UNIVERSIDADE POSITIVO
NÚCLEO DE CIÊNCIAS EXATAS E TECNOLÓGICAS
ENGENHARIA DA COMPUTAÇÃO

AMBIENTE PARA CONTROLE DE ELETRO-ELETRÔNICOS VIA DISPOSITIVOS MÓVEIS

Rafael Descio Trineto

Monografia apresentada à disciplina de Trabalho de Conclusão de Curso como requisito parcial à conclusão do Curso de Engenharia da Computação, orientada pelo Prof. Alessandro Brawerman.

UP/NCET
Curitiba
2008

TERMO DE APROVAÇÃO

Rafael Descio Trineto

Ambiente Para Controle de
Eletro-Eletrônicos Via Dispositivos Móveis

Monografia aprovada como requisito parcial à conclusão do curso de Engenharia da Computação da Universidade Positivo, pela seguinte banca examinadora:

Prof. Alessandro Brawerman (Orientador)

Prof. Mauricio Perretto

Prof. Marcelo Mikosz Gonçalves

Curitiba, 08 de dezembro de 2008.

AGRADECIMENTOS

Sobre tudo a Deus, que me deu forças para superar minhas limitações e superasse as adversidades encontradas no desenvolvimento desse projeto.

A minha mãe Vitoria dos Santos por ter me proporcionado a oportunidade de cursar engenharia, e estar apoiando e incentivando durante todo o curso.

A minha irmã Rafaela Descio pelo apoio durante todo o curso.

Ao meu professor e orientador Alessandro Brawerman pelo direcionamento nas pesquisas e incentivo no desenvolvimento.

A todos os professores que repassaram seus conhecimentos sem os quais não seria possível realizar este projeto.

A minha colega Maira Ranciaro pelas dicas no desenvolvimento do hardware.

Aos meus colegas Cristiano Costa, Eduardo Zuffo e Rafael França que me apoiaram durante o desenvolvimento deste projeto.

E a todos que de alguma forma colaboram para que fosse possível a conclusão desse projeto.

RESUMO

A criação de novas tecnologias para dispositivos móveis tem permitido o desenvolvimento de projetos integrando diversas áreas e conceitos. Partindo dessa premissa, este projeto apresenta um ambiente, conjunto de hardware e software, para controle remoto de aparelhos eletro-eletrônicos utilizando um programa instalado em um celular.

O sistema explanado neste trabalho é composto de um aparelho celular, um computador e uma interface de controle dos eletro-eletrônicos. A partir dessas interfaces o usuário pode escolher uma funcionalidade no celular para que uma ação seja disparada em um eletro-eletrônico.

A comunicação entre celular e computador é feita via Internet. Assim que a informação chega ao programa instalado no computador, ela é interpretada, gerando uma nova mensagem para execução na interface de controle dos eletro-eletrônicos.

Palavras chave: Celular, Eletro-eletrônico, Internet

ENVIRONMENT CONTROL FOR ELECTRO-ELECTRONICS BY MOBILE DEVICES

ABSTRACT

The creation of new technologies for mobile devices has allowed the development of projects integrating diverse areas and concepts. Based on that premise, this project presents an environment, set of the hardware and software, for remote control of electro-electronic devices using a program installed in a cell phone.

The system explained in this work is composed of a cell phone, a computer and an interface of electro-electronic control. From these interfaces the user can choose a functionality in the cell phone so that an action is gone off in an electro-electronic.

The communication between the cell phone and the computer are made by Internet. As soon as the information arrives at the program installed in the computer, it is interpreted, generating a new message for execution in the interface of control of the electro-electronic.

Keywords: Cell Phone, Electro-electronic, Internet

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS

NCET – Núcleo de Ciências Exatas e Tecnológicas

UP – Universidade Positivo

PDA – *Personal Digital Assistants* (Assistente Pessoal Digital)

CPU – *Central Processing Unit* (Unidade Central de Processamento)

ERP – *Enterprise Resource Planning*

ME – *Micro Edition*

MIDP – *Mobile Information Device Profile*

CLDC – *Connected Limited Device Configuration*

A/D – Analógico/Digital

D/A – Digital /Analógico

E/S – Entrada/Saída

I/O – *In/Out*

PC – *Personal Computer* (Computador Pessoal)

ROM – *Read Only Memory* (Memória Apenas de Leitura)

EPROM – *Erasable and Programable Read Only Memory* (Memória Apagável e Programável Apenas de Leitura)

RAM – *Random Access Memory* (Memória de Acesso Aleatório)

LED – *Light Emitting Diode* (Diodo Emissor de Luz)

P-N – Positivo-Negativo

ISDN – *Integrated Services Digital Network* (Rede Digital de Serviços Integrados)

ARM - *Advanced RISC Machine* (atualmente), *Acorn RISC Machine* (Antes)

RISC – *Reduced Instruction Set Computer* (Computador com um Conjunto Reduzido de Instruções)

USB – *Universal Serial Bus*

HTTP – *Hypertext Transfer Protocol* (*Protocolo de Transferência de Hipertexto*)

API – *Application Programming Interface* (Interface de Programação de Aplicativos)

PWM – *Pulse Width Modulation*

IP – *Internet Protocol*

LISTA DE SÍMBOLOS

Ω – Ohm

K – Quilo (10^3) unidades.

M – Mega (10^6) unidades

G – Giga (10^9) unidades

V – Volts

mV – Milivolt

A – Ampére

mA – Miliampére

mW – Miliwatts

h – Hexadecimal

bit – Dígito binário

byte – Conjunto de oito bits (octeto)

Mb – Mega bit

Gb – Giga bit

MB – Mega byte

GB – Giga byte

Mbps – Mega bits por segundo

nm – Nanômetro

CAPÍTULO 1 - INTRODUÇÃO

Com o aumento da capacidade de processamento e evolução dos sistemas operacionais para dispositivos móveis, houve um crescimento na demanda de softwares para celulares. Há muito a ser explorado nessa área de desenvolvimento, como exemplo, a comunicação com dispositivos à distância. A disponibilidade de Internet em celulares possibilita a comunicação com dispositivos em ambientes diferentes, permitindo a transferência de dados com muita agilidade e eficiência.

Com intuito de proporcionar comodidade ao usuário, este projeto visa elaborar um sistema de controle remoto aos diversos aparelhos eletro-eletrônicos que o usuário possua instalados em sua residência. Este sistema será instalado em um aparelho celular na forma de uma interface amigável, dispondo de diversas funcionalidades relativas aos eletro-eletrônicos a serem controlados.

O objetivo desse projeto é garantir o acesso a dispositivos eletro-eletrônicos via um aparelho celular tipo *smartfone*. Utilizando um software instalado no *smartfone* para a comunicação com um computador via Internet. O computador receptor interpreta as informações recebidas e aciona um dispositivo micro-controlado. Este dispositivo então fica responsável por executar as ações sobre os eletro-eletrônicos.

O restante do trabalho está dividido da seguinte maneira: o capítulo 2 apresenta os conceitos básicos para um melhor entendimento do projeto, o capítulo 3 descreve alguns trabalhos que possuem relação com este trabalho, o capítulo 4 cita os componentes utilizados para a construção desse projeto, o capítulo 5 demonstra os passos tomados para a conclusão do projeto, o capítulo 6 exibe os resultados obtidos e o capítulo 7 apresenta as conclusões e descobertas obtidas com os fatos analisados.

CAPÍTULO 2 - FUNDAMENTAÇÃO TEÓRICA

Este capítulo apresenta conceitos e tecnologias diversas que de alguma forma possuem ligação com esse trabalho e serviram de sustentação para os métodos adotados neste projeto.

2.1 - Domótica

Segundo (BRUGNERA, 2007), domótica é um recurso utilizado para controle de um ou mais aparelhos eletrônicos por meio de uma central computadorizada e tem o objetivo básico de melhorar a qualidade de vida, reduzir o trabalho doméstico e aumentar o bem estar e segurança. O termo surgiu da junção da palavra "*Domus*", que significa residência, com as palavras eletrônica e informática.

Esse projeto almeja automatizar aparelhos eletro-eletrônicos de forma que haja a possibilidade de controlá-los remotamente, mas com este controle agrupado em uma central computadorizada.

2.2 - Smartfone

Não há acordo entre fabricantes de celulares para definir atualmente um *smartfone*. "*Smartfones* diferem dos celulares habituais em dois aspectos fundamentais: como eles são feitos e o que eles podem fazer. A forma como eles são construídos – utilização de sistemas abertos que tiram partido das aptidões da energia e das inovações de inúmeras empresas pertencentes a uma vasta gama de indústrias – significa que os *smartfones* alargam o histórico de celulares, melhorando constantemente e rapidamente, ano a ano. Como o que eles podem fazer – de acordo com o "*fone*" parte do seu nome, *smartfones* fornecer todas as capacidades ordinárias dos celulares, em particular um estilo amigável para o usuário – mas isso é apenas o início." (SYMBIAN OS, 2008).

A maior parte dos *smartfones* possui um sistema operacional fácil de identificar, permitindo a inclusão de novos aplicativos, os quais podem ser desenvolvidos pelo fabricante do dispositivo, por um desenvolvedor terceirizado ou por operadoras de telefonia.

Em síntese um *smartfone*, é comparável a um PDA, mas com a diferença de ser centrado em voz, enquanto em um PDA é centrado em dados. Este projeto faz uso de um *smartfone* para que seja possível a comunicação remota com o servidor e a partir desse controlar os eletro-eletrônicos.

2.3 - Microcontrolador

Os primeiros microcontroladores – 8748 e 8048 – foram desenvolvidos pela Intel em 1976 (INTEL, 2008), a partir dessa época houve uma grande popularização desses dispositivos que devido sua constante evolução mantém-se em uso indústria até hoje.

Segundo (GIMENEZ, 2002), um microcontrolador é um computador programável, em um chip otimizado para controlar dispositivos eletrônicos. É uma espécie de microprocessador com memória e interfaces de E/S (I/O) integrados, enfatizando sua auto-suficiência, em contraste com um microprocessador de propósito geral, o mesmo tipo usado nos PC' s, que requer chips adicionais para prover as funções necessárias.

Por dispor de vários recursos este dispositivo é utilizado para solucionar os mais diversos problemas de controle. Atualmente existem microcontroladores que agregam recursos tais como: temporizadores, conversores A/D, conversores D/A, interface Serial e interface USB.

2.4 -Controle Remoto Infravermelho

Um grande número de dispositivos eletro-eletrônicos possui controle de suas funcionalidades a partir de dispositivos remotos infravermelho, tais como aparelhos de DVD, televisores, aparelhos de som, ar condicionados, portas, etc.

Estes dispositivos de controle funcionam com o uso de componentes – em sua maioria díodos emissores de luz – que emitem radiação infravermelha portadora de um código que será interpretado pelo dispositivo controlado.

A radiação infravermelha é uma radiação eletromagnética cujo comprimento de onda é maior do que o da luz visível. As radiações com comprimento de onda superior a 760nm são ditas infravermelhas (UFPA, 2008).

2.4.1 - Diodo Emissor de Luz

Diferente das lâmpadas incandescentes onde um filamento é aquecido para gerar luz, o Diodo Emissor de Luz ou LED emite luz quando os elétrons, excitados pela energia elétrica a partir de suas ligações químicas, rearranjam estas ligações, liberando a energia em forma de luz. O processo de emissão de luz em um LED é muito mais rápido e eficiente que em uma lâmpada incandescente onde a maioria da energia é desperdiçada com calor. A alta eficiência, rápido acionamento/desligamento, e baixa emissão de calor do LED trazem ao LED várias aplicações novas (WISC, 2008).

2.4.2 - Protocolo EXTENDED-NEC

O protocolo adotado como padrão nesse projeto é uma adaptação do protocolo NEC, conhecido com o EXTENDED-NEC, desenvolvido pela NEC Corporation.

Este protocolo é transmitido sobre uma onda com freqüência de 38kHz e 33% de *duty-cycle*, gerando as informações a partir da pulsação dessa onda. Sua formação principal possui um *header bit* de 13,5ms seguido de 16 bits de endereçamento e 16 bits de comando. Esses bits possuem dois estados lógicos: alto e baixo; onde estes possuem tempos de 2,24ms e 1,12ms, respectivamente (KNOWLEDGE BASE, 2008). A Figura 2.1 demonstra a formação dos bits que compõem esse protocolo. É possível perceber que para cada estado lógico apenas por 0.56ms a onda de 38kHz é transmitida.

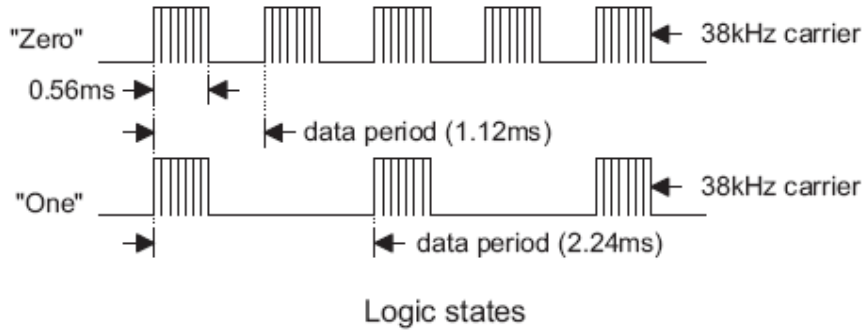

Figura 2.1 – Estados lógicos do protocolo EXTENDED-NEC

A Figura 2.2 mostra a formação completa de uma transmissão padrão. É possível perceber o *header bit* no início da transmissão – no sentido da esquerda para a direita – formado por

seus 9ms com onda de 38kHz e 4,5ms com o sinal constante, e posteriormente os 16 bits de endereçamento seguidos dos 16 bits de comando. As informações dos comandos são transmitidas utilizando apenas 8 bits de dados os últimos 8 bits são a negação lógica dos bits de dados.

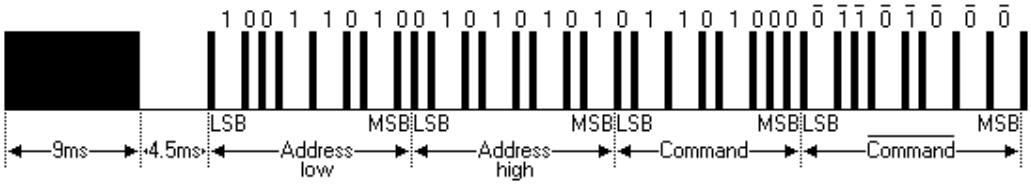

Figura 2.2 – Saída padrão de um comando segundo o protocolo EXTENDED-NEC

A Figura 2.3 exibe primeiramente uma saída padrão completa com duração de 108ms. O segundo comando de 108ms – da esquerda para direita – na Figura 2.3 é interpretado pelo receptor como um comando de repetição do comando anterior, ele é formado por um pulso inicial de 9ms seguido de um intervalo de 2,5ms e um pulso de 0,56ms (HOLTEK,2008).

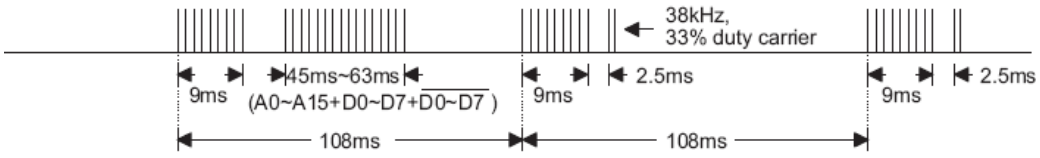

Figura 2.3 – Comando padrão seguido de um comando de repetição.

