;
    }

    // PROCESSOU A MSG, LOGO ELA NAO ESTÁ MAIS PRONTA
    msgPronta = false;
  }
}
```

Uma observação interessante é o uso de digitalWrite() para o caso em que a saída deve ser zero, uma vez que alterar DC para 0% ainda produz um pequeno pico, ocasionado pela ordem de execução, sendo o comparador o último passo a ser executado no ciclo de *clock*.

6. Servidor

Neste capítulo serão discutidos alguns aspectos básicos do *software* do servidor, para a listagem completa verificar o diretório ***/software/python*** no CD.

6.1. Projeto de Software

Os casos de uso para o servidor do protótipo são simples:

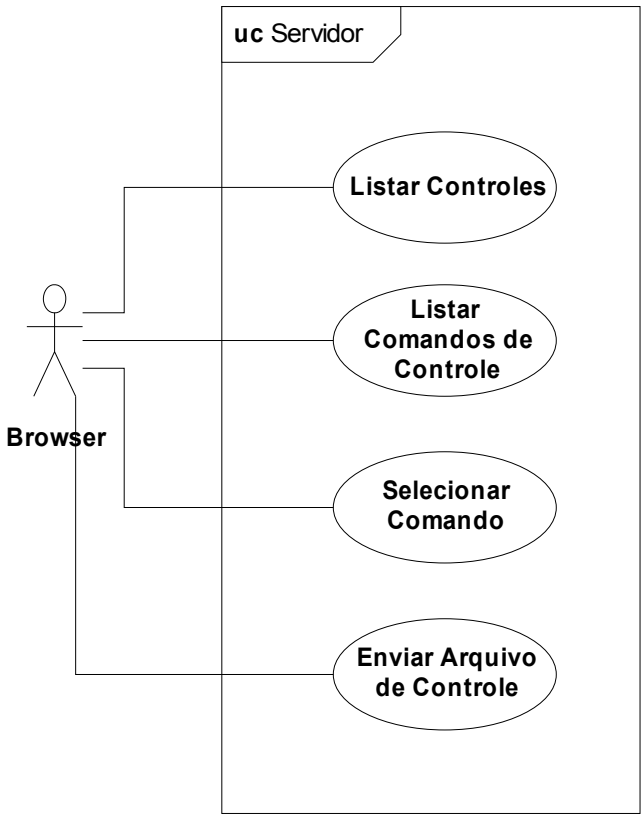

Figura 25: UC: Servidor

O *framework* Django implementa uma arquitetura do tipo MVC, cujos arquivos principais são:

- **urls.py** – Responsável pela camada de controle, recebe a requisição HTTP e aplica a rota configurada para a *view* solicitada, passando os parâmetros necessários a ela.
- **models.py** – Define os modelos de dados do sistema, suas operações e lógica de negócio. Neste caso trata-se apenas a classe *VirtualControl*.

- **views.py** – Implementa uma série de métodos a serem invocados quando da requisição HTTP apropriada, é responsabilidade destes métodos manipular o modelo e invocar o modelo a ser renderizado para o usuário.

Deve-se portanto projetar uma classe para modelar o comportamento do controle virtual, a qual será denominada V*irtualControl* e algumas *views*:

View	**Descrição**
index	Índice e página padrão, lista os controles disponíveis.
control	Para um controle, lista os comandos disponíveis e permite selecionar algum.
sendCommand	Envia um comando, retornando apenas uma confirmação, uma vez que será solicitado via AJAX.
upload_file	Permite realizar o *upload* de um arquivo de controle.

Tabela 5: Descrição das views

A classe VirtualControler possui os seguintes métodos:

Métodos da Classe *VirtualControler*	
Método	**Descrição**
__init__(self, controlFile)	Construtor, recebe como parâmetro o caminho para um arquivo de controle virtual, o qual interpretará.
getSetupString(self)	A partir do arquivo de controle virtual carregado, retorna a *string* de configuração do controle.
getCommandString(self, command)	A partir do controle, busca o comando selecionado e retorna a *string* formatada correspondente.
sendStringSerial(self, str)	Abre a conexão serial e envia uma *string*.
sendSetupString(self)	Envia a *string* de setup por meio da porta serial, utilizando os métodos *getSetupString(self)* e *sendStringSerial(self, str)*.
sendCommandString(self, command)	Envia a *string* correspondente ao comando por meio da porta serial, utilizando os métodos *getCommandString(self, command)* e *sendStringSerial(self, str)*.

