CENTRO UNIVERSITÁRIO DE BRASÍLIA -UniCEUB

CURSO DE ENGENHARIA DE COMPUTAÇÃO

PAULO HENRIQUE MARTINS PIMENTEL

PROPOSTA DE AUTOMAÇÃO PREDIAL UTILIZANDO CELULAR COM TECNOLOGIA BLUETOOTH

Orientador: Prof. M.C. Maria Marony Sousa Farias

Brasília

dezembro, 2010

PAULO HENRIQUE MARTINS PIMENTEL

PROPOSTA DE AUTOMAÇÃO PREDIAL UTILIZANDO CELULAR COM TECNOLOGIA BLUETOOTH

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

Orientador: Prof. M.C. Maria Marony Sousa Farias

Brasília

dezembro, 2010

PAULO HENRIQUE MARTINS PIMENTEL

PROPOSTA DE AUTOMAÇÃO PREDIAL UTILIZANDO CELULAR COM TECNOLOGIA BLUETOOTH

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

Orientador: Prof. M.C. Maria Marony Sousa Farias

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

_____

Prof. Abiezer Amarilia Fernandez
Coordenador do Curso

Banca Examinadora:

_____

Prof. M.C. Maria Marony Sousa Farias, Mestre em Engenharia Elétrica.
Orientador

_____

Prof. Antonio Barbosa Júnior, Especialista em Engenharia de Software.
UniCEUB

_____

Prof. Gil Renato Ribeiro Gonçalves, Doutor em Física
UniCEUB

_____

Prof. Luis Cláudio Lopes Araújo, Mestre em Matemática Pura.
UniCEUB

DEDICATÓRIA

Primeiramente à minha família, em especial a minha mãe, Vera Lúcia Martins (In Memória), minha avó, Lourdes Alves Martins e meu primo, Lúcio Rafael Martins, pela paciência, carinho, compreensão e força que me deram durante o curso, na elaboração deste projeto final e principalmente durante toda minha vida.

Aos meus grandes amigos (as) Gabrielle Macedo, Jorge Alberto, Sabrine Ferretti, Ricardo Ferretti, Romulo Müller, Alfredo Casanova e Daniel Bastos, pela credibilidade e confiança que vocês depositaram em mim, pelo mútuo aprendizado de vida, durante toda nossa convivência, no campo profissional e particular. Amigos (as), gratidão eterna!

Ao Professor Francisco Javier De Obaldía, que sempre me estimulou com seus exemplos a me tornar um competente engenheiro.

AGRADECIMENTOS

À minha professora orientadora Maria Marony que esteve disponível sempre que precisei, me apoiando com conselhos e críticas construtivas.

Aos amigos Daniel Ataíde Leite Campos e José Carlos da Silva Santa Cruz por todas as críticas e ajudas que possibilitaram a excelência desse trabalho.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE SÍMBOLOS

API – Application Program Interface (Interface de Programação de Aplicativos)

CI – Circuito integrado

CPU – Central Processing Unit (Unidade Central de Processamento)

DCE – Data Communication Equipament (Equipamento de Comunicação de Dados)

DPTR – Data Pointer (Ponteiro de Dados)

DTE – Data Terminal Equipament (Equipamento Terminal de Dados)

GND – Ground (terra elétrico)

J2EE – Java 2 Enterprise Edition (Edição para Empresas)

J2ME – Java 2 Micro Edition (Edição Micro)

J2SE – Java 2 Standard Edition (Edição Padrão)

JVM – Java Virtual Machine (Máquina Virtual Java)

KVM – KyloByte Virtual Machine (Máquina Virtual KyloByte)

LED – Light Emitting Diode (Diodo Emissor de Luz)

PAN – Personal Area Network (Rede de Área Pessoal)

PDA – Personal Digital Assintant (Assistente Pessoal Digital)

RAM – Random Access Memory (Memória de Acesso Aleatório)

RESUMO

Neste projeto, é demonstrada uma aplicação de automação predial utilizando *Bluetooth*. Uma interface com o usuário é estabelecida fazendo com que seja selecionada a opção desejada por este. A opção é enviada a um computador que recebe o dado, e então o envia pela porta serial. Um microcontrolador recebe o dado enviado, e dependendo do dado recebido, executará a ação desejada pelo usuário. Esta ação pode ser: ligar ou desligar um aparelho; abrir ou fechar um portão eletrônico etc. Neste projeto, o dispositivo a ser acionado é representado por um protótipo juntamente com um conjunto de LEDs.

Palavras Chaves: *Bluetooth*, Java, Microcontrolador, J2ME, Automação predial.

ABSTRACT

Inside this work, is demonstrated a residential automation application using Bluetooth. An interface with the user is established making him to select the option desired. The option is sent to a computer that receives the data, and then sends it through the serial port. A microcontroller receives the data, and depending on the data received, it will execute the user desired action. This action can be: turn on or turn off a device, turn on or turn off a lamp etc. Inside this work, the device to be turned on or off is represented by a set of LEDs.

Keywords: Bluetooth, Java, Microcontroller, J2ME, Residential automation.

CAPÍTULO 1 - INTRODUÇÃO

A qualidade de vida é um dos requisitos mais procurados pela população. Essa qualidade de vida, muitas vezes, está diretamente ligada a algum desenvolvimento eletrônico. A proposta deste trabalho é aumentar o conforto das pessoas, através do desenvolvimento de uma aplicação que visa ações simples, como, abrir um portão eletrônico de um prédio e chamar o elevador mais próximo do subsolo a partir de um simples comando do celular. Assim quando um morador (seja de casa ou apartamento) estivesse chegando nas proximidades do seu portão de acesso à garagem, faz uma ligação para um número específico. Este número aciona o *software* que controla os comandos de abrir e fechar do portão, enviando um código que informa que o portão deve ser aberto. Caso o morador esteja chegando a um apartamento, o *software* localiza qual o elevador está mais próximo do subsolo, da entrada daquele morador e o direciona para aquele andar.

Para que estas ações possam ser executadas, neste projeto são desenvolvidos 3 *softwares*: um para celular, um para computador e outro para microcontrolador. O primeiro estabelece uma interface entre o usuário e os outros dois softwares. O segundo recebe o comando enviado pelo celular, por meio da tecnologia *Bluetooth* e transmite estes dados pela porta serial. O Terceiro recebe os dados da porta serial e executa a ação de acordo com o dado recebido.