2.5 - Modelo Cliente/Servidor

Segundo (TANENBAUM, 2003), esse modelo é amplamente usado e constitui a base de grande utilização da rede. Ele é aplicável quando o cliente e o servidor estão ambos no mesmo edifício, mas também quando estão muito distantes um do outro.

Nesse modelo há dois processos envolvidos, um na máquina cliente e um na máquina servidora. A comunicação toma a forma do processo cliente enviando uma mensagem pela rede ao processo servidor. Então, o processo cliente espera por uma em resposta. Quando o processo recebe a solicitação, ele executa o trabalho solicitado ou procura pelos dados solicitados e envia de volta uma resposta.

2.6 - Plataforma Java, Micro Edition (J2ME)

Plataforma Java, Micro Edition (Java 2 ME) fornece um ambiente robusto e flexível para aplicativos executados em dispositivos móveis e outros dispositivos embarcados – telefones celulares, assistentes pessoais digitais (PDAs), *TV set-top boxes*, e impressoras.

Java ME inclui interfaces com usuário flexíveis, segurança robusta, acoplamento de protocolos de rede, e suporte para rede e aplicações off-line que pode ser baixado dinamicamente. Aplicativos baseados em Java ME são portáteis em muitos dispositivos e ainda alavancam suas capacidades nativas (SUN MICROSYSTEMS, 2008a).

Segundo (SUN MICROSYSTEMS, 2008b), plataforma Java ME é uma coleção de tecnologias e especificações que podem ser combinados para construir um ambiente Java *Runtime* completo, especificamente para se adequar às exigências de um determinado mercado ou dispositivo. Isto oferece uma flexibilidade e co-existência para todos os intervenientes no eco-sistema cooperando continuamente para oferecer a mais atraente experiência para o usuário final.

Uma configuração proporcionada pela plataforma Java ME é a *Connected Limited Device Configuration* (CLDC) que é especificamente concebido para satisfazer as necessidades de uma plataforma Java que roda em dispositivos com limitadas memória, capacidade de processamento e capacidades gráficas. Um exemplo amplamente adotado é combinar a CLDC com o *Mobile Information Device Profile* (MIDP) para fornecer um ambiente completo para aplicações Java para telefones celulares, como desse projeto, e outros dispositivos com capacidades semelhantes (SUN MICROSYSTEMS, 2008b).

CAPÍTULO 3 - TRABALHOS REALICIONADOS

Muitos trabalhos e pesquisas seguem os mesmo conceitos abordados por este projeto, alguns destes trabalhos são discutidos nesse capítulo com o objetivo de traçar um paralelo entre as técnicas adotadas nesse projeto e as técnicas usadas em outras instituições acadêmicas, e como a união e aperfeiçoamento dessas técnicas podem gerar novas de pesquisas.

Em (UNIFEI, 2008), os autores descrevem um sistema cujo objetivo é observar e aprender regras em uma casa de acordo com o comportamento de seus habitantes. O sistema utiliza o conceito de aprendizado com regras de indução, o qual já foi utilizado em outro sistema, conhecido como Sistema ABC (Automação baseada em comportamento). No sistema proposto, o Sistema ABC+, são apresentados, entre outros avanços, a janela de observação de eventos e as regras embrionárias.

O projeto desenvolvido no artigo reúne inúmeros conceitos de automação residencial com o objetivo de criar regras para ação de atuadores em função do aprendizado com o comportamento dos habitantes de uma casa, dando ênfase à obtenção, tratamento e manutenção das regras (UNIFEI, 2008).

Um exemplo citado é de um habitante entrando em um quarto que imediatamente acende a lâmpada ou quando sai do quarto e imediatamente desliga a lâmpada, um conjunto de regras são criadas e relacionadas para que seja possível saber antecipadamente se o habitante deseja manter a lâmpada acessa durante um determinado tempo, ou se ele deseja acender a lâmpada imediatamente após entrar no quarto, e então na próxima vez que este habitante entrar neste quarto o sistema dispara um conjunto de atuadores que anteciparão a ação do habitante.

Sistemas como o proposto nesse artigo, podem ser agregados a projetos de automação como o desenvolvido nesse trabalho. Isso poderia, por exemplo, fazer com que um sistema identificasse que uma pessoa que está se aproximando de sua casa com seu celular deva ter o acesso liberado.

Outro trabalho relevante foi apresentado em (UNIBRASIL, 2008). Seu principal objetivo é o desenvolvimento de um sistema em dispositivos móveis para automatização do processo de venda e atendimento aos clientes, independente de sistemas operacionais.

O projeto citado foi desenvolvido com base nas linguagens de programação Java e XML. O protótipo foi desenvolvido com o objetivo de utilizar uma das características principais da

linguagem JAVA, a portabilidade, bem como a sua utilização no desenvolvimento de aplicativos para dispositivos móveis. O paradigma adotado para o desenvolvimento do sistema foi o Orientado a Objetos assim como neste projeto.

Foram desenvolvidos dois módulos, um Módulo Desktop responsável pela sincronização dos dados do ERP com o PDA, e um Módulo PDA ,sendo este um aplicativo que executa no PDA do vendedor e a partir dele que o vendedor tem acesso aos dados do sistema ERP e pode emitir pedidos de venda.

Para a troca de informações utilizou-se a linguagem XML, que montava um arquivo um arquivo nos padrões DOM (*Document Object Model*) e transmitia entre servidor FTP (*File Transfer Protocol*) e o PDA e vice-versa.

A grande vantagem da adoção do XML é que para a transmissão de grande quantidade de dados sua compreensão e manipulação são simplificadas, além de permitir adicionar e extrair informações facilmente e ser suportada por um grande número de linguagens de programação. Como desvantagem tem o acréscimo de dados sem valor relacionado à informação final, que servem apenas como orientadores para a manipulação do pacote XML.

Ao contrário do projeto citado (UNIBRASIL, 2008), nesse projeto a utilização do padrão XML foi descartada, levando-se em conta o número limitado de dados a serem transmitidos. Além disso, reduzir o número de dados transmitidos em cada pacote também significa reduzir o custo com serviços de transmissão de dados.

Outro trabalho significativo foi apresentado em (UFLA, 2008), no qual os autores descrevem o desenvolvimento de uma ferramenta com uso de tecnologias móveis para auxílio no ambiente de dieta dos pacientes e oferecimento de diversos recursos ao nutricionista, a partir da coleta de informações sobre alimentos consumidos no momento da alimentação, disponibilizando estas informações aos Nutricionistas no momento das consultas ou em qualquer outra ocasião.

O projeto citado foi desenvolvido com base na plataforma Java por permitir desenvolver aplicações altamente portáveis entre uma determinada classe de dispositivos móveis, aproveitando as vantagens de conectividade desses dispositivos.

Os testes foram realizados nos simuladores disponíveis no *Wireless Toolkit* da Sun e os resultados obtidos favoreceram enormemente a sua implantação na prática. Os requisitos de

software necessários permitiam comercializar o sistema utilizando apenas softwares com versões gratuitas sem prejuízo de desempenho.

Foram verificadas carências de recursos (principalmente memória e processamento) dos dispositivos móveis podem comprometer o tempo de resposta do sistema para o usuário e, conseqüentemente, inviabilizar a sua utilização (UFLA,2008).

Com a evolução que os aparelhos celulares tiveram nesses últimos anos, assim como com o aumento de capacidade de transmissão de dados das redes sem celulares, este projeto tornar-se-ia viável. Isso demonstra como a evolução dinâmica das tecnologias móveis tem acelerado a viabilidade de projetos que necessitem de grandes quantidades de processamentos em dispositivos móveis.

Outro artigo significativo é descrito por (IEEE COMPUTER SOCIETY,2007). Nele os autores apresentam um ambiente inteligente chamado SmartOffice. A partir do monitoramento de usuários, SmartOffice antecipa a intenção dos usuários e aumenta a comunicação de informações úteis.

Para os desenvolvedores, o SmartOffice oferece um ambiente de teste para módulos colaborativos e combinados em uma aplicação coerente, no qual a integração requer um ambiente de trabalho flexível em que cada módulo desenvolvido não se preocupe com o módulo de comunicação baixo-nível. Este artigo apresenta um protocolo de integração flexível orientado a recursos, que se pensa ser necessário para criar tal ambiente. O módulo de comunicação, usando um protocolo baseado em XML, com um supervisor que age como servidor de recursos. O supervisor foi programado usando linguagem baseada em regras. Para ilustrar o papel do supervisor, foi utilizado dois dos principais módulos do SmartOffice: o MagicBoard e um módulo de localização de usuário.

SmartOffice compreende 50 sensores (câmeras e microfones) e três atuadores (um vídeo projetor e duas caixas de som). O supervisor controla inúmeros módulos independentes. O MagicBoard é o principal "atuador" que o SmartOffice pode utilizar para aumentar o ambiente do usuário. Previsão das intenções do usuário exige um módulo de monitoramento de usuário. Figura 3.1 mostra instalação completa do SmartOffice (IEEE COMPUTER SOCIETY, 2008).

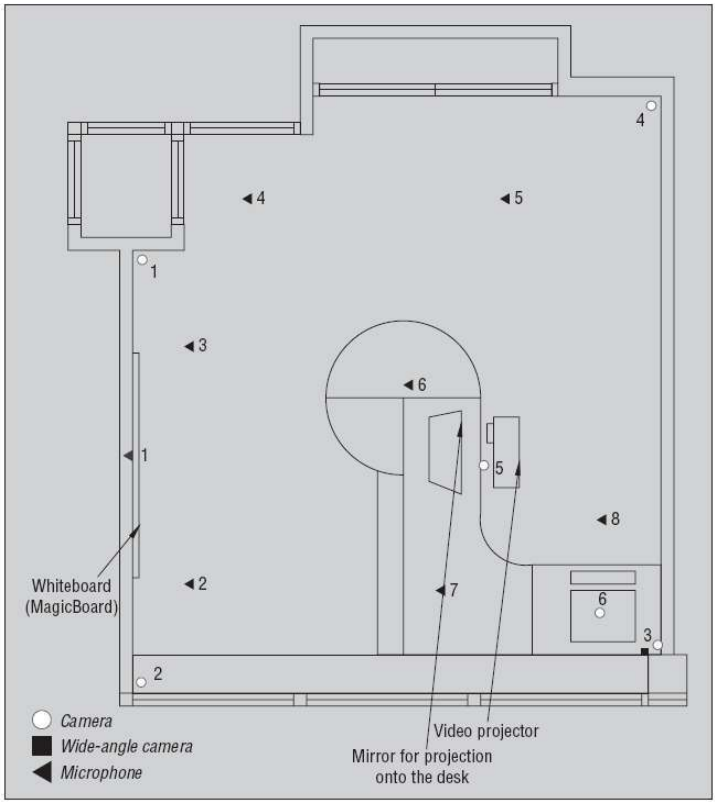

Figura 3.1 – Ambiente SmartOffice

Projetos como SmartOffice mostram que a automação residencial tem recebido grandes investimentos em pesquisas. Em alguns países, projetos como esses padrões já são realidade para os consumidores, tornando-se cada vez mais comuns.

A união de projetos estilo SmartOffice com projetos de comunicação móvel trazem grandes perspectivas de desenvolvimento. Em pouco tempo será possível monitorar e interagir com um ambiente a quilômetros de distância. A utilização de celulares, como a descrita nesse projeto, é uma das alternativas para distribuir e acessar remotamente sistemas no formato do SmartOffice.

Em (DOHMS, 2008) o autor apresenta o desenvolvimento de um circuito simples que através de informações enviadas pelo celular, PC (*Personal Computer*) ou PDA (*Personal Digital Assintent*) liga ou desliga as lâmpadas de uma residência.

Uma dos benefícios deste projeto que também está presente no apresentado nesta dissertação é a portabilidade, já que o sistema cliente pode ser executado em ambientes de arquitetura de computação diferentes, como por exemplo, PC, PDA ou celular. Outro ponto positivo, que também será adotado para implementação deste projeto, é o uso de sistemas *open source*, ou

seja, o custo de desenvolvimento sofre grande redução, tornando o sistema economicamente viável.

Entretanto, como o circuito proposto em (DOHMS, 2008) é simplificado, só é possível o controle de equipamentos do tipo liga/desliga, por exemplo, lâmpadas. Já neste projeto, o uso de microcontrolador permite que haja o controle de equipamentos com mais estados, além de ligar e desligar, este poderá controlar a velocidade de um motor de batedeira, ou a potência de um forno de microondas, ou até mudar o canal de TV e aumentar ou diminuir o volume da mesma.

Em (JU *et. al*, 2007), o autor descreve o uso do celular no controle e monitoramento de um robô, o robô utilizado na demonstração dos resultados foi denominado de ROBO-E. O controle do robô, assim como seu monitoramento, é feito em tempo real através de uma rede sem fio, da Internet e do celular, utilizando tecnologia 3G (Terceira Geração de Celular). Os comandos podem ser ativados através do acionamento manual ou automático, tornando possível a captura e envio das imagens, bem como o envio dos comandos de controle para o ROBO-E. As imagens obtidas no percurso são transmitidas e visualizadas através do celular.

As tecnologias utilizadas no projeto ROBO-E, como Internet, rede sem fio e celular também são utilizadas nesse projeto. A diferença está no equipamento que será controlado através do celular. Enquanto que no ROBO-E o objeto controlado trata-se de um robô, neste projeto os objetos de controle serão equipamentos eletro-eletrônicos de uma residência, entre outros. A demonstração das funcionalidades deste projeto, também será a partir de protótipos, uma característica muito forte encontrada em (JU *et. al*, 2007).

CAPÍTULO 4 - ESPECIFICAÇÃO DO PROJETO

O objetivo deste projeto é permitir que usuários acessem dispositivos eletro-eletrônicos remotamente via um aparelho celular. Para tal uma arquitetura cliente/servidor é adotada.

O sistema consiste em três módulos, um servidor instalado em um microcomputador, uma central de controle para os eletros-domésticos e um software instalado em um *smartfone*.

O software no *smartfone* envia via Internet os comandos selecionados pelo usuário para o servidor. Após interpretar os dados recebidos, o computador os transmite para central de controle que executa o comando recebido.

Descrição dos módulos:

- Software do *Smartfone* – Programa de interface com o usuário do sistema.

- Servidor (Computador) – Transmite os comandos recebidos do *smartfone* para o centro de controle.

- Centro de controle – Composto por um microcontrolador ligado ao computador e um sistema de controle infravermelho.