Tabela 6: Métodos da classe VirtualControler

Para implementação de uma interface mais interessante, opta-se por utilizar o

recurso conhecido como AJAX. Por meio de implementação de rotinas javascript, realizam-se diversas requisições HTTP sem que seja necessário recarregar a página, enviando, assim, os comandos quando o link é premido e mantendo diversas repetições enquanto esse estado não varie; similarmente a um controle remoto convencional.

6.1.1. AJAX

A W3C, por meio da *W3C Schools*, define o AJAX (*Asynchronous JavaScript and XML*) como um técnica de programação para criar aplicações web melhores, mais rápidas e mais amigáveis. Utilizando AJAX, uma rotina JavaScript pode comunicar-se diretamente com o servidor por meio do objeto XMLHttpRequest, trocando informações com este sem que haja necessidade de recarregar a página. Isso é feito graças a troca assíncrona de dados, na forma de requisições HTTP, diretamente com o servidor.

6.2. Descrição do Controle Virtual

Uma vez que o sistema contempla uma miríade de diferentes controles, não sendo estes integrados a priori ao projeto, deve-se implementar um modo de descrevê-los, de modo a serem independentes de plataforma e preferencialmente não utilizar aplicações adicionais, como banco de dados. A solução escolhida foi uso de arquivos XML.

6.2.1. XML

A *Extensible Markup Language* (XML) é um conjunto de regras para a codificação de documentos eletrônicos, tendo sido definida na *XML 1.0 Specification*, a qual foi produzida pela W3C (*World Wide Web Consortium*), e outras normas nela relacionadas. Trata-se de um padrão gratuito e aberto.

Como requisitos de projeto, o XML dá grande ênfase à simplicidade, generalidade e usabilidade na Internet. Trata-se de um formato textual, definido em UNICODE, e suportado pela grande maioria das linguagens de programação do mundo. Diversos

outros formatos famosos derivam do XML, como RSS, ATOM e XHTML. É também padrão na definição de documentos de suíte de aplicações para escritório como o Microsoft Office, iWork e OpenOffice.org.

Um documento XML é composto basicamente de marcadores (*tags*), elementos (*elements*) e atributos (*attributes*).

- ***Tag:*** Uma *tag* começa com um “<” e é encerrada com um “>”. Existem três tipos de *tag*: as iniciais, como <controle>, as finais, </controle> e as vazias, como,
 do XHTML.
- ***Element:*** Componente lógico de um documento XML, compreende a união de um par de *tag* de início e final, inclusive, com todo seu conteúdo. No caso de uma *tag* vazia, ela própria é um elemento.
- ***Attribute:*** Uma construção da marcação, encontrada dentro de uma *tag* de início ou final. Por exemplo em <minhatag exemplo=”a”>, *exemplo* é um atributo e *a* seu valor.

6.2.2. Mapeamento de Controle Real

Para o mapeamento do controle definiu-se a seguinte semântica:

Tag de ínicio: <ircontrol>

Atributos: name, carrier, ratio, ajusteLow, ajusteHigh, tempoLow, tempoHigh, numBits

Tag final: </ircontrol>

Tags filhas: <command>

Tag de ínicio: <command>

Atributos: name

Tag final: </command>

Tags filhas: nenhuma

Para a realização o mapeamento do controle, utiliza-se um osciloscópio e um circuito simples.