Para visualização e validação do projeto, são utilizados um conjunto de LEDs, ligados ao microcontrolador, representando os andares do prédio no qual o elevador se encontra no momento.

1.1 Objetivo

O objetivo do projeto é utilizar a tecnologia que envolve desenvolvimento para dispositivos móveis, utilizando a linguagem Java, atualmente a mais acessível por ser um *software* livre, ou seja, que pode ser usado, copiado, estudado ou modificado sem restrições. A vantagem maior da linguagem de programação Java é o grande número de artefatos para estudos disponíveis na internet e por meio dos livros e artigos. Através de um mecanismo de

busca, muitas comunidades, artigos e inúmeras informações sobre a linguagem podem ser encontrados.

Com essa ferramenta, que é um Software Livre, é possível a criação de programa, levando em consideração a topologia ideal, para que os edifícios sejam cada vez mais modernos e inteligentes e que os equipamentos eletrônicos que nele sejam encontrados sejam manipulados mais facilmente e com maior conforto.

A disposição do trabalho em termos de organização de capítulos será a seguinte:

- Capítulo 1 – Uma introdução contendo informações sobre o trabalho, objetivo e motivação.
- Capítulo 2 – Apresentação dos conceitos teóricos para o andamento do projeto.
- Capítulo 3 – Modelo proposto juntamente com a topologia (fluxogramas, diagramas).
- Capítulo 4 – Aplicação da solução e apresentação de resultados. Aqui é mostrado o que foi feito de acordo com a proposta para a conclusão do projeto.
- Capítulo 5 – Conclusão sobre pontos fortes e pontos fracos, ressaltando as principais vantagens, limites do projeto e sugestões para projetos futuros.

1.2 Apresentação do Problema

Na medida em que os custos aumentam e a preocupação com o meio ambiente passa a fazer parte cada vez mais frequente de cada projeto, a construção de edifícios inteligentes se torna uma realidade no mercado nacional e mundial. A necessidade de economizar energia tem mostrado que a automação predial é a melhor saída.

Aplicações de automação predial, portanto, projetam um edifício moderno à frente daqueles que desconsideram exigências do mercado, como conforto, praticidade e segurança.

Até pouco tempo, muitos prédios tinham os serviços automatizados, mas não integrados, o que comprometia o potencial desempenho da automação. No entanto, isso está mudando. As novas construções já nascem com a ideia prévia de seus idealizadores de que têm que possuir facilidades para os usuários e que lhes sejam oferecidas soluções integradas.

De acordo com pesquisas realizadas pela Anatel (WWW.ANATEL.GOV.BR, 2010) uma grande quantidade de aparelhos de telefone celular já saem de fábrica com a tecnologia

de troca de dados utilizando o *Bluetooth.* Levando em consideração o grande número de celulares por pessoas, e os computadores com baixo custo, pode-se configurar um sistema para a automação predial (ex.: abertura do portão da garagem, abertura da portaria, etc.), com o envio do comando pelo celular utilizando a tecnologia sem fio *Bluetooth* para o computador, de forma descomplicada e barata. O computador por sua vez envia o dado recebido para um microcontrolador que faz um tratamento lógico da situação e executa as ações demandadas.

O objetivo é fazer com que o morador do prédio tenha controle do portão por meio do celular, juntamente com a comodidade do elevador já está no subsolo o aguardando.

CAPÍTULO 2 - REFERENCIAL TEÓRICO

A seguir será apresentado o referencial teórico utilizado neste projeto. Neste capítulo estão descritos todos os componentes utilizados no projeto. Para evidenciar, a figura 2.1 apresenta uma visão geral do projeto. Cada dispositivo (celular, computador, portão (protótipo) e microcontrolador) tem um *software* associado.

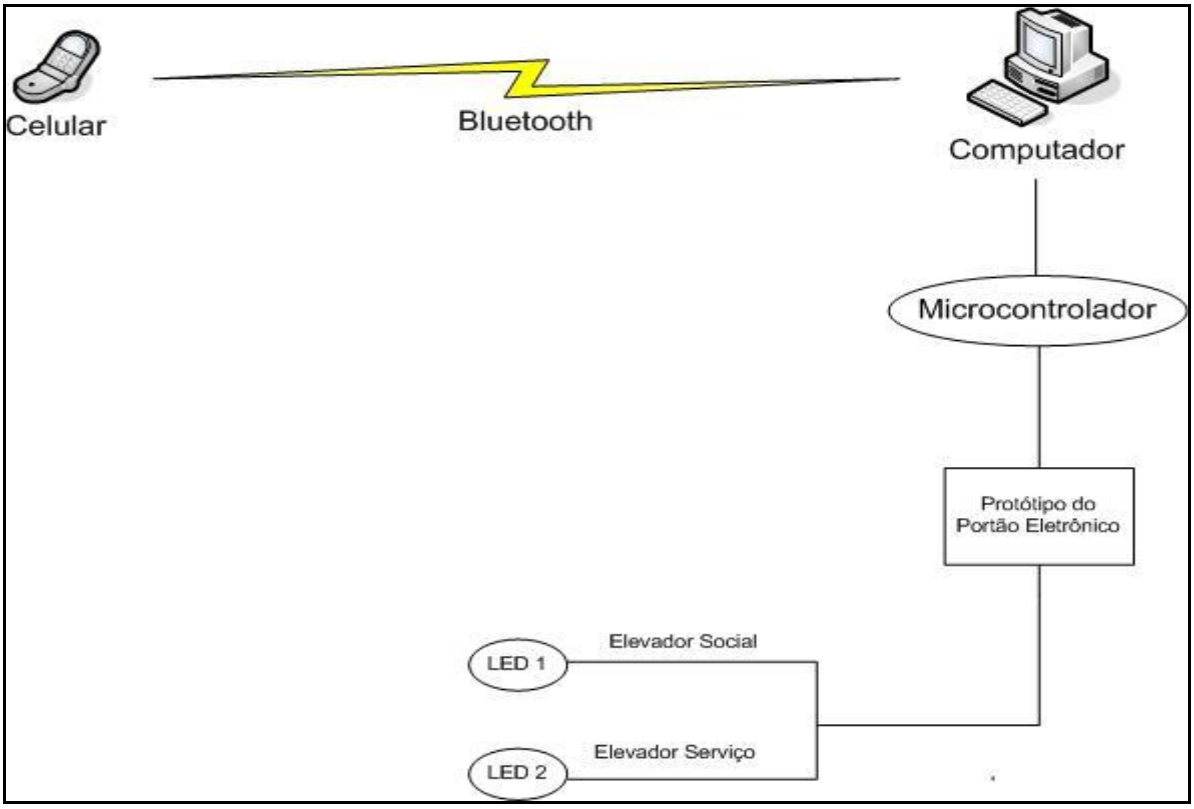

Figura 2.1 – Visão Geral do Projeto (Exemplo)