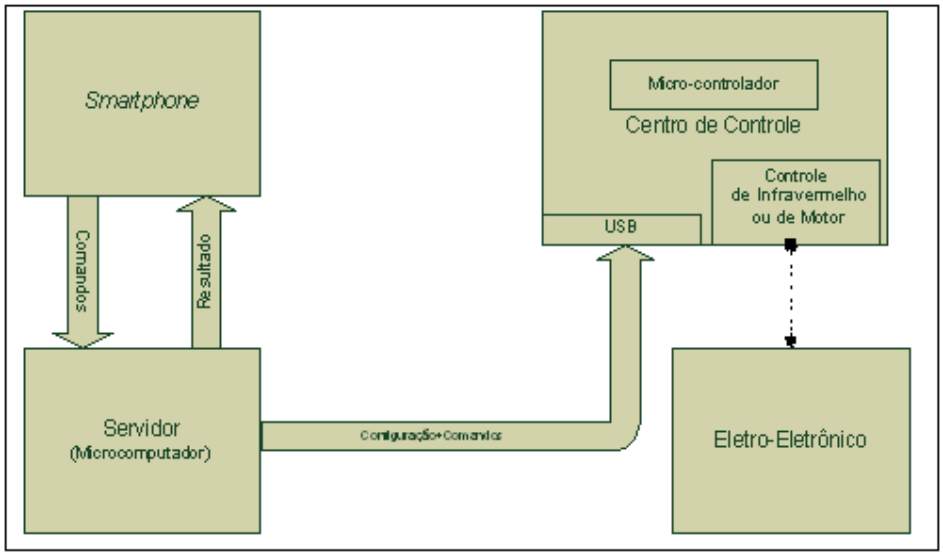

Figura 4.1 - Diagrama em blocos da estrutura física.

Como se observa na Figura 4.1, o *smartfone* envia o comando para o servidor que os interpreta e retorna um resultado. O servidor transmite as informações interpretadas do comando para o centro de controle via porta USB, e ao chegarem ao centro de controle são lidas pelo microcontrolador que dispara uma ação no controle infravermelho ou no controle de motor.

4.1 -Especificação do Hardware

O hardware implementado para esse projeto pode ser dividido em 3 partes: o celular que faz a interface com o usuário; o computador mantém o servidor executando; e o centro de controle que fica ligado ao computador para realizar interface com os eletro-eletrônicos.

A Figura 4.2 mostra o diagrama em blocos do hardware que será montado para o centro de controle. Três componentes importantes do centro de controle a serem citados são: conversor USB-Serial que faz a conversão dos dados recebidos da USB do computador para entrada serial do microcontrolador; o Kit Microcontrolador 8051 que interpreta os dados recebidos do computador; e o as placas de controle de eletro-eletrônicos que recebe comandos do microcontrolador e os efetua nos eletro-eletrônicos.

Figura 4.2 – Diagrama em blocos do hardware que será desenvolvido.

O acionamento remoto será feito em um aparelho celular da marca Nokia (Figura 4.3). Aparelho celular tipo *smartf*one. Dispõe de dual CPU de 332 MHz de clock, tipo ARM 11. Memória interna de 160Mb e suporte a cartão de memória adicional de até 4Gb. O sistema operacional instalado é Symbian 9.2 com suporte Java MIDP (Mobile Information Device Profile) 2.0, CLDC (Connected Limited Device Configuration) 1.1. Além disso, dispõem de um display de 240 x 320 pixels de resolução e profundidade de cores de 24bits. (Nokia, 2008a)

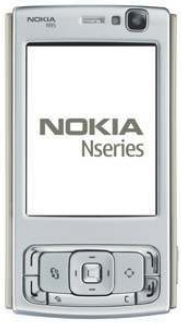

Figura 4.3 – *Smartfone* Nokia 95

O Kit Microcontrolador 8051 foi desenvolvido na Universidade Positivo com intuito de motivar e facilitar o desenvolvimento de aplicações utilizando um microcontrolador da família 8051(Figura 4.4).

Figura 4.4 – Kit Didático Microcontrolador 8031

Utiliza um microcontrolador 8031 que faz parte da família de microcontroladores 8051. Esta escolha foi feita porque apenas 4kbytes de memória ROM dos 8051 seriam insuficientes precisando então de uma ROM externa. Optou-se pelo microcontrolador 8031, que não contem memória ROM interna, porém tem menor custo.

Na placa é possível endereçar 32Kbytes posições de memória, visto que são utilizadas memórias RAM (62256) e EPROM (27C256) que possuem organização de 32Kbytes endereços X 8bits.

É possível também endereçar um hardware externo utilizando a expansão de 16 bits, neste caso, pode-se desabilitar a memória RAM da placa utilizando o endereço acima de 7FFFh,

pois a partir desse endereço o bit A15 ficará em 1 levando o pino /CE da memória RAM para o estado desabilitado do chip.

A placa ainda dispõe de entradas externas para os *timers* (*Timer* 0 e *Timer* 1) , entradas externas de interrupção(/INT0 e /INT1), pinos de RX e TX para comunicação serial e pinos de /RD e /WR informando que operação está sendo executada pelo microcontrolador (se escrita ou leitura na memória externa) (UNIVERSIDADE POSITIVO,2003).

Também é utilizado um conversor USB-Serial que consiste em uma ligação entre a porta USB do microcomputador e dispositivos seriais, realizando toda a conversão automaticamente. A utilização deste dispositivo permite que a porta USB seja usada como uma porta Serial RS-232.

O conversor USB / Serial é a mais rápida solução para utilização de periféricos com porta serial com uma forma simples de conectá-lo. Este permite conexão com modem, PDA, câmera digitais, impressora ou adaptadores com terminal ISDN com taxas de transferência superior a 1Mbps (Clone, 2008).

Além disso, para este projeto será utilizado um LED emissor TIL32 da Texas Instruments com 0.5mW de potência, alimentado por uma tensão de até 1.6V e corrente de 20mA. Sendo uma versão com embalagem plástica de baixo custo (Figura 4.5).

Figura 4.5 – Diodo Emissor de Luz Infravermelha

4.2 -Especificação do Software

Os softwares do sistema são separados em três partes, um software de interface com o usuário que é instalado no *smartfone*, um no computador que conecta o servidor e um firmware no microcontrolador que gerencia os dados recebidos.

A Figura 4.6 exibe a iteração entre os componentes do sistema representados em um diagrama de casos de uso. O usuário interage apenas com o celular, acessando o programa, escolhendo o aparelho e comando que deve ser executado e escolhendo o momento em que essas escolhas serão transmitidas. Fica como função do celular codificar estas informações e repassá-las ao servidor.

O servidor por sua vez tem a função de interpretador dos comandos recebidos do celular e transmissor dos comandos para a interface de controle.

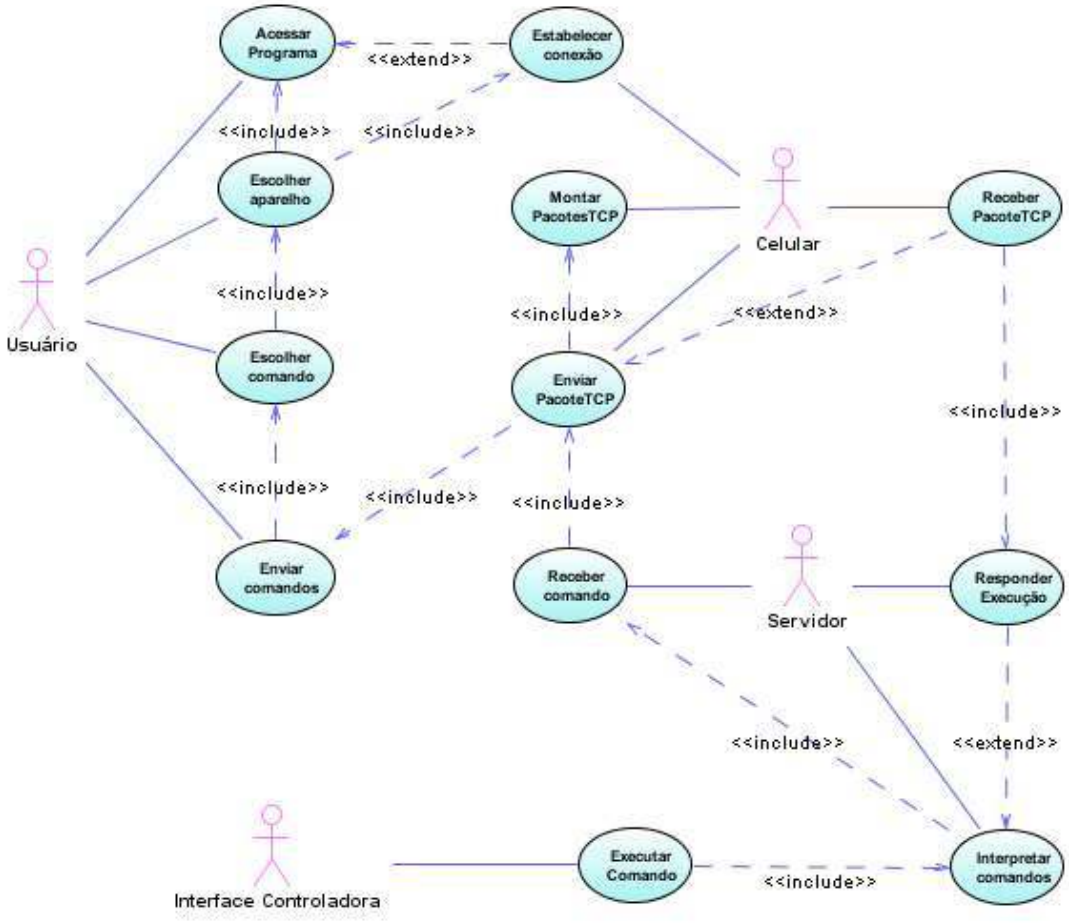

Figura 4.6 – Diagrama de casos de uso do sistema.

4.2.1 - Software do Smartfone

Consiste em uma interface gráfica, que exibe na tela as configurações possíveis dos eletro-eletrônicos. E em seguida, quando confirmado, envia as informações ao servidor através

de um protocolo próprio. Será desenvolvido usando J2ME, rodando em uma K *Virtual Machine*, desenvolvida para processadores de 16/32-bits (RIGGS,2001) e dispositivos com o mínimo de memória de 168Kbytes, 136Kbytes de memória não volátil para o MIDP e dados do aplicativo, 32Kbytes de memória volátil para o sistema Java (MORRISON, 2001).

O fluxograma do software do *smartfone* é representado na Figura 4.7 demonstrando quais são os passos imprescindíveis que o software deve executar para que o usuário consiga efetuar uma ação sobre os eletro-eletrônicos. A seqüência do fluxo pode ser resumida como: conexão com o servidor, envio de informações, recebimento de resultados e exibição dos resultados.

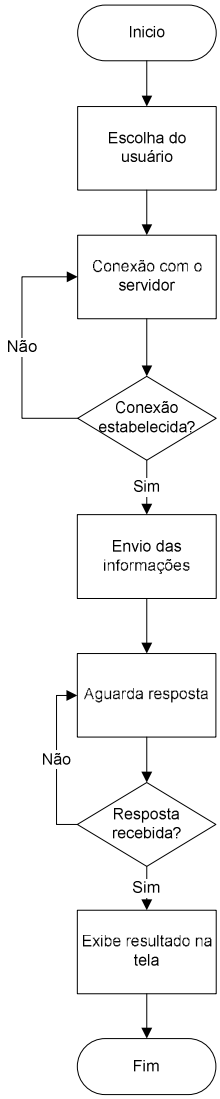

Figura 4.7 – Fluxograma do Software do *Smartfone*

A Figura 4.8 traz o diagrama de classe do software do *smartfone*, na qual é possível observar uma divisão em objetos de interface visual – classes *Visual* e *ControleRemoto* – , objetos de interface de conexão – classes *ClienteTCP*, *Pacote*, *PacoteEnvio* e *PacoteResposta*. Também pode se observar que as classes *PacoteEnvio* e *PacoteResposta* são classes 'filhas' da classe *Pacote*, pois herdam todos os métodos e atributos não privados da classe *Pacote*.

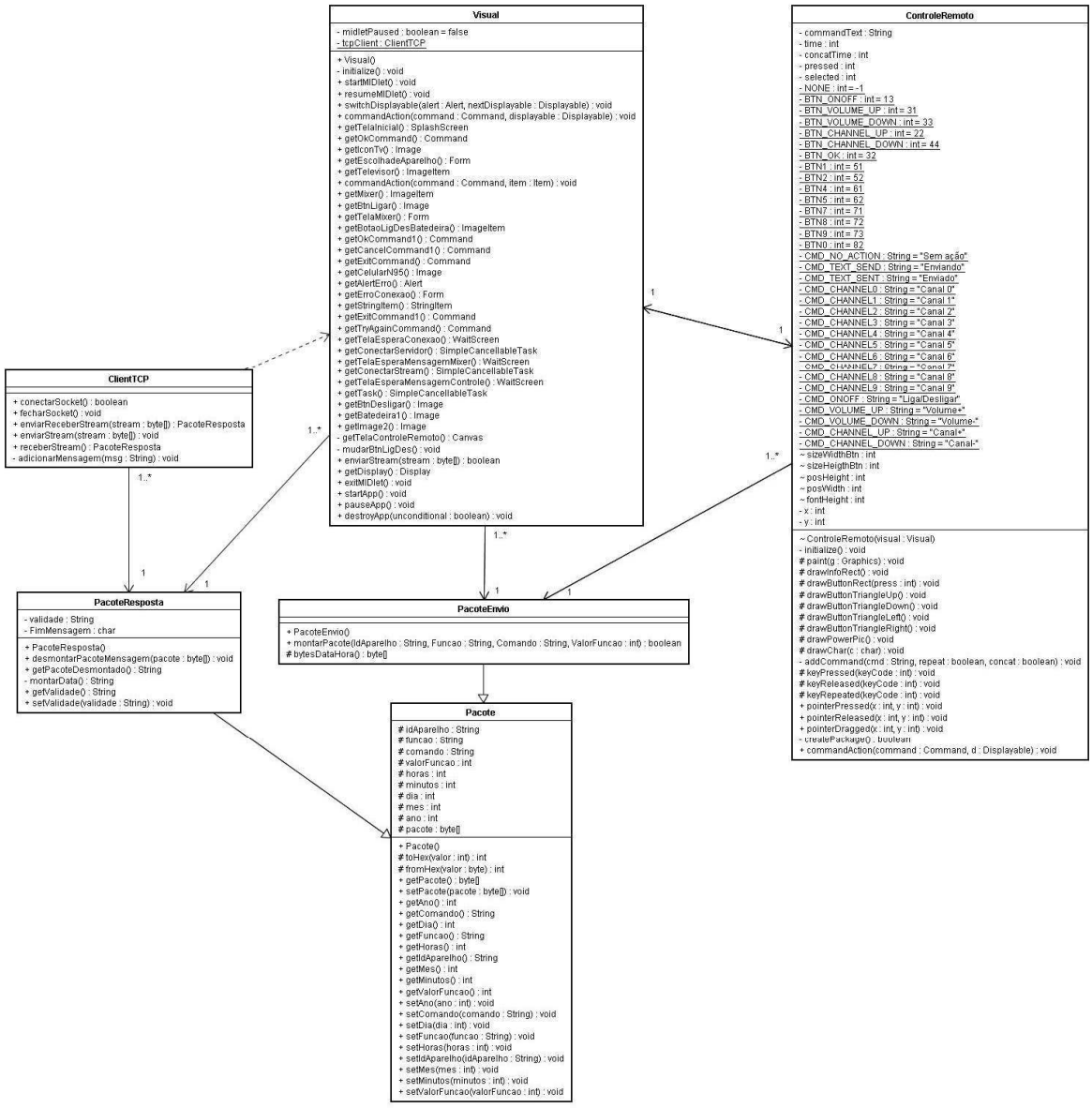

Figura 4.8 – Diagrama de classe do software do *smartfone*

A Figura 4.9 explana o diagrama de seqüência do software do *smartfone* a partir do momento que a conexão com o servidor é estabelecida, demonstrando em quais momentos e métodos os

objetos de conexão, neste caso as variáveis Objeto 1, Objeto 2 e Objeto 3, interagem entre si e com o objeto *Visual*.