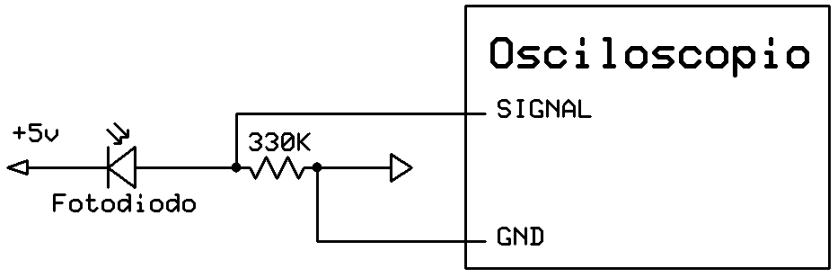

Figura 26: Circuito para mapeamento do botão

As medidas necessárias estão indicadas na imagem, de forma que este botão traduz-se em:

```
<ircontrol name="Rádio Philips" carrier="36000" ratio="40" ajusteLow="0"
ajusteHigh="0" tempoLow="889" tempoHigh="889" numBits="128">
      <command name="POWER">01011001101010011010011011010</command>
</ircontrol>
```

6.2.3. Exemplo

O controle mapeado utilizado para a demonstração é listado a seguir:

```
<?xml version="1.0"?>
<ircontrol name="Rádio Philips" carrier="36000" ratio="40" ajusteLow="0"
ajusteHigh="0" tempoLow="889" tempoHigh="889" numBits="128">

      <command name="VOL -">01010101101010101001101010001</command>
      <command name="VOL +">01010101101010101001101010010</command>
      <command name="TRAY">01101001100110101010100110</command>
      <command name="POWER">01011001101010011010011011010</command>
      <command name="CD">01011001100110100101010101</command>
      <command name="TUNER">01010101101010010101010101</command>
      <command name="AUX">01010101100110010101010101</command>
      <command name="PREVIOUS">01011001101010011001010101</command>
```

```
        <command name="NEXT">01011001101010011001010110</command>
        <command name="DOWN">01011001101010011011010101001</command>
        <command name="UP">0101100110101001011010101010</command>
        <command name="STOP">01010101101010010101101010001</command>
        <command name="PLAY">01010101101010010101101010001</command>
        <command name="PROGRAM">01010101101010010110100110010</command>
        <command name="DISPLAY">01100101101010101001100110010</command>
        <command name="TIMER">011001011010101010010110100010</command>
        <command name="SLEEP">01011001101010100110100101110</command>
        <command name="DBB">011001011010101010101001010110</command>
        <command name="DSC">011001011010101010100101010101</command>
        <command name="MUTE">01011001101010101010010110001</command>

</ircontrol>
```

7. Utilização do Sistema

7.1. Dispositivos

Para utilização do sistema, o transmissor XBee, por meio do adaptador fornecido pela Droids (4.2), deve ser conectado ao computador no qual será executado o servidor. O módulo com o Led infravermelho deve ser energizado, havendo para isso duas opções:

- Energização por meio do cabo USB, ligando o módulo a um computador hospedeiro ou carregador USB;
- Energização com tensão entre 7 e 12V por meio do conector disponível.

7.2. Inicialização do Servidor

O servidor, contido no diretório *software/python/base* do CD, deve ser executado após a instalação dos aplicativos necessários:

- Python 2.6.4;
- Django 1.1.1;
- PySerial 2.5 rc1;
- PyWin32 build 213 (somente para Windows);

Todos estes aplicativos estão disponíveis no CD, em *software/ambiente*. Uma vez configurados, basta executar o comando:

python manage.py runserver 127.0.0.1:8080

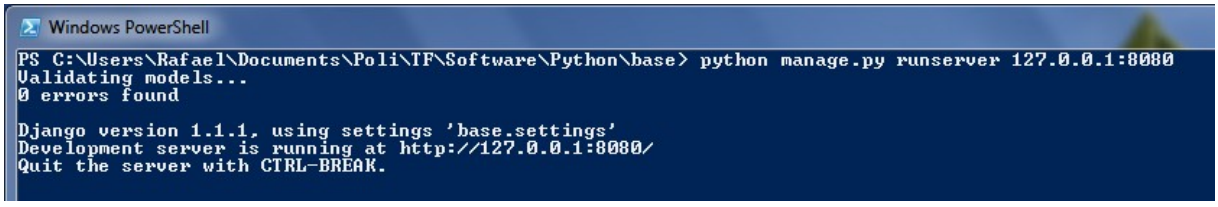

Figura 27: Inicialização do Servidor

Após esse passo, basta acessar o endereço http://127.0.0.1:8080/izapper/ no navegador para ter acesso à interface.