2.1 Kit Microcontrolador

É uma peça onde encontramos o microcontrolador com todos os aparelhos conectados as suas portas, conforme apresentado na figura 2.2. Percebe-se que o kit contém 8 LEDs, uma porta serial (para receber o cabo), um teclado e um display.

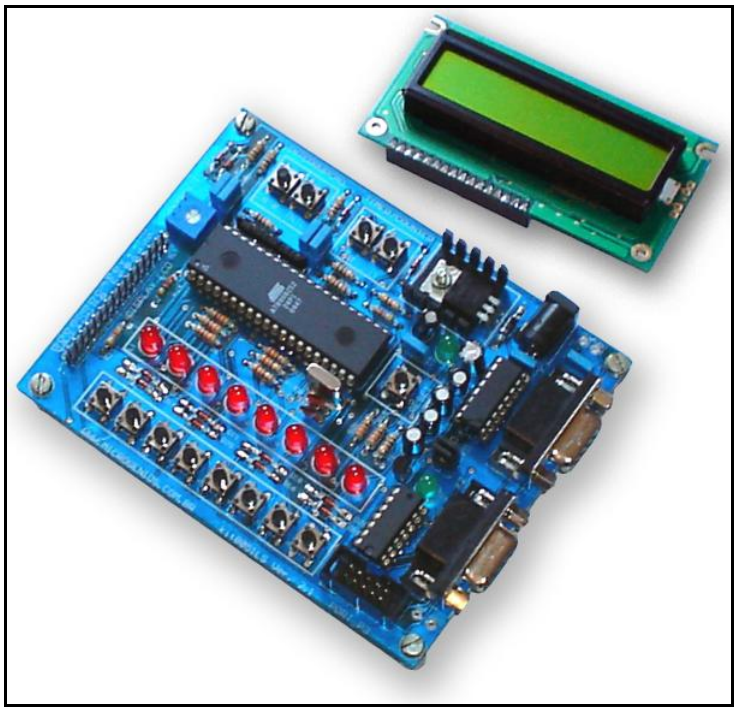

Figura 2.2 - Kit de microcontrolador (Exemplo)

2.1.1 Microcontrolador

Microcontroladores são dispositivos semicondutores em forma de Circuito Integrado. Não são utilizados em casos que requerem grandes quantidades de memória, pois são completamente limitados neste quesito. Alguns exemplos desse uso: Automação predial (tópico do projeto, exemplos portões eletrônicos, elevadores), Automação industrial (robótica), automação residencial (microondas, luz), (GIMENEZ, 2002).

2.1.2 Microcontrolador 8051

Em 1981 a Intel iniciou a produção do microcontrolador 8051, que possuía somente microcontrolador 8 bits e foi a mais bem sucedida venda na história. Em 1982 inicialmente foram produzidos 2 milhões de unidades, porém o mercado foi tão receptivo com o modelo 8051 que em 1993 foram produzidos 126 milhões, um crescimento bastante crescente em 11 anos.

Figura 2.3 – Microcontrolador Intel 8051

Fonte (http://eletronicadigital.co.cc/?p=8, 24/08/2010)

A tecnologia dos microcontroladores 8051 já existe aproximadamente há 30 anos. Alguns fatores podem ser citados para explicar o porquê desses componentes, somente agora, estarem sendo utilizados e divulgados amplamente.

Os principais são:

- Preço: O valor do componente teve uma queda de valor bem significativa na última década. Devido a isso, a fabricação e projetos que envolvem esse tipo de componente ficaram mais viáveis, barateando projetos e tornando-os mais compactos.
- Memória interna: O crescimento da linha 8051 se dá devido à utilização de memória interna, e muitas das vezes de memória flash, facilitando a programação, atualização e regravação.
- Ferramentas de desenvolvimento: Hoje existem inúmeras IDE's (Integrated Development Environment - ambiente integrado para desenvolvimento de software) que permitem programação em várias linguagens de alto nível e, algumas até de forma visual, sem mencionar os simuladores cada vez mais práticos e reais. Isso tornou muito mais agradável o trabalho de desenvolvimento com microcontroladores.

Esclarecimento sobre o porque deve-se conhecer a família do microcontrolador 8051 um pouco mais:

- Estrutura simples e de fácil entendimento. O bom conhecimento da estrutura do microcontrolador facilita o aprendizado e entendimento dos outros microcontroladores existente no mercado.

- Instruções do 8051 são bastante didáticas. Permite que a sua utilização seja mais fácil. Prova disso, é que esse microcontrolador é utilizado por dispositivos como controladores USB e SoC (*System-on-Chip*).
- Fazem parte da Família 8051. São componentes muito confiáveis para a grande maioria das aplicações.

2.1.3 Organização da memória

As memórias do microcontrolador são logicamente separadas conforme mostrado na figura 2.4. Com isso a memória de dados pode ser acessada por meio de endereços de 8 bits e também por endereçamento indireto de 16bits, gerados pelo registrador DPTR (Data Pointer). Isso faz com que a manipulação e armazenamento pela Central Unit Processing (CPU) de 8 bits sejam mais rápida. (CORRADI JUNIOR, 2006).