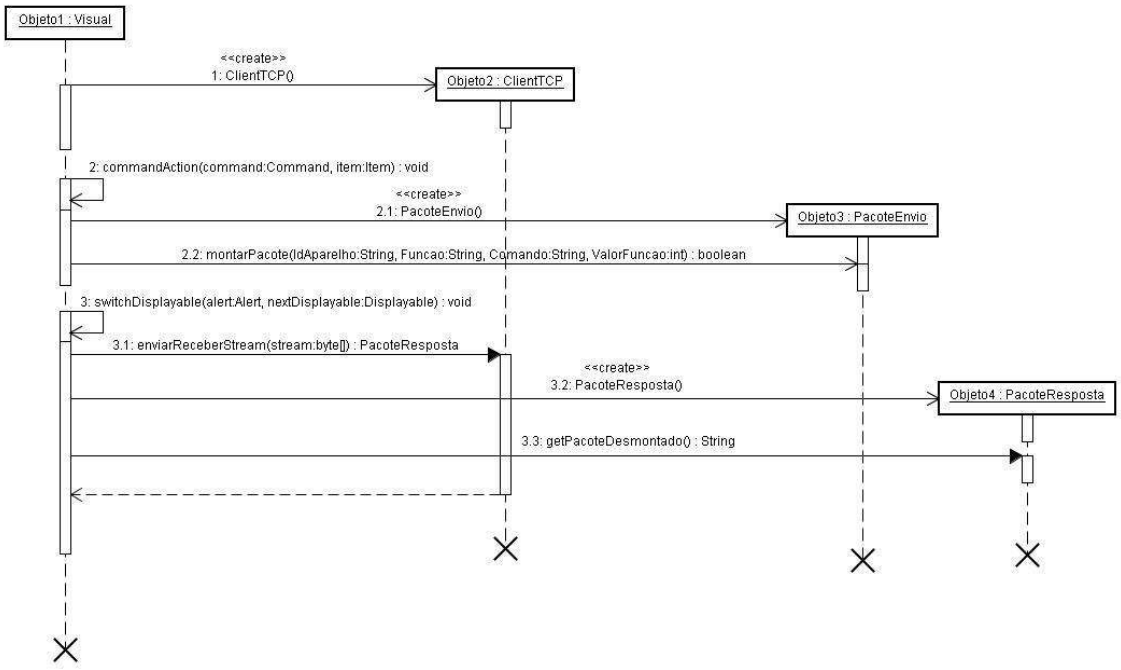

Figura 4.9 – Diagrama de Seqüência do software do *smartfone*

4.2.2 - Software do Servidor

O servidor é responsável por aguardar uma comunicação do *smartfone*. Quando uma conexão é realizada os dados são analisados e caso estejam corretos são passados ao microcontrolador, que envia uma resposta de recebimento ao *smartfone*.

A Figura 4.10 exibe o fluxograma do servidor, explicando como ocorre o fluxo de processos no servidor. O primeiro processo do fluxo é aguardar a conexão de um celular até que uma comunicação estabelecida, em seguida aguarda recebimento das informações. Assim que uma informação é recebida, ela passa por uma validação, caso seja válida a ação requerida é enviada ao microcontrolador e então respondida positivamente ao *smartfone*, do contrário uma resposta de erro é transmitida ao *smartfone*.

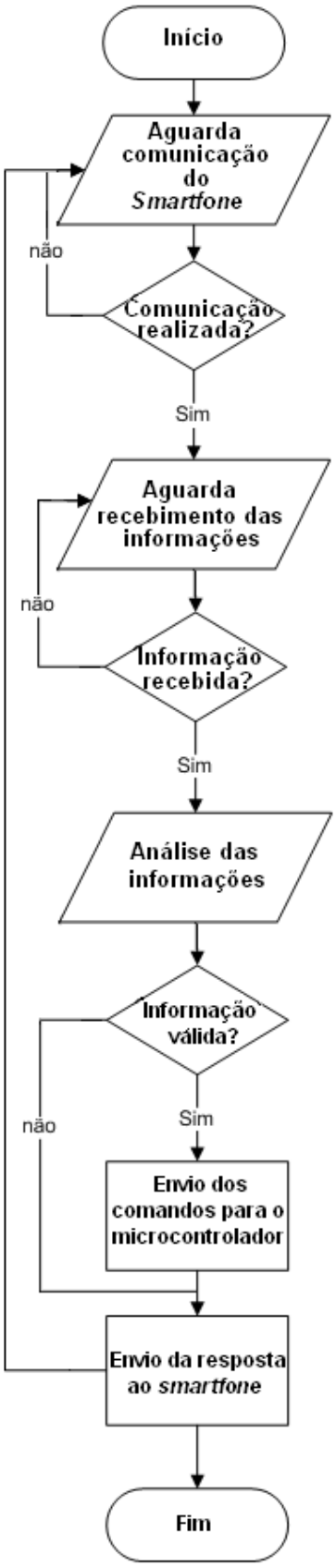

Figura 4.10 – Fluxograma Servidor

A Figura 4.11 exibe o diagrama de classe do servidor explicando a divisão das classes em parte visua: *SmartEletroGUI*, *TelaSobre* e *PortRequestDialog*; a parte de comunicação *Socket*: *Servidor*, *ClienteTCP*, *Pacote*, *PacoteEnvio* e *PacoteResposta*; e a parte de comunicação serial: *CommSerial*, *PacoteSerial*, *PacoteTV* e *PacoteBate*. Além de expressar como essas classes se relacionam entre si.

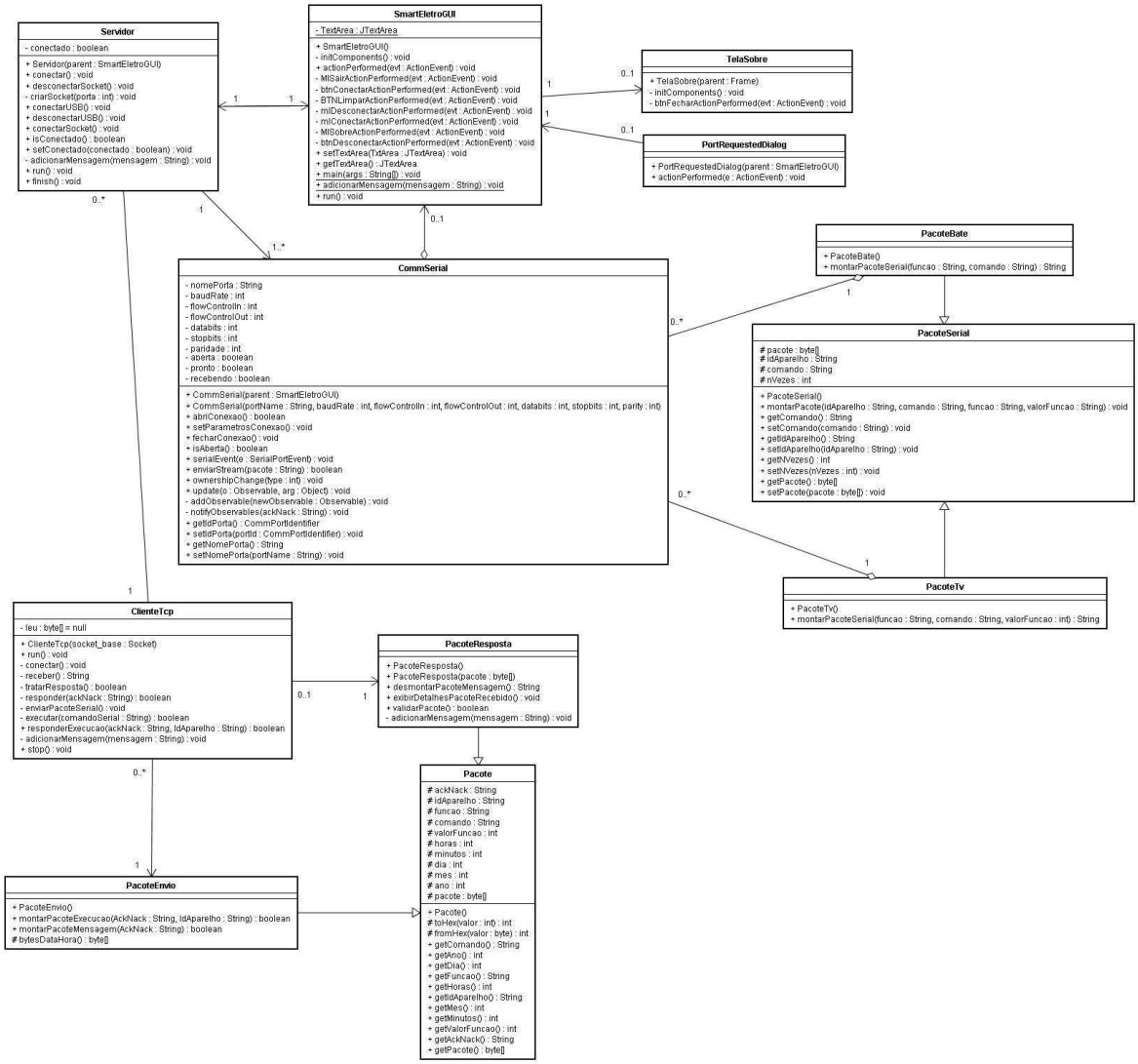

Figura 4.11 – Diagrama de classe do servidor

A Figura 4.12 é a representação do diagrama de seqüência do servidor desde o momento em que o servidor é iniciado até o momento de resposta de uma solicitação do *smarfone*. Por meio desse pode-se observar a seqüência de criação dos objetos e os métodos invocados por eles.

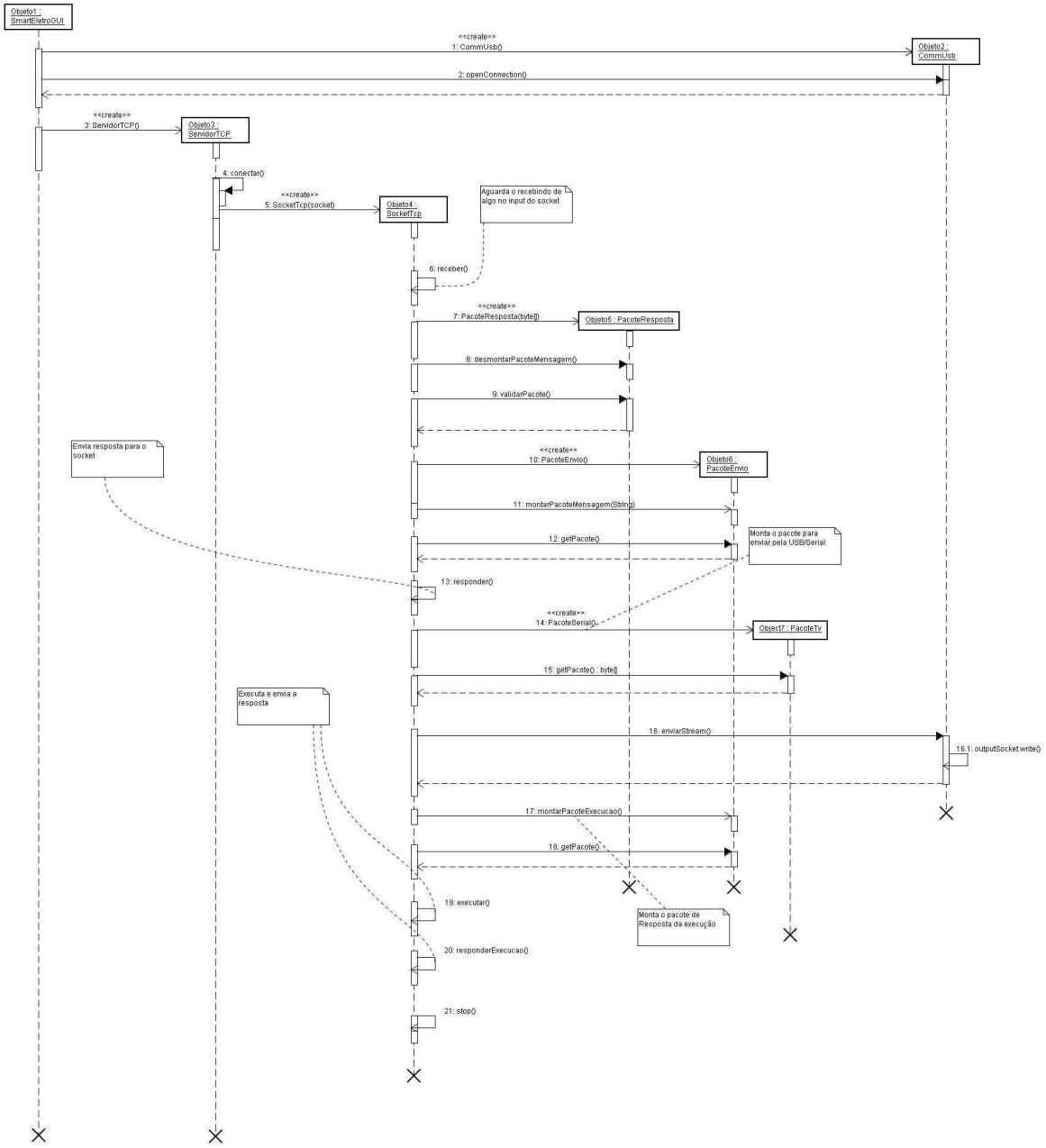

Figura 4.12 – Diagrama de seqüencia do servidor.

4.2.3 - Firmware

O firmware é responsável por distribuir e executar os comandos enviados pelo *smartfone* ao servidor.

A Figura 4.13 mostra o fluxo da informação repassada pelo servidor e como após esta ser analisada é enviada para o dispositivo correspondente.

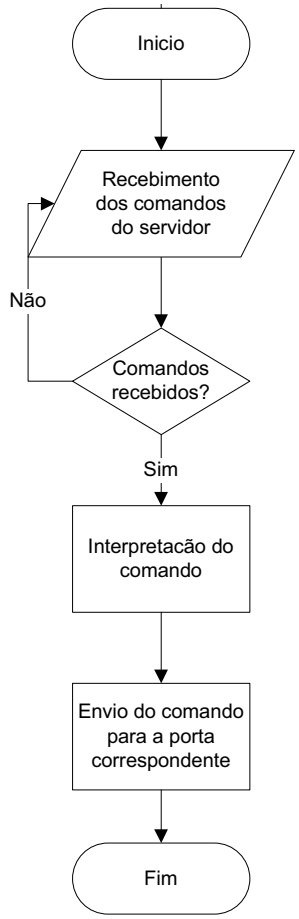

Figura 4.13 – Fluxograma do Firmware

Há cinco estados que o firmware pode assumir: configuração de parâmetros; espera de comando; validação de comando; execução de comando; e envio de resposta ao servidor. A Figura 4.14 exibe o diagrama de estados do firmware, demonstrando como eles interagem entre si. Como pode se perceber no diagrama, o estado 'configurar comando' é processado apenas uma vez, isto ocorre no momento em que o firmware é carregado pela primeira vez. Já os outros estados repetem-se cada vez que um novo comando chega.