7.3. Interface

A interface do sistema é bem simples, a primeira página traz uma listagem dos controles instalados e a opção de adicionar outros. A seguir, apresentam-se as telas do sistema, com comentários quando necessário.

7.3.1. Seleção do Controle

Na tela principal, para selecionar um controle, basta clicar sobre seu nome. São listados os controles presentes no diretório *software/python/base/ircontrols*.

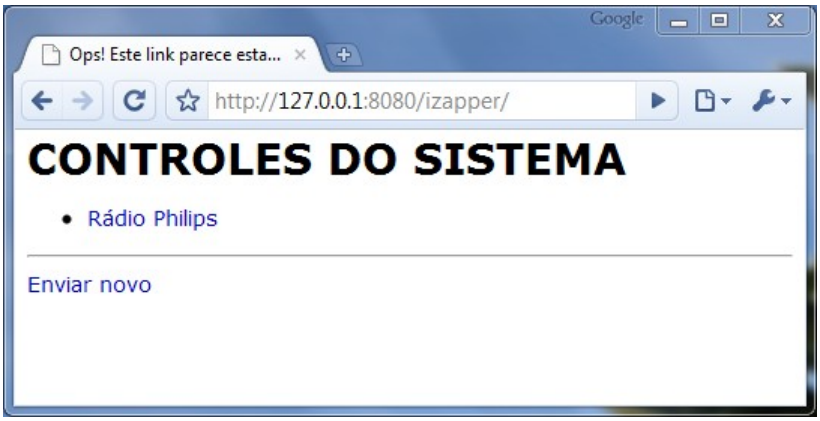

Figura 28: Interface, Seleção do Controle

Uma vez que um controle seja selecionado, os dados de configuração são enviados (ver comando ao final da tela) e a listagem de comandos disponíveis apresentada.

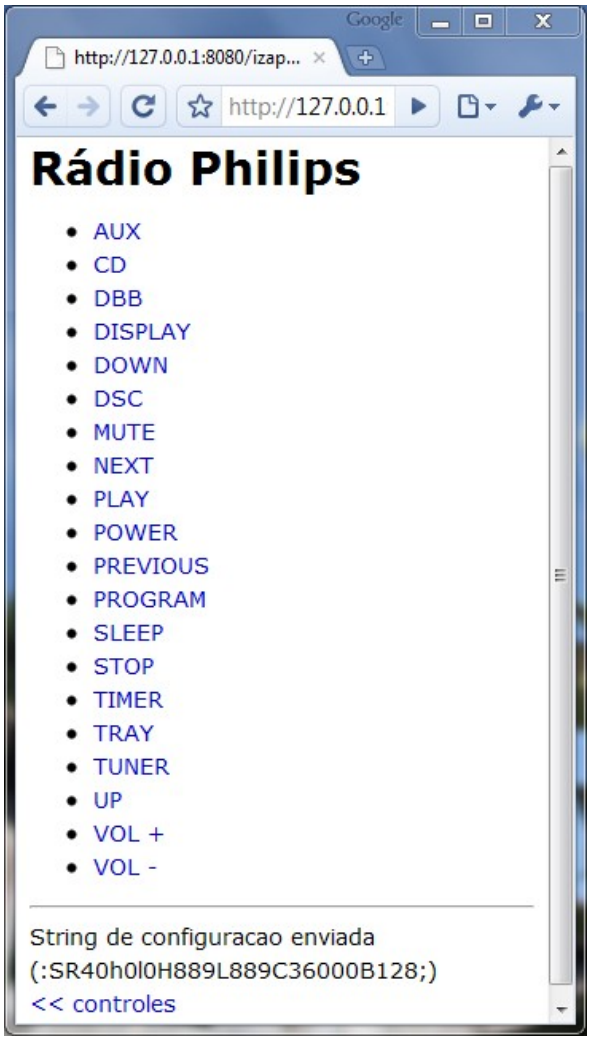

Figura 29: Interface, Listagem dos Comandos

7.3.2. Envio de Comandos

O envio de um comando é realizado por meio do clique no seu título, apresentado na tela de listagem de comandos, aqui utilizou-se o recurso do AJAX para fazer com que, caso o botão do mouse seja mantido pressionado, o comando seja enviado múltiplas vezes – simulando assim o comportamento de um controle convencional. Os dados enviados podem ser visualizados no final da tela.