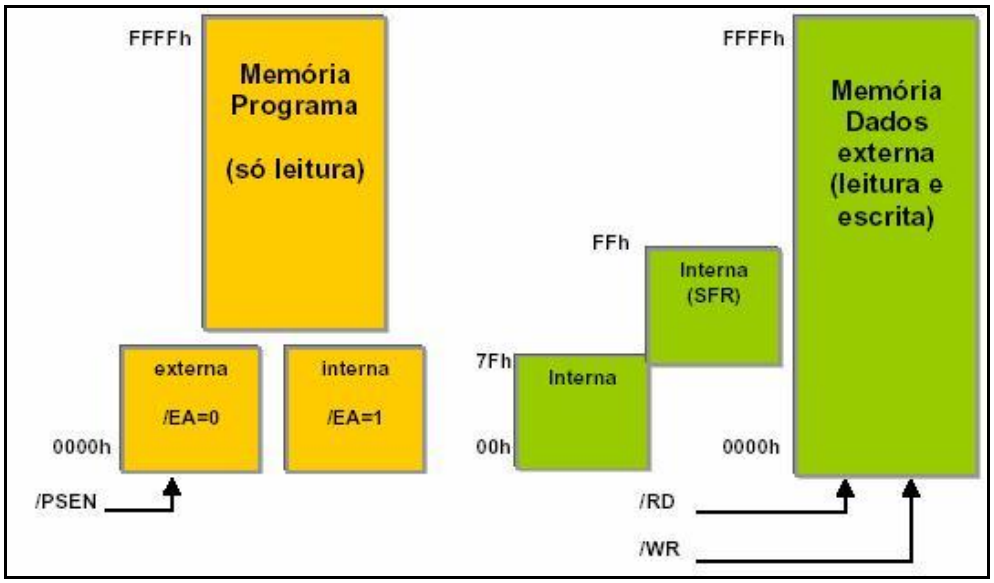

Figura 2.4 - Separação entre memórias do microcontrolador

2.1.4 Central Processing Unit (CPU)

É na CPU onde ficam os registradores, contadores e circuitos lógicos necessários. Ela é responsável pela procura de instruções para serem processadas, decodificação e execução das instruções. Nela, podemos basicamente verificar um ciclo de trabalho bem característico: primeiro a CPU busca por alguma instrução a ser executada na memória, depois decodifica a instrução que deverá ser executada e em seguida executa a tal instrução. (CORRADI JUNIOR, 2006).

2.1.5 LED's do kit microcontrolador

Os LED's de um microcontrolador são listados em uma das portas do kit. A porta utilizada é a P2. O kit utilizado possui 8 LEDs, estando listados da seguinte forma: P2.0 para o primeiro LED, até a P2.7 para o oitavo LED. Para acendê-los, basta igualar o respectivo LED a 0 (zero) e para desligá-los, basta igualá-lo a 1 (um).

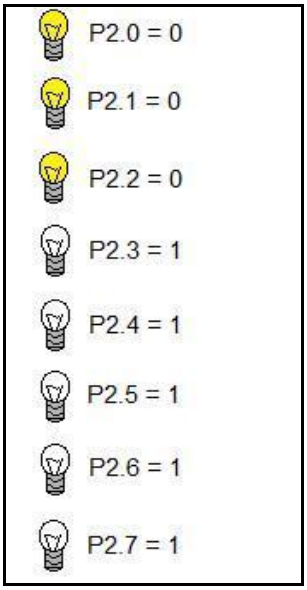

Figura 2.5 - Exemplo de configuração da porta P2

A Figura 2.5 é um exemplo de configuração da porta em que os diodos LEDs estão ligados. No exemplo, os 3 primeiros LEDs foram igualados a 0 para que fossem acesos e o restante foi igualado a 1 para que permanecesse desligado. (CORRADI JUNIOR, 2006).

2.1.6 Porta serial do kit microcontrolador

A família 8051 oferece uma porta serial que pode trabalhar de forma síncrona ou assíncrona em forma *full-duplex*, ou seja, permite o envio e recepção simultâneos. E também possui um buffer onde os dados são armazenados para envio. Com a presença do buffer no microcontrolador é possível receber bits antes mesmo que os bits anteriores já tenham sido lidos. Devido a isso não há interrupção nem na leitura nem na recepção de dados.

2.2 Comunicação Serial

Comunicação em serie é o tipo de conexão em que o transmissor envia um bit de cada vez, e o receptor recebe um bit de cada vez. Contrariando o tipo de conexão paralela que envia e recebe mais de um bit de cada vez. Separando os bits de um em um, é percebida uma melhora com relação à interferência eletromagnética; problema muito frequente nas conexões paralelas. O estabelecimento de uma conexão serial pode ser efetuado de duas formas: síncrono e assíncrono. O modo síncrono é executado com a ajuda de um sinal de relógio que sincroniza o transmissor e o receptor. Nas transmissões e recepções assíncronas, os dados trafegam numa velocidade constante sem sinal de relógio e obedecem a um formato específico. Para o estabelecimento de uma conexão serial assíncrona é preciso que alguns parâmetros sejam definidos. Os parâmetros serão explanados na próxima seção. (CORRADI JUNIOR, 2006).

2.2.1 Normas para conexões seriais

A Figura 2.6 apresenta algumas normas criadas para conexões seriais como: RS422, V35, X.21, V24 e RS 232, sendo que essas duas últimas normas citadas são as mais utilizadas.

Figura 2.6 - Padrão RS232 (3 ligações apenas)

2.2.2 Comunicação serial RS-232

Conforme (Soares, 2009), este padrão prevê a comunicação entre "Equipamentos de Dados – DTE - Data Terminal Equipment" (PC, por exemplo) e "Equipamentos de Comunicação de Dados – DCE - Data Communications Equipment" (modem ou outro periférico). A norma definiu as características elétricas (níveis de sinal e seus respectivos valores) e mecânicas (tipo de conectores e sua respectiva pinagem). O cabeamento pode ser feito com uma grande quantidade de pinos assim como pode ser com poucos. Os cabos mais comuns são o de 25 pinos e o de 9 pinos. O de 25 pinos liga todos os pinos e possui todos os sinais possíveis.

No padrão DB9 que utiliza 9 pinos, o cabo não faz algumas conexões incomuns. É uma versão mais enxuta da versão com 25 pinos conforme figura 2.6. (OMEGA.COM, 2008).