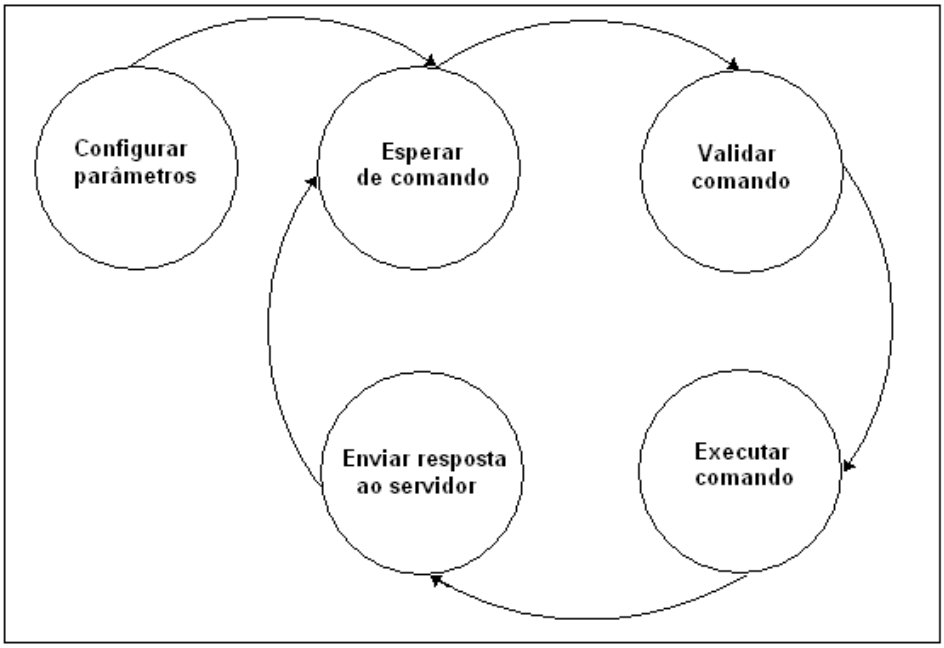

Figura 4.14 – Diagrama de estados da comunicação serial no firmware

4.2.4 - Protocolo de comunicação *Smartfone*-Servidor

A seguir é descrito a forma de comunicação entre o telefone celular (*Smartfone*) e o servidor TCP. Para fácil compreensão um exemplo de utilização é mostrado com um dispositivo específico, mas nada impede que outros dispositivos sejam adicionados e manipulados através desse protocolo.

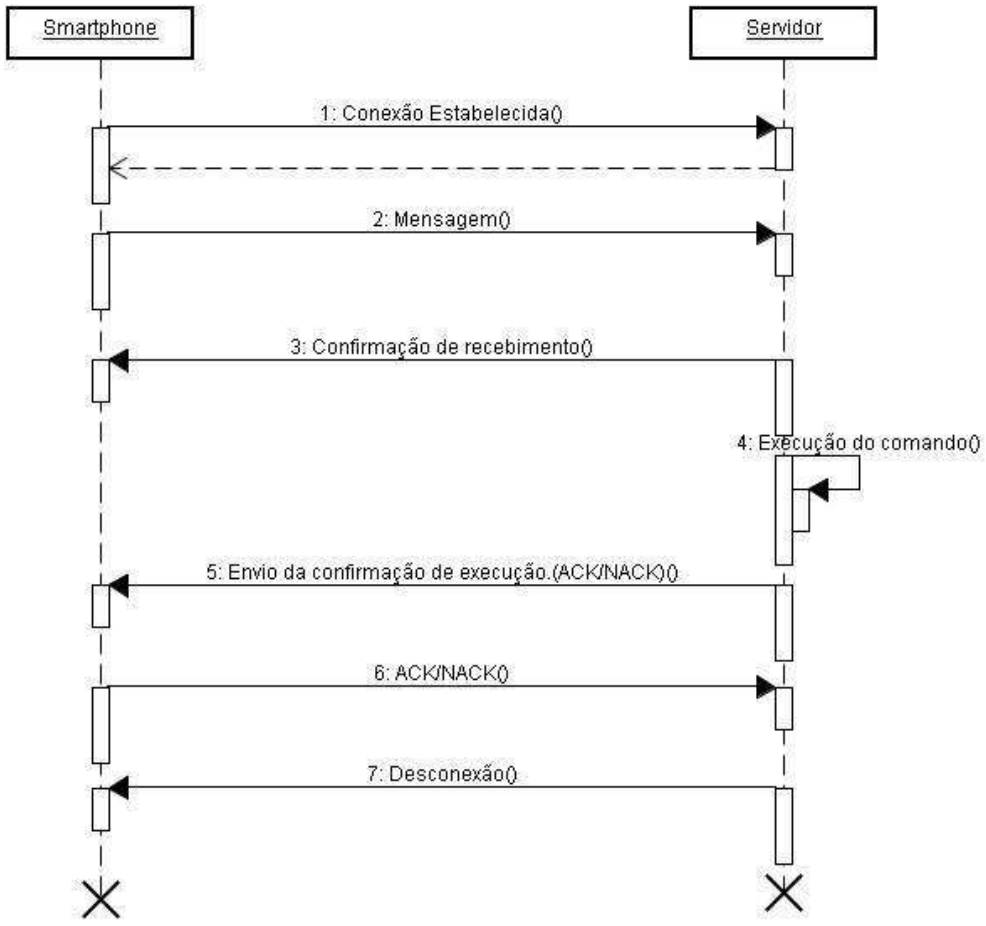

Figura 4.15 – Diagrama de Seqüencia da Comunicação *Smartfone*-Servidor

4.2.4.1 - Mensagem do *Smartfone*

A mensagem do *Smartfone* é composta de 20 bytes, como segue:

TABELA 1 - Protocolo de comunicação e número de bytes

Descrição	Nº de Bytes
ID do Aparelho	6
Identificação da função	4
Comando	4
Valor da função	1
Data	5

Abaixo é descrito cada byte, como é transmitido e como deve ser feita a conversão para manipulação dos dados.

TABELA 2 - Descrição de cada Byte e tipo de conversão

Byte	Descrição	Tipo Original	Tipo de Conversão
1	ID do Aparelho	Decimal	ASCII
2	ID do Aparelho	Decimal	ASCII
3	ID do Aparelho	Decimal	ASCII
4	ID do Aparelho	Decimal	ASCII
5	ID do Aparelho	Decimal	ASCII
6	ID do Aparelho	Decimal	ASCII
7	ID da função	Decimal	ASCII
8	ID da função	Decimal	ASCII
9	ID da função	Decimal	ASCII
10	ID da função	Decimal	ASCII
11	Comando	Decimal	ASCII
12	Comando	Decimal	ASCII
13	Comando	Decimal	ASCII
14	Comando	Decimal	ASCII
15	Valor da função	Hexadecimal	Decimal
16	Data: hora	Hexadecimal	Decimal
17	Data: minuto	Hexadecimal	Decimal
18	Data: dia	Hexadecimal	Decimal
19	Data: mês	Hexadecimal	Decimal
20	Data: ano	Hexadecimal	Decimal

Descrição dos campos:

1) Id do Aparelho - Identificação do aparelho que será acionado ou manipulado. Exemplo “tvsala”.
2) Id da Função - Identificação da função que deve ser executada, exemplo: “volum” para volume.
3) Comando - Qual o tipo de função que deve ser executada, por exemplo, se a função for “volum” o tipo poderá receber “aume” para aumentar, ou “dimi” para diminuir.
4) Valor da Função - Valor da função a ser executada. Exemplo se o valor passado for 10 com a função “volum” do tipo “aume”, será executado 10 vezes a função aumentar o volume.
5) Data - Data e hora do envio da mensagem.

4.2.4.2 - Mensagens do servidor

Abaixo pacote da mensagem confirmação ou erro.

TABELA 3 - Mensagem de resposta padrão do servidor

Descrição	Nº de bytes
ACK/NACK	4
Data	5
F	1

Caso a mensagem tenha sido recebida com sucesso um ACK é enviado, caso contrário um NACK é enviado.

Abaixo é descrito cada byte, como é transmitido e como deve ser feita a conversão para manipulação dos dados.

TABELA 4 - Descrição de cada Byte e tipo de conversão

Byte	Descrição	Tipo Original	Tipo de Conversão
1	N/ A	Decimal	ASCII
2	A/C	Decimal	ASCII
3	C/K	Decimal	ASCII
4	K/(Espaço)	Decimal	ASCII
5	Data: hora	Hexadecimal	Decimal
6	Data: minuto	Hexadecimal	Decimal
7	Data: dia	Hexadecimal	Decimal
8	Data: mês	Hexadecimal	Decimal
9	Data: ano	Hexadecimal	Decimal
10	F	Decimal	ASCII

Abaixo pacote da mensagem que confirma a execução.

TABELA 5 - Mensagem de resposta do servidor

Descrição	Nº de bytes
AKC/NACK	4
ID do aparelho	6
Data	5
F	1

Abaixo é descrito cada byte, como é transmitido e como deve ser feita a conversão para manipulação dos dados.

TABELA 6 - Descrição de cada byte e tipo de conversão

Byte	Descrição	Tipo Original	Tipo de Conversão
1	N/ A	Decimal	ASCII
2	A/C	Decimal	ASCII
3	C/K	Decimal	ASCII
4	K/(Espaço)	Decimal	ASCII
5	ID do Aparelho	Decimal	ASCII
6	ID do Aparelho	Decimal	ASCII
7	ID do Aparelho	Decimal	ASCII
8	ID do Aparelho	Decimal	ASCII
9	ID do Aparelho	Decimal	ASCII
10	ID do Aparelho	Decimal	ASCII
11	Data: hora	Hexadecimal	Decimal
12	Data: minuto	Hexadecimal	Decimal
13	Data: dia	Hexadecimal	Decimal
14	Data: mês	Hexadecimal	Decimal
12	Data: ano	Hexadecimal	Decimal
15	F	Decimal	ASCII

Descrição dos campos:

1) Id do Aparelho - Identificação do aparelho que foi acionado ou manipulado.
2) ACK/NACK - A seqüência ACK indica que a mensagem foi reconhecida e o comando vai ser executado. A seqüência NACK indica que um erro ocorreu e a mensagem não pode ser interpretada.
3) F - Indica o final do protocolo.
4) Data - Data e hora do envio da mensagem.

CAPÍTULO 5 - DESENVOLVIMENTO E IMPLEMENTAÇÃO

As práticas e métodos utilizados para o desenvolvimento dos softwares e hardwares serão apresentados nesse capítulo.

5.1 - Hardware

Foram desenvolvidos um módulo de controle infravermelho e um módulo para controle de motor, ambos controlados pelo Kit Didático Microcontrolador 8031. A seguir o desenvolvimento desses módulos é detalhado.

5.1.1 - Kit Didático Microcontrolador 8031

O desenvolvimento do firmware para o kit foi realizado com auxílio de um emulador de EPROM desenvolvido pela UP. Este emulador permite que o código em hexadecimal seja executado no kit sem necessidade de que uma EPROM seja gravada, pois o código é carregando através de um computador conectado ao emulador. A Figura 5.1 mostra o emulador de EPROM conectado ao kit do microcontrolador 8031.

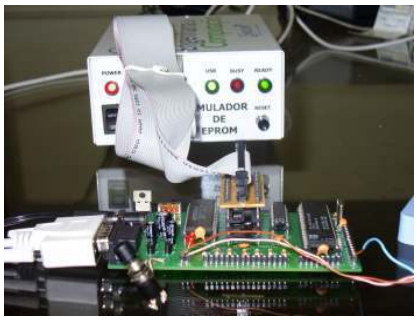

Figura 5.1 – Emulador de EPROM conectado ao Kit Microcontrolador 8031

A conexão entre kit microcontrolador 8031 e o computador foi realizada com a utilização de um conversor USB - Serial padrão, da marca Clone, mostrado na Figura 5.2. A comunicação foi feita sem controle de fluxo via hardware. Para isso o adaptador da Figura 5.3 foi montado. Existem diversas maneiras de se construir um adaptador serial utilizando-se dois conectores DB-9. A Figura 5.4 apresenta a maneira *Null Modem*, que foi utilizada para construir o adaptador deste projeto. Esse adaptador femea-femea pode ser usado em

qualquer aplicação que seja necessário conectar dois DTE (*Data Terminal Equipment*). Um adaptador equivalente, mas macho-macho seria utilizado para conectar dois DCEs (*Data Communications Equipment*). Essa configuração é usada em comunicações assíncronas para RS-232. Para utilizar comunicação síncrona, o adaptador deverá possuir conexões adicionais para o sincronismo (PUCRS,2008).

Figura 5.2 – Conversor USB - Serial

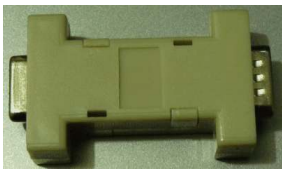

Figura 5.3 – Adaptador DB9

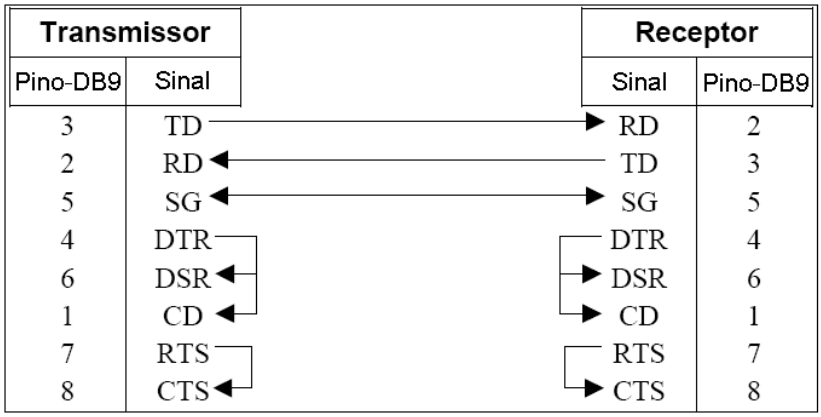

Figura 5.4 – Esquemático do Adaptador DB9

FONTE: adaptada de (UNESP,1999).

5.1.2 - Controle Infravermelho

Para testar o sistema infravermelho um circuito simples foi montado em protoboard. A Figura 5.5 mostra o esquemático do circuito de teste. Composto de um resistor – apresentado na Figura 5.5 como Rb – que controla a corrente gerada pelo sinal recebido do microcontrolador; um LED infravermelho; um transistor que é usado para acionar e desligar o LED; e segundo resistor – apresentado na Figura 5.5 como Rc – para controlar a corrente sobre o LED e o transistor.