Figura 30: Interface, Envio de um comando

7.3.3. Adicionar Controle

Um arquivo XML com a descrição de um controle pode ser enviado por meio do formulário de envio, sendo armazenado no diretório *software/python/base/ircontrols* e automaticamente listado no acesso subsequente.

Figura 31: Interface, Adicionar Controle

8. Finalização

8.1. Desempenho

O protótipo possui excelente desempenho, sendo capaz de controlar perfeitamente o aparelho utilizado para testes, o qual implementa um protocolo muito próximo ao RC5x apresentado em anexo. A comunicação ZigBee mostrou-se eficiente, com mais de 20 metros sem obstáculos e com bom funcionamento ainda que atravessando até duas paredes.

8.2. Conclusão

Todos os objetivos do trabalho proposto foram alcançados, contemplando projeto de software e hardware, fabricação e implementação. Uma vez que o orçamento final do protótipo sitou-se abaixo dos mil reais propostos, deve-se ressaltar um êxito especial neste quesito.

Uma vez fabricado o protótipo e implementado o software necessário, a adição de novos controles virtuais esbarra apenas na necessidade de mapeá-los, já que o controle-teste apresenta perfeito funcionamento. Com poucas alterações no software do servidor é possível oferecer uma API consistente para o desenvolvimento de pequenos clientes (*desktop widgets*, por exemplo), melhorias na apresentação (gerenciada por meio de modelos HTML) ou possibilitar a criação de macros, integrando diversos comandos de diferentes controles. Todavia estas alterações estão fora do escopo proposto para o trabalho.

8. Bibliografia

(1) SB Projects

Disponível em: <http://www.sbprojects.com/knowledge/ir/ir.htm>

Último Acesso em: 28/04/2008

(2) ePanorama

Disponível em: <http://www.epanorama.net/links/irremote.html>

Último Acesso em: 28/04/2008

(3) Majority IR

Disponível em: <http://majority.110mb.com/index.htm>

Último Acesso em: 28/04/2008

(4) INCROPERA, F . **Fundamentos de Transferência de Calor e de Massa**. 5a Edição. LTC. Rio de Janeiro, 2003

(5) MRF24J40 Data Sheet

Disponível em: <http://ww1.microchip.com/downloads/en/DeviceDoc/DS-39776b.pdf>

Último Acesso em: 08/06/2009

(6) MRF24J40MA Data Sheet

Disponível em: <http://ww1.microchip.com/downloads/en/DeviceDoc/70329b.pdf>

Último Acesso em: 08/06/2009

(7) PIC18F97J60 Family Data Sheet

Disponível em: <http://ww1.microchip.com/downloads/en/DeviceDoc/39762d.pdf>

Último Acesso em: 08/06/2009

(8) PIC24F16KA102 Family Data Sheet

Disponível em: <http://ww1.microchip.com/downloads/en/DeviceDoc/PIC24F16KA102_Family_datasheet_39927b.pdf>

Último Acesso em: 08/06/2009

(9) Datasheet do Atmel ATMEGA328

Disponível em: <http://www.atmel.com/dyn/products/product_card.asp?part_id=4198 >

Último Acesso em: 18/11/2009

(10) MiWi Wireless Networking Protocol Stack

Disponível em: <http://ww1.microchip.com/downloads/en/AppNotes/MiWi%20Application%20Note_AN1066.pdf>

Último Acesso em: 08/06/2009

(11) ZigBee 2006 Protocol Stack

Disponível em: <http://www.microchip.com/Microchip.WWW.SecureSoftwareList/secsoftwaredownload.aspx?device=en538049&lang=en&ReturnURL=http://www.microchip.com/stellent/idcplg?IdcService=SS_GET_PAGE&nodeId=1406&dDocName=en538049#>

Último Acesso em: 08/06/2009

(12) Documentação oficial do Python 2.6.4

Disponível em: <http://docs.python.org/>

Último Acesso em: 18/11/2009

(13) Documentação oficial do pySerial

Disponível em: <http://pyserial.sourceforge.net/>

Último Acesso em: 18/11/2009

(14) Documentação oficial do Django

Disponível em: <http://www.djangoproject.com/>

Último Acesso em: 18/11/2009

(15) Extensible Markup Language (XML) 1.0 (W3C)

Disponível em: <http://www.w3.org/TR/REC-xml/>

Último Acesso em: 18/11/2009

(16) Tutorial sobre AJAX (W3C Schools)

Disponível em: <http://www.w3schools.com/Ajax/Default.Asp>

Último Acesso em: 10/11/2009

Anexos

Anexo 1: Teoria de Comunicação Infravermelho

Dispositivos de comunicação através de infravermelho utilizam-sedo chamado infravermelho próximo, uma faixa do espectro de ondas eletromagnéticas cujo comprimento vai de 750 a 1400 nm. Tais onda apesar de invisíveis ao olho humano, ainda apresentam comportamento similar ao da luz visível para fenômenos físicos. Dada esta característica, o fato de ser inofensivo e a facilidade de emitir ondas de frequência bem definida utilizando um LED, o infravermelho tornou-se o padrão para controles remoto em aparelhos como televisores, sistemas de som, ar condicionado e outras aplicações domésticas.