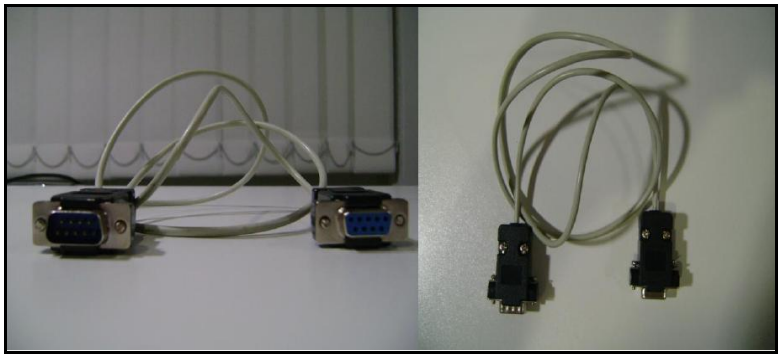

Figura 2.7 - Cabo serial RS-232 DB9.

Fonte (AUTOR DO PROJETO, 2010)

2.3 Bluetooth

Bluetooth é um padrão de comunicação de redes PAN (Personal Area Network). É usado para conectar redes de curta distância e também para a substituição dos cabos. Uma de suas principais utilizações é para a interligação entre palmtops, celulares e periféricos de computador. (MILLER, 2001)

A tecnologia Bluetooth utiliza sinais de rádio frequência visando estabelecer conexões ponto-a-ponto e ponto-a-multiponto para assim realizar a transferência de dados e voz. Essa conexão só existirá entre os dispositivos se ambos contiverem rádios Bluetooth. Cada conexão pode suportar até 8 dispositivos um master (mestre) e 7 slave (escravos), no entanto, é possível fazer com esse número seja maior através da sobreposição de piconets. Isso significa fazer com que uma piconet se comunique com outra dentro de um limite de alcance, esquema esse denominado scatternet. Conforme a figura 2.8 um dispositivo slave pode fazer parte de mais de uma piconet ao mesmo tempo, no entanto, um master só pode ocupar essa posição em uma única.

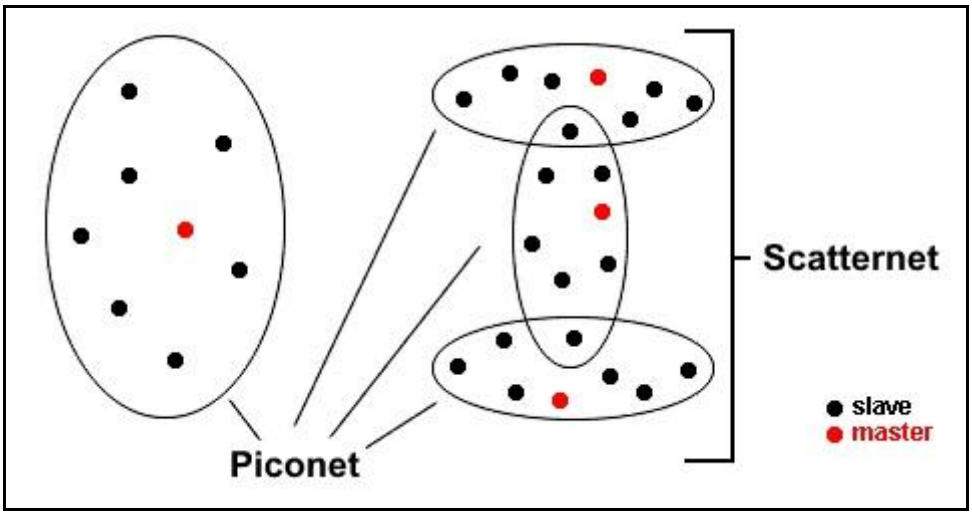

Figura 2.8 - Uma piconet juntamente com uma Scatternet

FONTE: http://www.infowester.com/bluetooth.php

Com o advento da tecnologia *Bluetooth* algumas vantagens foram alcançadas, tais como, menos preocupação com as especificações de fios existentes, consumo de pouca energia, redução na complexidade da interligação de dispositivos, entre outras.

2.3.1 Bluetooth e sua conexão

O *Bluetooth* foi criado para funcionar em todo mundo e devido a isso fez-se necessário a doação de uma frequência de rádio aberta padrão em todo planeta. A faixa ISM (Industrial Scientific Medical), que opera a frequência de 2,45 GHz ainda não é licenciada, e pode ser usada por outros equipamentos de rádio. O *Bluetooth* possui um mecanismo o qual evita a interferência e redução de taxa de transferência de dados. Este mecanismo faz com que a conexão seja dinâmica, vai saltando aleatoriamente e constantemente de frequência em frequência, dentro do escopo determinado, para que a conexão não permaneça tempo bastante transmitindo na mesma frequência para ser interferida. (MILLER, 2001).

Essa tecnologia permite conexões com até 100 metros de distância, porém, dispositivos menores, como o celular, possuem normalmente uma antena com alcance de 10 metros. O padrão atual do *Bluetooth* pode alcançar taxas de conexão de 3Mbps, isso sem contar com o overhead que o protocolo necessita. (MORIMOTO, 2008).

2.4 Tecnologia Java

2.4.1 Visão geral

É uma linguagem Orientada a Objetos, simples, robusta, independente de arquitetura, portável, livre, segura. Com essas características fica claro que é uma linguagem de excelente escolha para várias aplicações. A primeira versão foi lançada em 1995. Ela permitia a programação de aplicações usando o mesmo código não importando o sistema operacional que estivesse executando o programa. Isso acontece devido a um compoente chamado Java Virtual Machine (JVM) que ajuda a traduzir a linguagem para o sistema operacional usado.

A linguagem Java é dividida principalmente de acordo com suas aplicações fim. Elas são: J2ME, J2SE e J2EE. O J2ME que significa Java 2 Micro Edition é voltada para desenvolvimento de aplicações com recursos reduzidos. Esse tipo de programa será executado em dispositivos com processamento e memórias reduzidas. O J2SE que significa Java 2 Standard Edition é a base da plataforma e inclui o ambiente de execução e as bibliotecas

comuns. Já o J2EE que significa Java 2 Enterprise Edition é voltado para o desenvolvimento de aplicações corporativas e para internet. (JOHNSON, 2008)

2.4.2 J2ME

A edição Micro Edition foi desenvolvida para equipamentos de performances reduzidas como PDAs (Personal Digital Assistants, Assistentes Digitais Pessoais), celulares e pagers. Das limitações envolvidas podemos citar: processamento e memórias reduzidas, menores telas com suas menores resoluções. Seria muito bom se pudéssemos utilizar a API (Application Program Interface, Interface de Programação de Aplicativos) J2SE pudesse ser utilizada para o desenvolvimento das aplicações desejadas, porém nem todas as funcionalidades podem ser executadas nesse tipo de dispositivo. (JOHNSON, 2008).