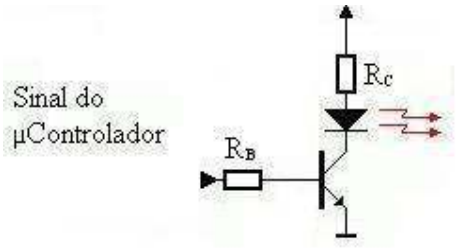

Figura 5.5 – Circuito de Testes do Infravermelho

Para a montagem é usado um transistor BC548B e um LED infravermelho TIL32, alimentado por com 2 baterias de 1.5V. Como no TIL32 a tensão máxima é de 1.6V e a tensão base-emissor para o BC548B é de 0,7V, tem-se a tensão V_E de 0,7V no resistor R_C (Eq.(1)). Aplicando a Lei de Ohm (Eq.(2)) tem-se que R_C é 35 Ω.

$$V_C = V_{total} - V_{LED} + V_{BE} \tag{1}$$

$$R = \frac{V}{I} \tag{2}$$

Para o resistor de base R_B tem-se que primeiro calcular a corrente de base, sabendo que o ganho h_{FE} típico do BC548B é 180 e a corrente no coletor é 20mA pela Eq.(3) chega-se a uma corrente de base de 0,111µA. Assim sendo, aplicando a Lei de Ohm com base na tensão mínima do pino P1.0 do microcontrolador, tem-se que R_B é 21.621Ω ou aproximadamente 22kΩ.

$$I_b = \left(\frac{I_c}{h_{FE}} \right) \tag{3}$$

Este sistema não pode ser mantido, pois ele não atendia as necessidades do protocolo infravermelho adotado. Com esse sistema todo o protocolo teria que ser implementado via software. Para resolver este problema um novo circuito foi desenvolvido.

O novo sistema é composto de oscilador NE555 que gera o um sinal de 38kHz com um *duty-cycle* de 38kHz, gerando o *duty carrier* necessário para o protocolo EXTENDED-NEC. Na Figura 5.6 é exibido o circuito responsável por acionar o LED infravermelho. O sinal resultante sobre o LED D2 é resultado da operação lógica 'E', realizada no circuito integrado 7408, entre o sinal de saída do pino P1.0 do microcontrolador e a saída do pino 3 do oscilador NE555.

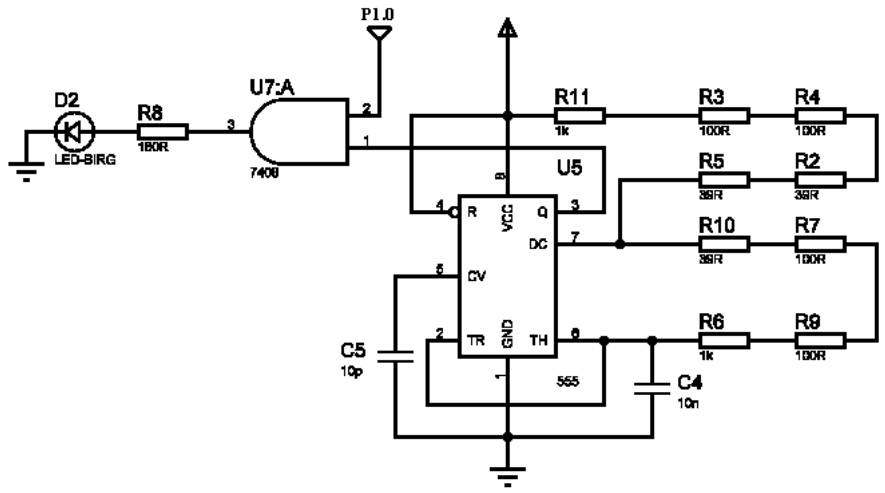

Figura 5.6 – Circuito do Controle Infravermelho

Com essa configuração para gerar o sinal infravermelho com o padrão EXTEND-NEC basta termos nível lógico alto na saída do pino P1.0 do microcontrolador, pois a saída do CI NE555 gera o *duty-cycle* necessário com freqüência de 38kHz.

O layout da placa de circuito impresso é exibido na Figura 5.7, este layout foi desenvolvido com auxilio do software OrCAD (ORCAD, 2008), com base no esquemático da Figura 5.6.

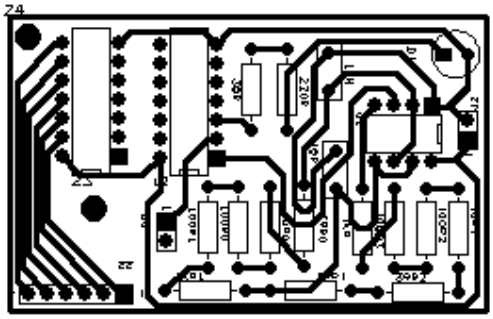

Figura 5.7 – Layout da placa do controle infravermelho

5.1.3 - Controle do Motor

Esta interface permite que através de um pino de saída P1.1 do microcontrolador seja acionado um motor de corrente alternada. Para que isto ocorra é realizado um isolamento óptico com um MOC3010 entre o circuito 127V do motor e 5V do microcontrolador, ou seja, não há contato físico no acionamento, o que garante a integridade do sistemas de baixa tensão.

Apesar de realizar o isolamento do circuito, o MOC3010 não possui uma dissipação de calor eficiente e não é indicado para acionamento direto de cargas indutivas, sendo assim um opto-isolador nesse circuito é usado na função de *trigger* pra um TRIAC BTA08-600V que esta colocado em série com o motor. Com isso, quando o MOC3010 está em estado aberto o TRIAC em série com o motor também se abre, desligando o motor, e quando o MOC3010 encontra-se fechado o TRIAC também se fecha, acionando o motor.

Um problema encontrado foi o momento de desligamento do motor. No momento em que ocorre a abertura do TRIAC, a tensão passa por zero e a tensão de alimentação é reaplicada em toda a estrutura. Em alguns momentos o TRIAC não tem a capacidade de bloquear esta tensão reaplicada e então liga espontaneamente. Com cargas indutivas a razão de subida desta tensão em relação ao tempo é maior e conseqüentemente mais impactante. Para resolução deste problema foi utilizado um circuito RC em paralelo com o TRIAC, conhecido como *snubber* (STMICROELETRONICS,2008).

O *snubber* faz a função de amortecedor, mantendo a freqüência, mas reduzindo a amplitude da oscilação. Calculando o circuito RC chega-se a uma tensão amortecida

suportada pelo TRIAC, fazendo com que o desligamento do motor ocorra sem problemas (HAGTECH, 2008).

A Figura 5.8 mostra o esquemático do circuito montado. O resistor R1 foi calculo com na corrente de saída do microcontrolador e resistência máxima do LED interno do opta-acoplador MOC3010. Os resistores R1, R3, R4 e o capacitor C1 foram calculados levando em conta a taxa de crescimento crítico da tensão de comutação e a corrente aplica sobre o motor.

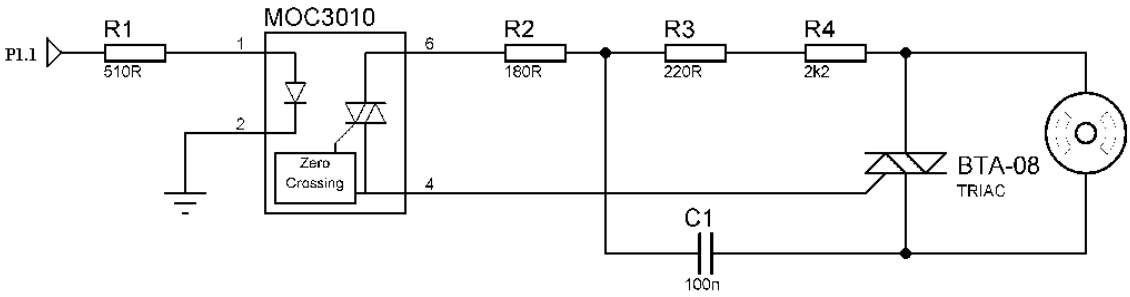

Figura 5.8 – Circuito de controle do motor

Com base no circuito descrito acima, foi montada um placa de circuito impresso com auxílio do software OrCAD. O layout dessa placa é exibido na Figura 5.9. As trilhas elaboradas com maior espessura são para garantir a dissipação do efeito Joule quando ocorre a passagem da corrente do motor que pode chegar a 2A.

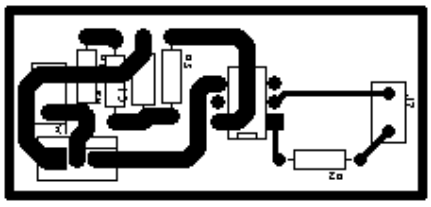

Figura 5.9 – Layout da placa de controle de motor

5.2 -Software

Algumas análises preliminares foram decisivas para fixar um método de programação consistente, principalmente para o software do *smartfone* que possui limitações como

memória e conexão. A seguir são descritos com mais detalhes algumas técnicas e métodos de programação adotados durante o desenvolvimento.

5.2.1 - Software do Smartfone

Este software é desenvolvido em Java 2 ME utilizando a ferramenta Netbeans (NETBENS, 2008) a uma comunicação via *socket*.

A comunicação por *socket* permite um maior controle sobre o aplicativo e com a utilização de um protocolo do tipo TCP garante a integridade da mensagem. O trecho de código da Figura 5.10 mostra como é criado este *socket* e feita esta conexão com o servidor localmente.

```
String enderecoServidor = "socket://localhost:12010";
SocketConnection  socket = (SocketConnection) Connector.open(enderecoServidor);
```

Figura 5.10 – Código de conexão com o servidor

Em J2ME as conexões são realizadas através de um conector generalizado, sendo assim é necessário especificar, na string de conexão, que se está utilizando *socket*. A string de conexão também tem a função de definir o endereço e porta que será feita a comunicação com o servidor.

Após a conexão estabelecida, é possível enviar e receber dados. O envio de dados é feito como mostrado no código da Figura 5.11. A variável *output*, que é do tipo *OutputStream*, permite acessar a fila de envio do *socket*; e a variável *stream* é onde fica armazenado no os dados que serão enviados através da execução do método de escrita *write* na fila de envio. O método flush força o sistema a enviar todos os dados presentes na fila de envio do *socket*.

```
byte[] stream =  new byte[20];
stream = "NACKOOOOOOOOOOOOOOOO".getBytes();
OutputStream output = socket.openOutputStream();
output.write(stream);
output.flush();
```

Figura 5.11 – Código para envio de dados em um *socket*

O recebimento de dados é feito como mostrado no código da Figura 5.12. A variável *input*, que é do tipo *InputStream*, permite acessar a fila de recebimento do *socket*; e a variável *recebe* é onde ficam armazenados os dados recebidos quando o método de leitura *read* é executado na fila de recebimento do *socket*.

```
byte[] recebe = new byte[10];
InputStream input = socket.openInputStream();
input.read(recebe);
```

Figura 5.12 – Código de recebimento de dados em um *socket*

Os dados enviados e recebidos seguem um protocolo desenvolvido para esta aplicação, este protocolo pode ser visto na

Para obter uma interface gráfica mais rica em detalhes optou-se por desenvolver uma das telas utilizando a API Canvas pra J2ME. Por se tratar de uma API de baixo-nível ela permite desenhar diretamente sobre o display, além de dispor de métodos para captura de eventos, como por exemplo o pressionamento de um botão (SUN MICROSYSTEMS, 2008c) .

A codificação utilizando Canvas é complicada, mas a liberdade de desenho compensa sua utilização, a Figura 5.13 exemplifica um método desenvolvido para desenho de um dos botões da Figura 5.14. O hoje em dia já existem alguns *Frameworks* que facilitam o desenvolvimento para este tipo de programação, como por exemplo J2ME Polish (J2ME POLISH, 2008).

```
protected void drawButtonRect(int press){

    graphics.setColor(0x000000);
    graphics.drawRoundRect(posWidth, posHeight,sizeWidthBtn , sizeHeigthBtn,10,10);

    if(pressed != press){
        graphics.setColor(0x888888);
        graphics.fillRoundRect(posWidth+2, posHeight+2,sizeWidthBtn-2 , sizeHeigthBtn-2,10,10);
    }
    else{
        graphics.setColor(0x909090);
        graphics.fillRoundRect(posWidth+1, posHeight+1,sizeWidthBtn-2 , sizeHeigthBtn-2,10,10);
    }
}
```

Figura 5.13 – Código de criação de um botão com uso de Canvas

O código da Figura 5.13 exibe o método *drawButtonRect.* Este método recebe como parâmetro um inteiro representante de uma tecla pressionada, e desenha uma imagem diferente para o botão correspondente. A variável graphics é um objeto do tipo *Graphics* que nos permite manipular o display como uma matriz de pixels. Através dos métodos disponibilizados por este objeto, pode-se alterar a cor dos pixels individualmente ou em grupos, como demonstrado no método *fillRoundRect* utilizado no exemplo da Figura 5.13.

A Figura 5.14 exibe o controle remoto desenhado com a utilização da API Canvas. Sempre que um botão é pressionado a função que o botão representa é descrita na caixa de texto no topo da imagem. Quando a tecla de 'Enviar' é pressionada esse comando é enviado ao servidor e a mensagem da caixa de texto alterada para o status da execução.

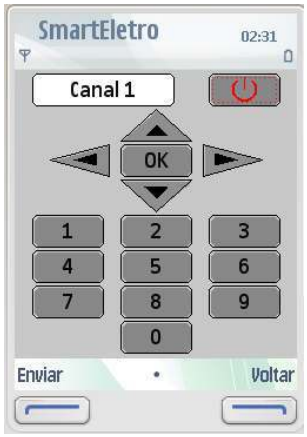

Figura 5.14 – Display desenvolvido com uso de Canvas.

5.2.2 - Software do Servidor

Este software é desenvolvido utilizando Java Standard Edition e Java Communication com a utilização da ferramenta Netbeans.

O servidor realiza a comunicação entre o *smartfone* e o centro de controle. Para que isso ocorra, um servidor *socket* aguarda a conexão do *smartfone* e outra conexão utilizando a API javax.comm faz a comunicação com o firmware do centro de controle.

A utilização da API javax.comm é possível porque a porta USB esta sendo tratada como uma porta serial padrão. Essa API descreve uma interface de baixo nível para comunicação com a porta serial (SUN MICROSYSTEMS, 2008c).

A Figura 5.15 exibe o código de conexão com a porta serial. A variável *portaId* é um objeto do tipo *CommPortIdentifier* e representa uma porta de comunicação disponível no sistema. A variável *serialPort* é um objeto do tipo *SerialPort* e recebe uma porta de comunicação que será utilizada para a comunicação serial.

```
CommPortIdentifier portaId =  CommPortIdentifier.getPortIdentifier("COM9");
SerialPort serialPort = (SerialPort)portaId.open("Serial", 30000);
```

Figura 5.15 – Conexão com a porta Serial.

Para que a comunicação serial ocorra alguns parâmetros devem especificados, no servidor e no centro de controle. Estes parâmetros são:

Baudrate – Taxa de transmissão utilizada na comunicação. Se o *baudrate* passado pela aplicação não for suportado, será gerada uma exceção JavaComm.

DataBits – Tamanho em bits do dado transmitido. Pode receber os valores: 5, 6, 7 e 8.

StopBits – Número de bits de parada. Pode receber os valores: 1, 1.5, ou 2.

Parity – Define o tipo de paridade usada na transmissão. Pode receber os valores: PARITY_NONE, PARITY_ODD, PARITY_EVEN, PARITY_MARK, PARITY_SPACE.