Ondas eletromagnéticas são emitidas por corpos em temperatura diferente de 0 K, incluindo ondas na faixa do infravermelho próximo, por exemplo, modelando o sol como um corpo negro a 5800K, INCROPERA fornece a seguinte abordagem para o cálculo da intensidade de radiação emitida na frequência do infravermelho próximo:

$$F_{\lambda_2,\lambda_1} = F_{0,\lambda_2} - F_{0,\lambda_1}$$

$$F_{1400,750} = f(1{,}4 * 5800) - f(0.75 * 5800) = 0.8606848 - 0.5406005 = 0.3200843$$

$$F_{0,\lambda} = f(\lambda T) = \int_0^{\lambda T} \frac{E_{\lambda,b}}{\sigma T^5} d(\lambda T) \quad P_{IR\,Próximo} = E_b * F_{1400,750} = 20{,}5 * 10^6\, W/m^2$$

$$I_{IR\,Próximo} = \frac{E_b * F_{1400,750}}{\pi} = 6{,}54 * 10^6\, W/m^2$$

Sendo assim, uma vez que estamos cercados de fontes de infravermelho, lâmpadas, velas, nossos corpos e principalmente o sol emitem grandes quantidades de radiação nesta faixa, para evitar interferências trabalha-se usualmente com apenas um comprimento de onda, em geral 940 nm, e utiliza-se modulação.

A modulação pode ser entendida de maneira simples, fazendo-se o LED piscar em determinada frequência e ajustando o circuito receptor, através de um filtro passa-

faixa, para ela é possível eliminar o sinal enviado pelas demais fontes. A figura abaixo ilustra o processo de modulação:

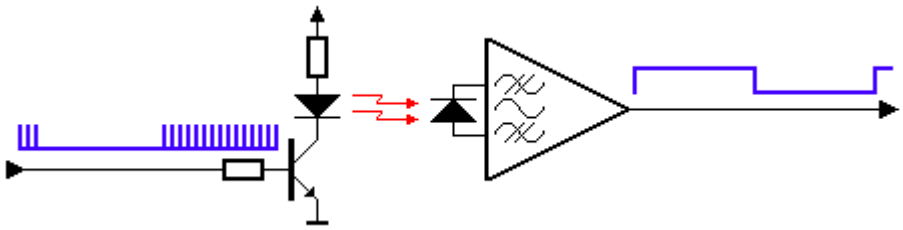

Figura 32: Processo de Modulação, SB Projects

Uma vez que na comunicação serial trata-se o sinal por traços e espaços, podemos dizer que o receptor enxerga como marca todo o tempo em que o LED pisca na frequência modulada e como espaço o tempo restante. Deve-se notar porém que os traços e espaços não correspondem diretamente aos valores lógicos de bit 1 e 0, sendo a forma destes definida no protocolo de comunicação.