2.5 Motor DC

Motores DC são dispositivos mecânicos que transformam energia elétrica em energia mecânica, em geral energia cinética. São conhecidos por seu controle preciso de velocidade e por seu ajuste fino, sendo utilizados em aplicações que exigem tais características.

Os três principais tipos de motores DC são os de imã permanente, *shunt* (em paralelo), e em serie. Os em série são os motores de arranque, possuem alto torque inicial e alta velocidade máxima.

A figura 2.9 mostra um exemplo de um motor DC.

Figura 2.9 – Exemplo de motor DC

FONTE: http://www.pyroelectro.com/

A utilização dos motores de corrente contínua teve um grande incremento nos últimos anos, graças à eletrônica de potência. Fontes estáticas de corrente contínua com transistores confiáveis, de baixo custo e manutenção simples, substituíram os grupos conversores rotativos. Com isso, motores de corrente contínua passaram a constituir alternativa mais atrativa em uma série de aplicações.

O processo de seleção de um acionamento elétrico corresponde à escolha de um motor que possa atender a, pelo menos, três requisitos do utilizador:

- Fonte de alimentação: tipo, tensão, frequência, simetria, equilíbrio.
- Condições ambientais: agressividade, periculosidade, altitude, temperatura.
- Exigências da carga e condições de serviço: potência solicitada, rotação, esforços mecânicos, configuração física.

Para descobrir o comportamento de um motor DC (seja ele de imã permanente ou tipo *shunt*) é preciso conhecer 4 parâmetros:

- V_{imput} – a tensão aplicada aos terminais (medida em V).
- K_t – a constante de torque do motor, que é a razão entre o torque gerado pelo motor e a corrente elétrica nele aplicada (medida em N.m/A).
- R_{motor} – resistência elétrica entre os terminais do motor (medida em Ω), quanto menor o seu valor maiores são as correntes que o motor consegue puxar, e maior o seu torque.
- I_{no_load} – corrente elétrica requerida para o motor girar sem nenhuma carga no sei eixo (medida em A), quanto menor o seu valor menos atrito existe nos rolamentos.

As equações utilizadas para um motor DC são as seguintes:

$$\tau = K_t \times (I_{input} - I_{no\_load})$$

$$\omega = K_v \times (V_{imput} - R_{motor} \times I_{input})$$

Onde:

- τ – toque aplicado pelo motor em um instante (em N•m).
- ω – velocidade angular do motor (em rad/s).
- I_{input} – corrente elétrica que atua no motor (em A).

- K_v – a constante de velocidade do motor, é a razão entre a velocidade e a tensão nele aplicada, medida em (rad/s)/V, é calculada por $K_v = 1/K_t$.

As equações acima são aproximadas se a corrente não variar bruscamente. A potência elétrica consumida vale $P_{imput} = V_{imput} \times I_{imput}$ e a potencia mecânica gerada pelo motor é $P_{output} = \tau \times \omega$. Querendo maior potência mecânica possível gastando o mínimo de potência elétrica, isso pode ser quantificado pela eficiência $\eta = P_{imput}/ P_{output}$ que resulta em um numero entre 0 e 1.Lembrando que $K_t \times K_v = 1$, as equações resultam em:

$$\eta = \frac{P_{input}}{P_{output}} = \frac{V_{input} \cdot I_{input}}{(I_{input} - I_{no\_load}) \cdot (V_{input} - R_{motor} \cdot I_{input})}$$

Com auxílio das equações acima os resultados são :

- 250 RPM
- $\tau = 0{,}0027422$ N.m
- $\omega = 26{,}17$ Rad/s
- Tensão = $V_{imput} = 3$ Volts
- $I_{no_load} = 20$mA
- $I_{input} = 50$mA
- $K_v = 10{,}94$ (Rad/s)/V
- $K_t = 0{,}0914077B$ (Rad/s)/V
- $R_{Motor} = 12{,}2\ \Omega$
- $P_{Input} = 0{,}15$ W
- $P_{output} = 0{,}717634$ W

CAPÍTULO 3 - PROPOSTA DE SOLUÇÃO E MODELO

A visão do projeto é mostrada na figura 3.1. Na proposta abaixo, cada dispositivo (celular, computador, portão (protótipo) e microcontrolador) tem um software associado.