A Figura 5.16 exibe como é feito a configuração desses parâmetros para a variável *serialPort* utilizando o método *setSerialPortParams*. Além disso, demonstra a configuração do controle de fluxo para FLOWCONTROL_NONE utilizando o método *setFlowControlMode*, ou seja, não haverá controle de fluxo para as comunicações ocorrida na porta apontada pelo objeto *serialPort*.

```
int baudRate = 2400;
serialPort.setSerialPortParams(baudRate,
        SerialPort.DATABITS_8,
        SerialPort.STOPBITS_1,
        SerialPort.PARITY_NONE);

 serialPort.setFlowControlMode(SerialPort.FLOWCONTROL_NONE);
```

Figura 5.16 – Especificação dos parâmetros da conexão serial no servidor

A Figura 5.17 exibe o código resumido de envio e recebimento de dados pela porta serial. Observa-se que da mesma forma que ocorre na comunicação *socket*, a comunicação serial também possui filas (*streams*) de envio e recebimentos, representadas na Figura 5.17 pelas variáveis *output* e *input*, respectivamente.

```
OutputStream output = serialPort.getOutputStream();
output.write("tvsala".getBytes());
output.flush();
InputStream input = serialPort.getInputStream();
int novoDado = input.read();
```

Figura 5.17 – Envio e recebimentos de dados pela porta serial.

Para que o *smartfone* comunique-se um *serversocket* deve estar escutando a porta que ele tentará conectar. Quando uma nova comunicação é realizada um novo *socket* é criado, é através desse *socket* que a transferência de dados é realizada. A Figura 5.18 mostra a declaração de um objeto do tipo *ServerSocket*, chamado *servidor*, apontado para a porta de comunicação 12010. Quando o método *accept* é chamado, o objeto *ServerSocket* aguarda uma conexão de um cliente *socket* e em seguida direciona essa conexão para o objeto representado pela variável *novoSocket*.

```
ServerSocket servidor = new ServerSocket(12010);
Socket novoSocket = servidor.accept();
```

Figura 5.18 – Criação e conexão do *serversocket*

A Figura 5.19 exibe como ocorre o envio e recebimento de dados usando o *socket* criado pelo S*erverSocket*. A objeto do tipo *InputStream*, representado pela variável *input*, permite acessar a fila de recebimento do cliente *socket*; e a variável *novoDado* armazena o dado

recebido quando o método de leitura *read* é executado na fila de recebimento do *socket*. O objeto do tipo *OutputStream*, representado pela variável *output*, permite acessar a fila de envio do *socket*; quando o método de escrita *write* é executado a *string* "ACK" é adiciona como uma seqüencia de bytes na fila de envio. O método flush força o sistema a enviar todos os dados presentes na fila de envio do *socket*.

```
InputStream input = novoSocket.getInputStream();
int novoDado = input.read();

OutputStream output = novoSocket.getOutputStream();
output.write("ACK".getBytes());
output.flush();
```

Figura 5.19 – Envio e recebimento de dados no *socket* do servidor.

A Figura 5.20 exibe a tela do servidor. A primeira linha exibida é a mensagem de conexão com portal serial; a segunda linha mostra a mensagem que confirma a conexão do *ServerSocket*; a terceira linha exibe a chegada de uma nova conexão de celular, o IP(Internet Protocol) que o celular está conectado e a porta em que a comunicação ocorrerá no servidor; a quarta linha mostra o comando recebido no *socket*; a quinta linha exibe qual o comando foi devolvido ao celular. A sexta linha mostra o comando enviado para o centro de controle usando a porta serial e a sétima linha mostra o comando respondido na serial.

Figura 5.20 – Tela do servidor.

5.2.3 - Firmware

O firmware foi desenvolvido em Assembly utilizando a ferramenta Reads51 (RIGELCORP, 2008) e é executado no microcontrolador 8031.

Foi desenvolvido um código simples para comunicação serial. A seguir, na Figura 5.21, uma parte desse código é mostrada. O primeiro passo que pode ser visto nas quatro primeiras linhas da Figura 5.21 é a configuração e habilitação do Timer 1 para rodar com uma taxa de comunicação de 2400 bps, é este timer que possibilita a comunicação serial assíncrona. As linhas cinco e seis são necessárias para habilitar a comunicação serial. Já da linha sete até a linha nove ocorre à recepção de um byte que é armazenado no acumulador B. E as três últimas linhas escrevem o caráter 'A' na saída serial.

```
MOV TMOD,#20h   ; TMOD = 00100000..Timer 1 no Modo 2, controle por software
MOV TH1,#244    ; valor 244 em TH1 e TL1 para gerar a Taxa deComunicação de
MOV TL1,#244    ; 2400 BPS comfc=11,0592MHze; K=1(default)
SETB TR1        ; Dispara Temporizador
MOV SCON, #40h  ; SCON = 01000000 ..modo 1 do Canal Serial

SETB REN        ; Habilita a Recepção
JNB RI,$        ; Espera terminar a recepção
MOV B, SBUF
CLR RI          ; Prepara para nova recepção

MOV SBUF, #'A'  ; Transmite o caractere ASCII A
JNB TI,$        ; Espera terminar a transmissão
CLR TI          ; Prepara para nova transmissão
```

Figura 5.21 – Conexão serial no firmware

O cálculo da carga dos timers TH1 e TL1 é dado pela Eq. (4).

$$"C\arg a\_do\_Timer"(em\_decimal) = 256 - \frac{\left(f_{clock} \times 2^{SMOD}\right)}{384 \times taxa\_de\_transferência} \qquad (4)$$

O SMOD é o primeiro bit do registrador PCON (*Power Control Register*) que dobra a relação de divisão de freqüência na serial (NICOLOSI, 2000).

A primeira tentativa de funcionamento do infravermelho foi implementada com PWM (Modulação da Largura de Pulso) via software, para isso é utilizado o Timer 0 do microcontrolador 8031 no Modo 0, como ilustra a Figura 5.22. A partir do momento que

TR0 é habilitado e o Timer 0 é iniciado pode-se alternar a saída da porta P1.0 com base no tempo de contagem do timer.

```
PWM_SETUP:
        MOV TMOD,#00H           ; Timer0 em Modo 0
        MOV R7, #001            ; Seta o controle de largura do puslo
        SETB EA                 ; Habilita a interrupção
        SETB ET0                ; Habilita o Timer 0 Interrupt
        SETB TR0                ; Inicia o Timer
        RET
        TIMER_0_INTERRUPT:
        JB F0, HIGH_DONE        ; Se o flag F0 é setado então nos apenas
                                ; finalizamos a seção do ciclo high
                                ; e pulamos  para HIGH_DONE
```

Figura 5.22 – Setup do PWM

A parte alta do pulso então é dada pelo código da Figura 5.23, o *flag* F0 em nível alto indica que a seção alta do pulso foi iniciada, então o pino P1.0 é alterado para nível lógico alto e mantido até que o timer que chegue a zero. O byte alto do timer é gravado com o valor do registrado R7, mas como o timer está habilitado este valor é decrementado a cada contagem que ocorre.

```
LOW_DONE:
        SETB F0                 ; Faz F0=1 para indicar o inicio da seção high
        SETB P1.0               ; Faz o pino P1.0 alto(1)
        MOV TH0, R7             ; Grava o byte alto do timer com R7
                                ; (pulsa o valor de controle de largura)
        CLR TF0                 ; Limpa o flag de interrupção do Timer 0
        RETI
```

Figura 5.23 – Código da parte alta do pulso modulado.

A parte baixa do pulso é mostrada pelo código da Figura 5.24, na qual pode ser visto o *flag* F0 alterado nível baixo indicando que a seção baixa do pulso foi iniciada, então o pino P1.0 é alterado para nível lógico baixo e mantido até que o timer que chegue a zero. O byte alto do timer é gravado com o valor do acumulador A que corresponde ao resto da subtração de FFH e o valor do registrador R7.

```
HIGH_DONE:
        CLR F0                  ; Faz F0=0 para indicar o inicio da seção low
        CLR P1.0                ; Faz o pino P1.0 baixo(0)
        MOV A, #0FFH            ; Move FFH (255) para o acumulador A
        CLR C                   ; Limpa C (o bit de carry) para ele nao afetar
                                ; a subtração
        SUBB A, R7              ; Subtrai R7 de A. A = 255 - R7.
        MOV TH0, A              ; então o valor gravado em TH0 + R7 = 255
        CLR TF0                 ; Limpa o flag de interrupção do Timer 0
        RETI                    ; Retorna para a interrupção que chamou
```

Figura 5.24 – Código da parte baixa do pulso modulado

Chamando a rotina PWM_SETUP tem-se uma saída modulada no P1.0. Com isso há como controlar a largura do pulso apenas modificando o valor do registrador R7.

Esta configuração funcionou como esperado, mas o controle do pulso era difícil com a freqüência de 38kHz exigida pelo protocolo e também causava deformação na onda de resposta da saída P1.0.

Com a adição do circuito com o oscilador NE555 foi possível modificar a modulação do firmware para um método mais fácil de manipular comandos e endereços. Como o circuito formado pelo NE555 garante o *duty-cycle* quando há a necessidade de um pulso na saída infravermelha basta enviar nível lógico alto na saída do pino P1.0.

Para facilitar a adição de diferentes comandos e endereços foram codificadas 3 rotinas, uma que envia o *header bit*, uma que envia nível lógico alto e uma que envia nível lógico baixo, segundo o protocolo EXTENDED-NEC. A Figura 5.25 exibe o código da rotina que envia o *header bit*. Esta rotina está divida em duas sub-rotinas, onde a primeira sub-rotina corresponde à parte alta de 9ms do *header bit* e a segunda corresponde à parte baixa de 4ms. A rotina *ATRASO* atrasa a execução em 560 nanosegundos, no caso da primeira sub-rotina ela é chamada a quantidade de dezesseis, ou seja, o valor do registrador R0. Já na segunda sub-rotina a rotina *ATRASO* é chamada oito vezes.

```
PULSOINICIAL:               ;É chamado no inicio de cada comando
    MOV R0,#16              ;também conhecido como LEAD
    SETB P1.0
PULSOPARTE1:                ;Mantém 9ms em alto
    LCALL ATRASO;
    DJNZ R0, PULSOPARTE1
    CLR P1.0;
    MOV R0,#8
PULSOPARTE2:                ;Mantém 4,5ms em baixo
    LCALL ATRASO;
    DJNZ R0, PULSOPARTE2
    RET                     ;Fim da rotina PULSOINICIAL
```

Figura 5.25 – Rotina que envia o *header bit*

A Figura 5.26 exibe a rotina que envia o bit em nível lógico alto, segundo protocolo EXTENDED-NEC que determina que o pulso correspondente ao valor lógico alto deve ter 560 nanosegundos em alta seguido de 1,68 microssegundos em baixa.

```
BIT1:                       ;Executa o equivalente a nivel lógico 1
    SETB P1.0               ;segundo o protocolo NEC
    LCALL ATRASO;
    CLR P1.0
    LCALL ATRASO;
    LCALL ATRASO;
    LCALL ATRASO;
    RET                     ;Fim da rotina BIT1
```

Figura 5.26 – Rotina de envio nível lógico alto

A Figura 5.27 exibe a rotina que envia o bit em nível lógico baixo, segundo estabelecido pelo protocolo EXTENDED-NEC, que requer para o pulso de nível lógico baixo tenha 560 nanosegundos em alto e 560 nanosegundos em baixo.

```
BIT0:                       ;Executa o equivalente a nivel lógico 0
    SETB P1.0               ;segundo o protocolo NEC
    LCALL ATRASO;
    CLR P1.0
    LCALL ATRASO;
    RET                     ;Fim da rotina BIT0
```

Figura 5.27 – Rotina de envio de nível lógico baixo

A rotina ATRASO chamada dentro das rotinas anteriores gasta 516 ciclos de máquina para realizar um atraso de 560ns. A estrutura dessa rotina é mostrada na Figura 5.28, onde 510

ciclos são gasto na execução do comando DJNZ e mais 6 no *loop*. Considerando que cada ciclo de máquina tem 1.08x10-9 segundos, tem-se o atraso total esperado.

```
ATRASO: ;espera 560ns
    MOV R1,#FFH
    DJNZ R1,$
    RET
```

Figura 5.28 – Rotina de atraso.

Para o televisor utilizado foi utilizado o endereço 04FBH ou em binário 0000010011111011b. Como o endereço é usado em todos os comandos disparados foi criada uma rotina para enviar este endereço. A Figura 5.29 mostra a estrutura dessa rotina, que chama as rotinas *BIT0* e *BIT1* para gerar a seqüência dos 16 bits do endereço.

```
ENDERECO:                   ;Endereco tv sala 0000 0100 1111 1011
                            ;                  0    4    F    B
                            ;com tamanho de 26,88ms
        LCALL BIT0          ;A0
        LCALL BIT0          ;A1
        LCALL BIT0          ;A2
        LCALL BIT0          ;A3
        LCALL BIT0          ;A4
        LCALL BIT1          ;A5
        LCALL BIT0          ;A6
        LCALL BIT0          ;A7

        LCALL BIT1          ;A8
        LCALL BIT1          ;A9
        LCALL BIT1          ;A10
        LCALL BIT1          ;A11
        LCALL BIT1          ;A12
        LCALL BIT0          ;A13
        LCALL BIT1          ;A14
        LCALL BIT1          ;A15
        RET
```

Figura 5.29 – Rotina de envio do endereço.

Com o uso dessas rotinas é possível enviar um comando de forma simples, a Figura 5.30 exemplifica o envio do comando que aumenta o volume do televisor.

```
COMANDO_AUMENTAR_VOLUME:
        LCALL PULSOINICIAL
        LCALL ENDERECO
        LCALL K22;
        LCALL BIT0 ;BIT FINAL
        RET
K22:                            ;K22 FORMADO POR 8 bits : 1010 1000
                                ;Aumentar Volume 12,88ms x2
        LCALL BIT1              ;D0
        LCALL BIT0              ;D1
        LCALL BIT1              ;D2
        LCALL BIT0              ;D3
        LCALL BIT1              ;D4
        LCALL BIT0              ;D5
        LCALL BIT0              ;D6
        LCALL BIT0              ;D7
                                ;e 8 bits inversos : 0101 0111
        LCALL BIT0              ;D8  : /D0
        LCALL BIT1              ;D9  : /D1
        LCALL BIT0              ;D10 : /D2
        LCALL BIT1              ;D11 : /D3
        LCALL BIT0              ;D12 : /D4
        LCALL BIT1              ;D13 : /D5
        LCALL BIT1              ;D14 : /D6
        LCALL BIT1              ;D15 : /D7
        RET
```

Figura 5.30 – Rotina de envio do comando “Aumentar Volume”

Todos os comandos seguem o mesmo padrão do comando da Figura 5.30. Primeiro há a chamada da rotina de envio do header bit, em seguida, a chamada da rotina de envio do endereço do dispositivo e, na seqüência, a composição do comando com as chamadas das rotinas *BIT0* e *BIT1*. No caso da seqüência A857H ou 101010001010111b da Figura 5.33, será interpretado pelo televisor como aumento de volume.

A lista completa dos comandos implementados e suas descrições podem ser vistas no Apêndice B.