A1.1. Protocolos Comerciais

O desenvolvimento dos sistemas de controle remoto por meio de infravermelho deu-se paralelamente em diversas empresas, cada qual originando um protocolo próprio de comunicação. Estes buscavam atender as seguintes necessidades:

1. Eliminação de Erros e Interferências

   Além das interferências ambientais, as quais já foram tratadas na modulação, o receptor infravermelho está submetido a sinais modulados enviados por demais aparelhos assim como a erros de recepção e interpretação dos sinais corretamente enviados. É dever do protocolo prever uma maneira de lidar com esses fatores sem acarretar comportamentos inesperados no sistema.

2. Endereçamento

   O protocolo deve especificar maneiras de realizar o endereçamento do dispositivo a obedecer os comandos, ou seja, garantir que o par controle/aparelho controlado seja único para determinado modelo. Desta

forma, ao regular-se a intensidade de som do televisor, o rádio, ainda que entenda o protocolo e os comandos deve detectar que o endereçamento não é o dele e comportar-se apropriadamente, ou seja, permanecer inalterado.

A seguir serão descritos alguns protocolos comuns: Philips RC5x e RC6x.

A1.1.1. Philips RC5x

O protocolo Philips RC5 foi definido no final da década de 1980 e adotado por diversos fabricantes europeus e americanos, tendo sido até mesmo referenciado como padrão internacional. No começo da década de 1990, o padrão foi revisado a fim de alocar mais um bit de comando, dando origem ao RC5X. Ambos protocolos são largamente utilizados por hobbistas por sua facilidade de implementação.

Características

- 5 bits para endereçamento
- 6 bits para comando (7 bits no RC5X)
- Codificação Manchester (Bifásica)
- Frequência de modulação 36kHz
- Tempo constante por bit, 1.778ms (64 ciclos de 36 kHz)

Modulação

Como mencionado anteriormente, cabe ao protocolo definir um padrão de transmissão para os bits 1 e 0, o Philips RC5x o define utilizando o código bifásico, também conhecido com Manchester Encoding. A figura abaixo ilustra ambos os casos:

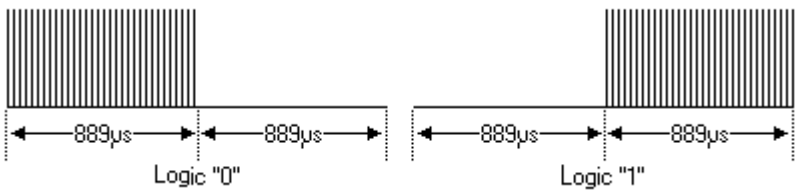

Figura 33: Modulação em RC5, SB Projects

Assim sendo, o valor lógico 0 é composto de 32 pulsos seguidos de 32 ciclos em sinal, enquanto o valor lógico 1 é definido como 32 ciclos sem sinal seguidos por 32 pulsos. O duty cicle deve ser em torno de 1/3 e 1/4, para economia de energia.

Formato

O protocolo especifica um formato padrão de mensagem, uma unidade fechada contendo bits de início, toogle, endereçamento e código de comando. A figura a seguir será utilizada para elucidar a composição dos trens:

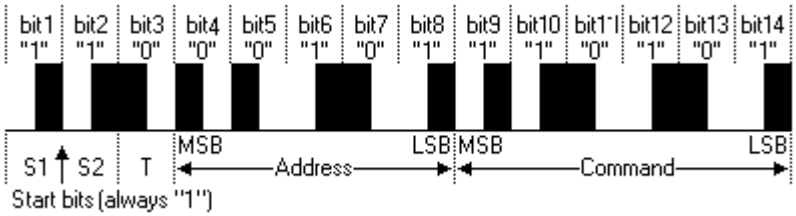

Figura 34: Trem de pulsos em RC5, SB Projects

Os primeiros dois bits definem o início da mensagem e são sempre 1. Em seguida temos um bit de toogle que é alterado sempre que uma tecla é pressionada, desta forma o receptor pode diferenciar se uma tecla está sendo mantida pressionada ou ativada sequencialmente. Após o bit de toogle há cinco bits dedicados ao endereçamento e, por fim, outros seis bits para o código de comando. O protocolo RC5x especifica o inverso do bit S2 como sétimo bit do comando.

A1.1.2. Philips RC6x

Como é de se esperar, o protocolo RC6x é o sucessor do RC5x, tendo sido revisto em função de alocação de mais endereços e comandos e aumento da versatilidade. Embora a documentação oficial deste protocolo seja extensa, faz-se aqui uma síntese.