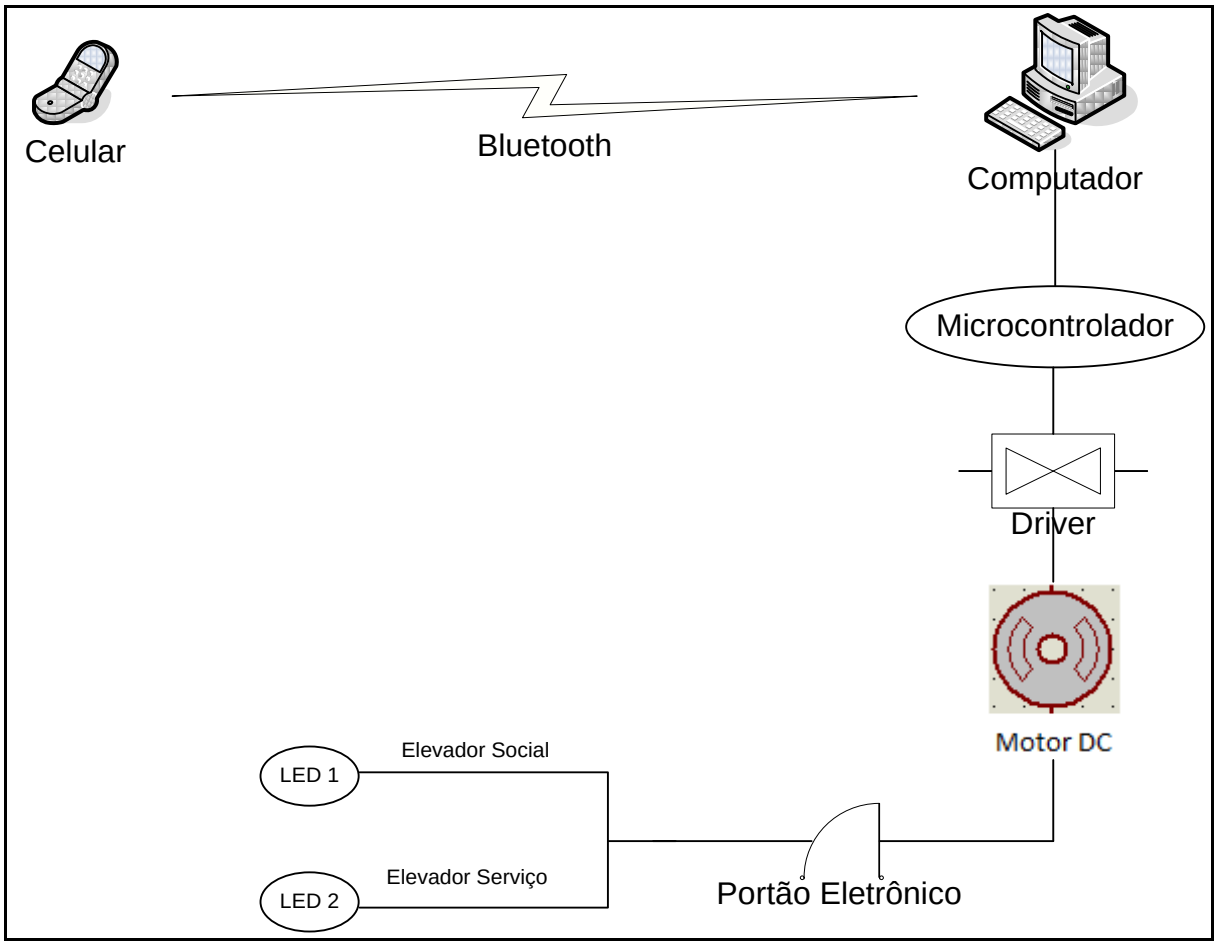

Figura 3.1 - Visão do projeto

FONTE: Autor do Projeto

3.1 Projeto para a Aplicação para o Celular

Desenvolvido na plataforma J2ME a MIDlet (no celular), faz interface com usuário, e possui um comando para estabelecer uma conexão *Bluetooth* com o computador. No software, o usuário pode selecionar o que ele deseja e em seguida enviá-lo para o servidor escolhido

previamente, celular ou computador. Informações de erro de conexão ou outro qualquer deve ser informado pela própria interface. Essa funcionalidade é feita pelo *software*. Quando não obtemos êxito na conexão, esse erro é apresentado para o usuário.

O Fluxograma das operações do celular está representado na figura 3.2, abaixo:

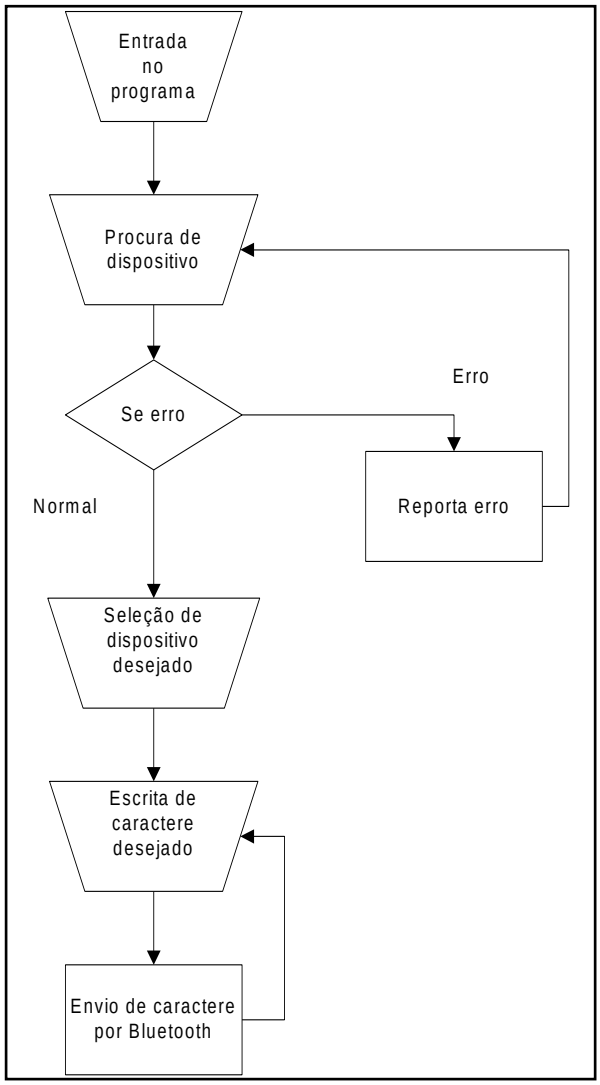

Figura 3.2 - Fluxograma do celular

3.2 Projeto para a Aplicação do Servidor (Computador)

O computador apresentará um software desenvolvido em J2ME, que utilizará uma API especifica para comunicação *Bluetooth* e outra para comunicação serial. A utilização das APIs é necessária para que essa possa fazer um papel de intermediadora na solução, onde

neste caso, é. Fica aguardando a conexão cliente ser estabelecida. Estabelecendo essa comunicação, o computador recebe os dados enviados pelo celular enviando assim para porta serial conectada a um microcontrolador.