CAPÍTULO 6 - VALIDAÇÃO E RESULTADOS

Os testes de validação foram realizados utilizando dois celulares da marca Nokia, um modelo N95 que possui sistema operacional Symbian S60 3rd Edition, e um modelo 2760 com sistema operacional Symbian Series 40 5th Edition LE. Para os testes de memória foram utilizados dois SDKs desenvolvidos pela Nokia, são eles S60 3rd Edition SDK for Symbian OS (NOKIA, 2008b) e S40 5RD Edition SDK for Symbian OS (NOKIA, 2008c).

O servidor de teste foi executado no sistema operacional Windows XP SP2, em um computador com processador Intel Core Duo 1.86GHz, com 1GB de memória e 80GB de *HardDisk*, com o centro de controle conectado via porta USB.

Em todos os testes havia 2 celulares – um modelo Nokia 95 e um modelo Nokia 2760 – e 3 emuladores conectados simultaneamente ao servidor.

Os testes de uso de memória no celular foram extraídos das informações fornecidas pelos emuladores.

Os tempos de respostas foram obtidos usando uma versão modificada do software do celular que calcula a diferença entre o tempo de envio de um comando e a obtenção de sua resposta.

Os seguintes resultados foram obtidos para o software nos celulares:

- O software no celular apresentou estabilidade e baixo uso de memória quando executado juntamente com o sistema operacional, próximo de 11,7MB de RAM para o sistema operacional Symbian S60 3rd Edition e 1,1MB para Symbian S40 5rd Edition.

- A conexão do celular com o servidor mostrou-se eficiente e rápida com um tempo médio de resposta de 1.729s. E mesmo utilizando tecnologia de conexão GPRS, que possui um baixo tráfego de dados (entre 56kbsp e 115kbs) o tempo médio foi de 2.738s. O gráfico da Figura 6.1 exibe as médias de tempo de resposta aos comandos para conexões GPRS, WI-FI e 3G, com 40 amostras de tempo utilizadas para cada conexão. Os tempos adquiridos nos testes podem ser vistos no Apêndice C. Como pode ser visto, a conexão WI-FI apresentou melhores resultados, isto está ligado as características da conexão que possui uma taxa de transferência em torno de 54Mbps e a comunicação é feita sem a necessidade de desvio por servidores externos. Uma grande diferença também pode ser

notada entre as tecnologias de comunicação sem fio 3G e GPRS. Isto se deve ao fato que a banda larga digital utilizada na comunicação 3G permite a transferência de uma quantidade maior de informações por segundo, quando comparada com a banda estreita e analógica utilizada na comunicação GPRS.

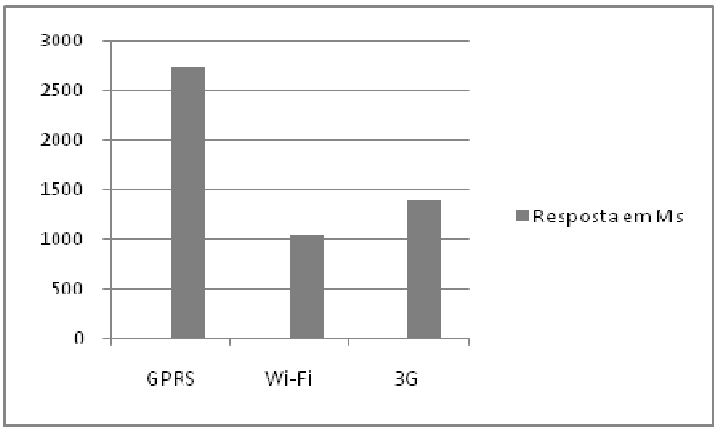

Figura 6.1 – Gráfico dos Tempos de Resposta dos Comandos.

- A interface gráfica não apresentou mudanças de cores e distorções significativas em relação ao display especificado para o display do celular 2760 que possui 128 x 160 pixels de resolução e profundidade de cores de 16bits.

- Em alguns lugares onde o sinal de dados do celular sofria interferência ou queda, o envio de comandos apresentou grandes atrasos, que por sua vez causavam queda na conexão, mas o sistema de reconexão automática garantia a integridade das informações reenviando estas informações logo após a conexão ser restabelecida.

Os seguintes resultados foram obtidos para o software do servidor:

- O servidor ficou estável mesmo com mais de um celular conectado. A falta de recursos impediu um teste de carga mais preciso, mas com dois aparelhos e três emuladores conectados nenhum problema foi detectado para até 200 comandos.

- A conexão entre servidor e interface de controle foi executada com êxito e não apresentou problemas nos testes realizados.

Os seguintes resultados foram obtidos para a interface de controle:

- A interface de controle infravermelho apresentou problemas relacionados ao alcance do infravermelho, a intensidade do LED utilizado mostrou-se muito baixa, com isso o sistema necessita agregar um repetidor para longas distâncias.

- Para distâncias de 10 centímetros a 1 metro o infravermelho apresentou bons resultados.

- O sistema infravermelho funcionou para ângulos de 85 a 105 graus, perpendicularmente, em relação ao aparelho de TV utilizado para os testes.

- A interface de controle de motor foi testada com dois motores de corrente continua ligados a uma tensão de 127V, o primeiro com 250W e o segundo com 180W de potência. Ambos não apresentaram problemas de funcionamento.

- O hardware desenvolvido não apresentou problemas, não houve sobreaquecimento dos componentes e o tempo de resposta foi satisfatório.

- O firmware desenvolvido em sua versão final não apresentou problemas durantes os testes, sendo testados com até 200 comandos aleatoriamente.

CAPÍTULO 7 - CONCLUSÃO

O acesso remoto a dispositivos eletro-eletrônicos ainda é pouco comum em nossa sociedade, este projeto se propôs a desenvolver um sistema que permitisse acionar os mais diversos eletro-eletrônicos remotamente com uso de um aparelho celular. Para provar o conceito foi escolhido um aparelho eletro-eletrônico como base de desenvolvimento, mas no decorrer do projeto mais um eletro-eletrônico foi incluído.

Ao final, o sistema cumpriu com os objetivos traçados no escopo do projeto, mas para tornar-se viável comercialmente são necessárias algumas modificações, principalmente em relação à interface de controle.

A utilização de infravermelho, mesmo apresentando resultados satisfatórios, precisa ser revista e talvez substituída por uma tecnologia compatível com um número maior de dispositivos.

O controle de motor que foi adicionado tardiamente como um complemento ao projeto inicialmente definido, demonstrou que o uso de um microcontrolador proporciona a inclusão de novos dispositivos eletro-eletrônicos sem a necessidade de uma reestruturação do *hardware*. Além de provar a flexibilidade dos softwares, que por estarem focados no paradigma de orientação objeto provêem um modo fácil de adição de novos códigos.

As tecnologias adotadas para o desenvolvimento dos softwares estão em plena evolução, isso possibilitou que a mesma tecnologia seja utilizada em dispositivos antigos e novos, o que representa um atrativo a mais para a continuidade de pesquisas com uso dessas tecnologias.

No caso do software do celular, a portabilidade e compatibilidade com um grande número de dispositivos trás grandes vantagens, pois para que o mesmo software seja executado em diferentes sistemas, poucas modificações necessitam ser realizadas. Por ser um mercado relativamente novo, ainda não há uma tecnologia predominante para este tipo de aplicação, J2ME está sendo adotado por um grande número de indústrias na área de dispositivos móveis e torna-se cada vez mais uma tendência a ser seguida.

O software do servidor também tem como vantagem a portabilidade da tecnologia Java, pois é possível executá-lo em mais de um sistema operacional, sem a necessidade de grandes modificações.

Por abranger um grande número de áreas de pesquisa o projeto dispõe de inúmeras possibilidades de extensão. A princípio um sistema de persistência de dados no servidor e no celular parece ser a maior necessidade, pois uma das situações encontrada durante os testes foi a de durante a utilização do sistema a bateria do aparelho celular ou do computador acabar e o sistema ter de ser reiniciado. Sem um sistema de persistência de dados todos os comandos devem ser reenviados para que a operação seja concluída e as operações em andamento são perdidas.

Uma segunda possibilidade de extensão seria disponibilizar o mesmo sistema de controle presente no *smartfone* via interface Web. Como o protocolo de comunicação é simples e utiliza conexão *Socket* seria viável desenvolver este mesmo sistema de controle em plataforma Web.

Em síntese, com a evolução dos sistemas de automação residencial, projetos como esse são cada vez mais viáveis e atrativos, pois em pouco tempo permitirão a inclusão de funcionalidades como controle de iluminação, climatização, controle de portas, janelas, segurança, sistemas de vigilância, etc.

CAPÍTULO 8 - REFERÊNCIAS BIBLIOGRÁFICAS

TANENBAUM, **Rede de computadores**. Rio de Janeiro, Editora Campos, 2003.

GIMENEZ, Salvador P.. **Microcontroladores 8051: Teoria de hardware e software**, aplicações em controle digital, laboratórios/simulação, São Paulo, Pearson Education do Brasil, 2002.

NICOLOSI, Denis Emílio Campion. **Microcontrolador 8051 Detalhado**, 2ª Ed. São Paulo: Érica, 2000.

RIGGS, Roger. TAIVALSAARI, Antero. VANDENBRINK, Mark. **Programming Wireless Devices with the Java 2 Platform,Micro Edition**, Upper Saddle River, Addison-Wesley, 2001.

MORRISON, Michael. **Wireless Java with J2ME**, Sams,2001.

MORAES, ROGÉRIO. **Sistemas Embarcados Para Automação De Força de Vendas.** Trabalho de conclusão de curso. Sistemas de Informação, Faculdade Integradas do Brasil - Unibrasil, 2008.

DOHMS, RAFAEL M. I**mplementando uma solução de automação residencial com custo reduzido baseada em open source**. Trabalho de conclusão de curso. Centro Universitário de Brasília, 2005.

JU, DONG-YING; ZHONG, RUI; TAKAHASHI, MASAYA. **Development of Remote Control and Monitor System for autonomous Mobile Robot Based on Virtual Cell Phone**. Saitama, 2007

MAURO RICARDO, **Domótica. Trabalho de Conclusão de Curso**. Ciência da Computação, Centro Universitário Feevale - Feevale, 2007 .

SYMBIAN OS, **Insing 1: Smartfones and mass market**. Disponível em: <http://www.symbian.com/symbianos/insight/insight1.htm>. Acesso em Março/2008.

INTEL, **Backgrounder**. Disponível em: <http://www.intel.com/pressroom/enhanced/40th_Anniversary/40th_anniversary_backgrounder.pdf?iid=SEARCH>. Acesso em Novembro/2008.

WISC, **Light Emitting Diode**. Disponível em: <http://mrsec.wisc.edu/Edetc/cineplex/LED/index.html>. Acesso em Novembro/2008.

UFPA, **Classificação de Métodos de Medida ou Comparação de "Cor"**. Disponível em: <http://www.ufpa.br/ccen/quimica/classificacao%20de%20metodos.htm>. Acesso em Março/2008.

NOKIA**, Nokia N95**. Disponível em: <http://www.forum.nokia.com/devices/N95>. Acesso em Março/2008.

NOKIA**, S60 Platform SDKs for Symbian OS, for Java.** Disponível em: <http://www.forum.nokia.com/info/sw.nokia.com/id/6e772b17-604b-4081-999c-31f1f0dc2dbb/S60_Platform_SDKs_for_Symbian_OS_for_Java.html> Acesso em Novembro/2008.

NOKIA**, Series 40 Platform SDKs.** Disponível em: <http://www.forum.nokia.com/info/sw.nokia.com/id/cc48f9a1-f5cf-447b-bdba-c4d41b3d05ce/Series_40_Platform_SDKs.html> Acesso em Novembro/2008.

UNIVERSIDADE POSITIVO, **Manual da Placa** 8031. Disponível em: <http://engcomp.up.edu.br/arquivos/engcomp/File/Placas%20DOCs/Manual%20Placa%208031.pdf>. Acesso em Março/2008.

NETBEANS, **Netbeans IDE 6.5**. Disponível em: <http://www.netbeans.org>. Acesso em Dezembro/2008.

KNOWLEDGE BASE, **NEC Protocol**. Disponível em: <http://www.sbprojects.com/knowledge/ir/nec.htm >. Acesso em Agosto/2008.

HOLTEK, **HT6221/HT6222 Multi-Purpose Encoders**. Disponível em: <http://www.holtek.com/English/docum/consumer/6221_2.htm >. Acesso em Agosto/2008.

SUN MICROSYSTEMS, **J2ME at a Glance**. Disponível em:<http://java.sun.com/javame/index.jsp>. Acesso em Agosto/2008.

SUN MICROSYSTEMS, **J2ME Technology**. Disponível em: <http://java.sun.com/javame/technology/index.jsp>. Acesso em Novembro/2008.

SUN MICROSYSTEMS, **J2ME Technology APIs & Docs**. Disponível em:<http://java.sun.com/javame/reference/apis.jsp>. Acesso em Setembro/2008.

SUN MICROSYSTEMS, **Class SerialPort**. Disponível em : <http://java.sun.com/products/javacomm/reference/api/javax/comm/SerialPort.html>. Acesso em Maio/2008.

J2ME POLISH, **J2ME Polish**. Disponível <www.j2mepolish.org>. . Acesso em Setembro/2008.

UNIFEI, **Aprendendo Regras Por Observação Em Uma Casa Inteligente**. Disponível em: <http://www.fei.edu.br/~flaviot/pub_arquivos/cba2006.pdf>. Acesso em Outubro/2008

UFLA**, Uso da Tecnologia Móvel no Auxílio à Reeducação Alimentar.** Disponível em: <www.dcc.ufla.br/infocomp/artigos/v3.1/art04.pdf>. Acesso em Outubro/2008.

IEEE COMPUTER SOCIETY, **Smart Office: Design of an Intelligent Environment**. Disponível em: <http://www2.computer.org/portal/web/csdl/doi/10.1109/5254.941359>. Acesso em Outubro/2008.

ORCAD, **Cadence OrCAD Solution**. Disponível em: <http://www.cadence.com/products/orcad/pages/default.aspx>. Acesso em Outubro/2008.

FARCHILD SEMICONDUCTOR, **NE/SA/SE555/SE555C Timer.** Disponível em: <http://www.fairchildsemi.com/ds/LM/LM555.pdf>. Acesso em Novembro/2008.

HAGTECH, **Calculating Optimun Snubber**. Disponível em: <http://www.hagtech.com/pdf/snubber.pdf >. Acesso em Outubro/2008.

STMICROELETRONICS, **RC Snubber Circuit Design for TRIACs**. Disponível em: <http://www.st.com/stonline/books/pdf/docs/6785.pdf>. Acesso em Outubro/2008.

CLONE, **Adaptador / Conversor USB / Serial**. Disponível em: <http://www.clone.com.br/db/detalhes_prod.asp?detalhe=05087>. Acesso em Abril/2008.

UNESP, **Sistemas Microcomputadorizados Laboratório III – Interação com a Porta Serial**. Disponível em: <http://dee.feg.unesp.br/Disciplinas/SEL3103/Labs/Lab3_Porta_Serial.pdf>. Acesso em Novembro/2008.

PUCRS, **DB9 Null Modem Cable.** Disponível em: <http://www.ee.pucrs.br/~dbarros/d2005_1/Microproc/T2_Serial/cabos.html>. Acesso em Novembro/2008.

RIGELCORP, **Reads51 .** Disponível em: < www.rigelcorp.com/reads51.htm>. Acesso em Novembro/2008.