Características

- Diferentes modos de operação
- Comandos de comprimento variável, dependendo do modo
- Codificação Manchester (Bifásica)
- Frequência de modulação 36kHz

Modulação

No protocolo RC6 cinco símbolos são definidos, a unidade mínima de tempo é definida como 16 períodos do ciclo de 36Hz (1t = 444µs) com *duty cicle* entre 25% e 50%:

- Pulso Líder (Leader Bit)
  Possui um traço de 6t (2.666ms) e espaço de 2t (0.889ms).
- Bits Lógicos 0 e 1
  Utilizando a codificação Manchester, possuem traço e espaço de 1t cada. Ao contrário do RC5x, o 1 lógico é definido como traço-espaço e o 0 lógico como espaço-traço.
- Bits Finais (Trailer Bits)
  Iguais os bits lógicos acima, porém traço e espaço possuem 2t cada.

As figuras a seguir exemplificam tais símbolos:

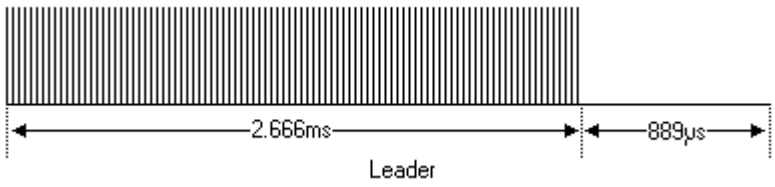

Figura 35: Leader Bit, SB Projects

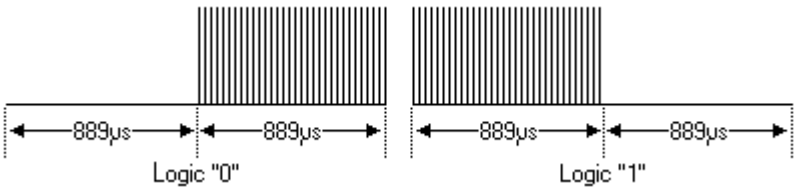

Figura 36: Trailer Bits, SB Projects

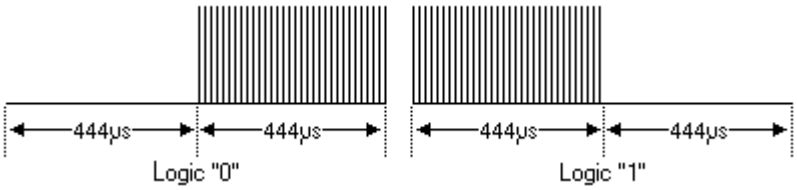

Figura 37: Bits Lógicos

Formato

A seguir será feita uma descrição sucinta do modo zero, usualmente reconhecido por televisores. A figura ilustra o bloco de mensagem:

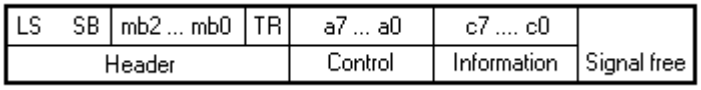

Figura 38: Mensagem RC6x em Modo 0, SB Projects

Cabeçalho

Inicialmente é transferido um leader bit, com o qual o ganho do receptor de infravermelho, o próximo bit é denominado start bit e vale sempre 1, sendo utilizado para calibrar o tempo no receptor. Três bits lógicos m2, m1 e m0 definem o modo de operação, neste caso todos valerão zero.
Por fim, um Trailer Bit determina o final do cabeçalho, servindo também de Toogle Bit, comportando-se como seu equivalente no protocolo RC5x.

Controle

Este campo conta com 8 bits para especificar o dispositivo a ser controlado, permitindo o controle de até 256 diferentes aparelhos.

Informação

Campo de 8 bits, permitindo que cada aparelho possa ter até 256 comandos diferentes.

Tempo Livre de Sinal (Signal Free Time, SFT)

Trata-se de um intervalo de 6t, 2.666ms, onde nenhuma informação é transmitida (por qualquer dispositivo). Sua presença no final da mensagem evita recepções incorretas.