O Fluxograma das operações do computador está representado na figura 3.3, abaixo:

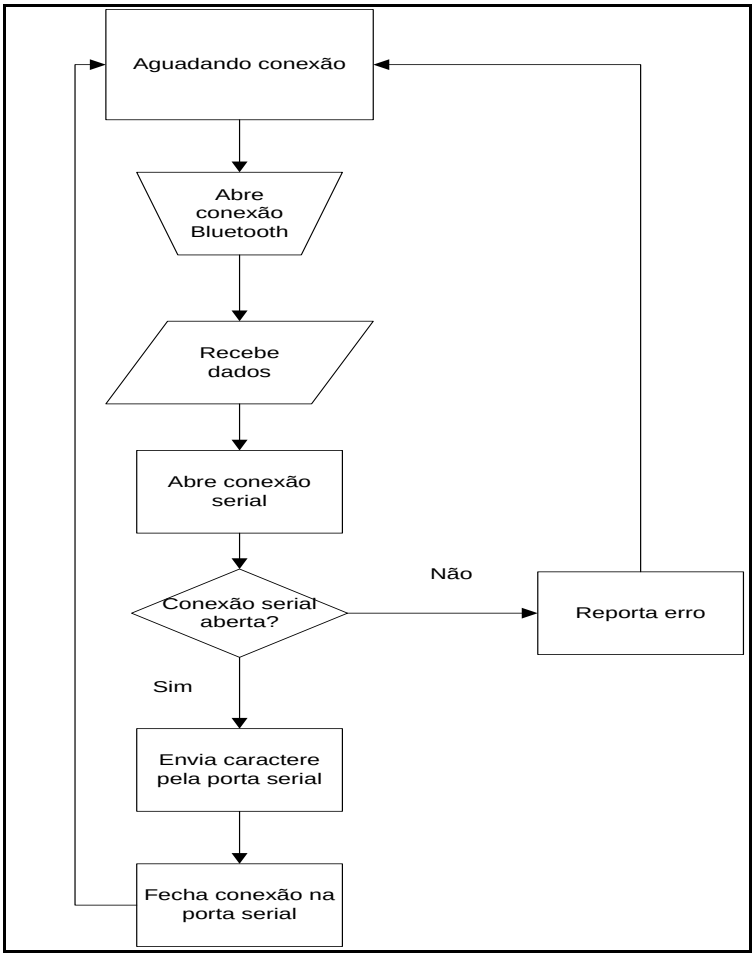

Figura 3.3 – Fluxograma do computador

3.3 Projeto para Aplicação do Microcontrolador

O microcontrolador é programado na linguagem C. O programa faz com que este microcontrolador receba os dados da porta serial e armazene em seu *buffer*, fazendo a comparação dos dados com as opções permitidas. Posteriormente executa os comandos relativos às opções. O diagrama com as execuções e comparações feitas pelo *software* do microcontrolador é mostrado na figura 3.4.

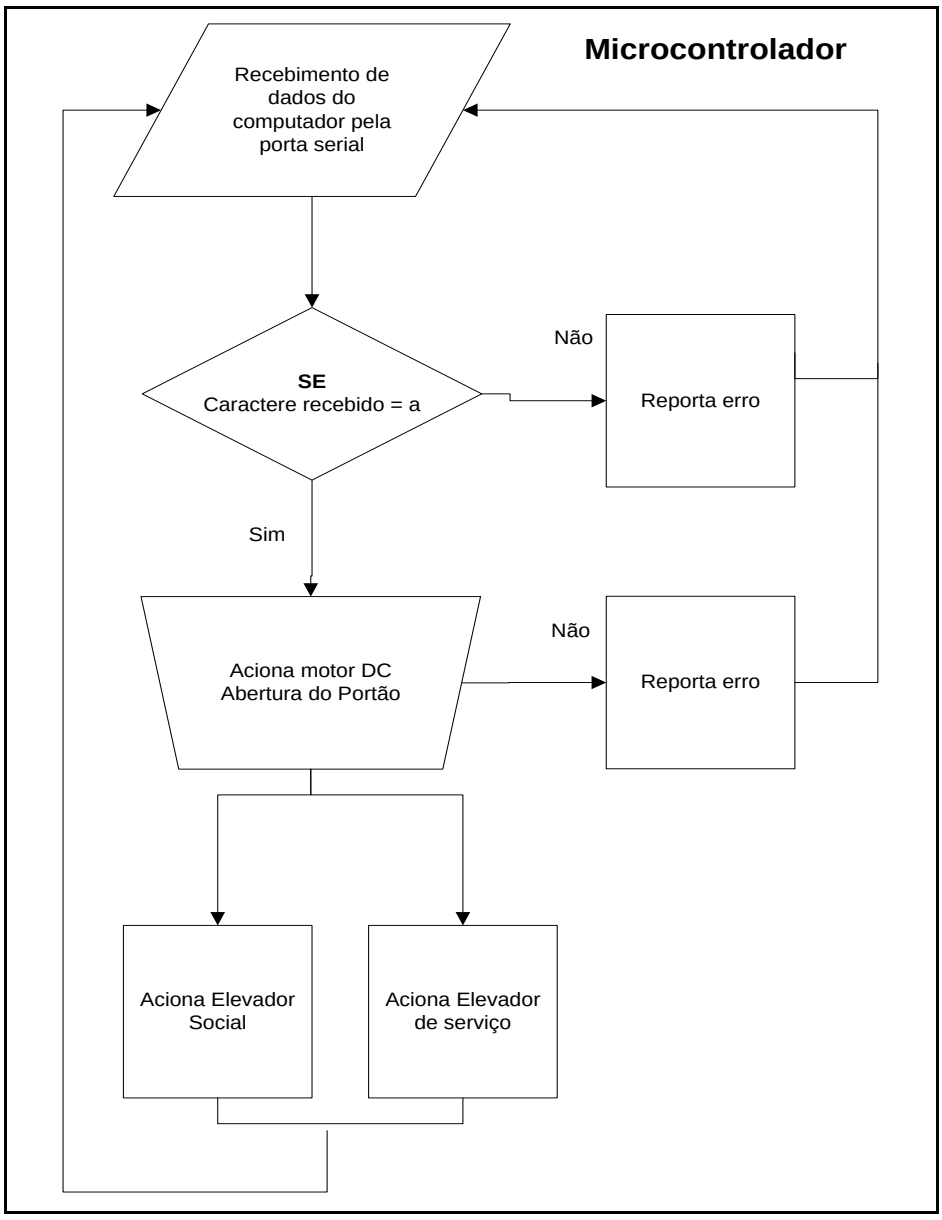

Figura 3.4 – Fluxograma do microcontrolador

3.4 Projeto de Envio e Recebimento de Dados entre os Dispositivos

Os programas devem fazer as comunicações uns com os outros. A figura 3.5 ilustra o estabelecimento de conexão entre os dispositivos e também ilustra o envio de dados entre eles.

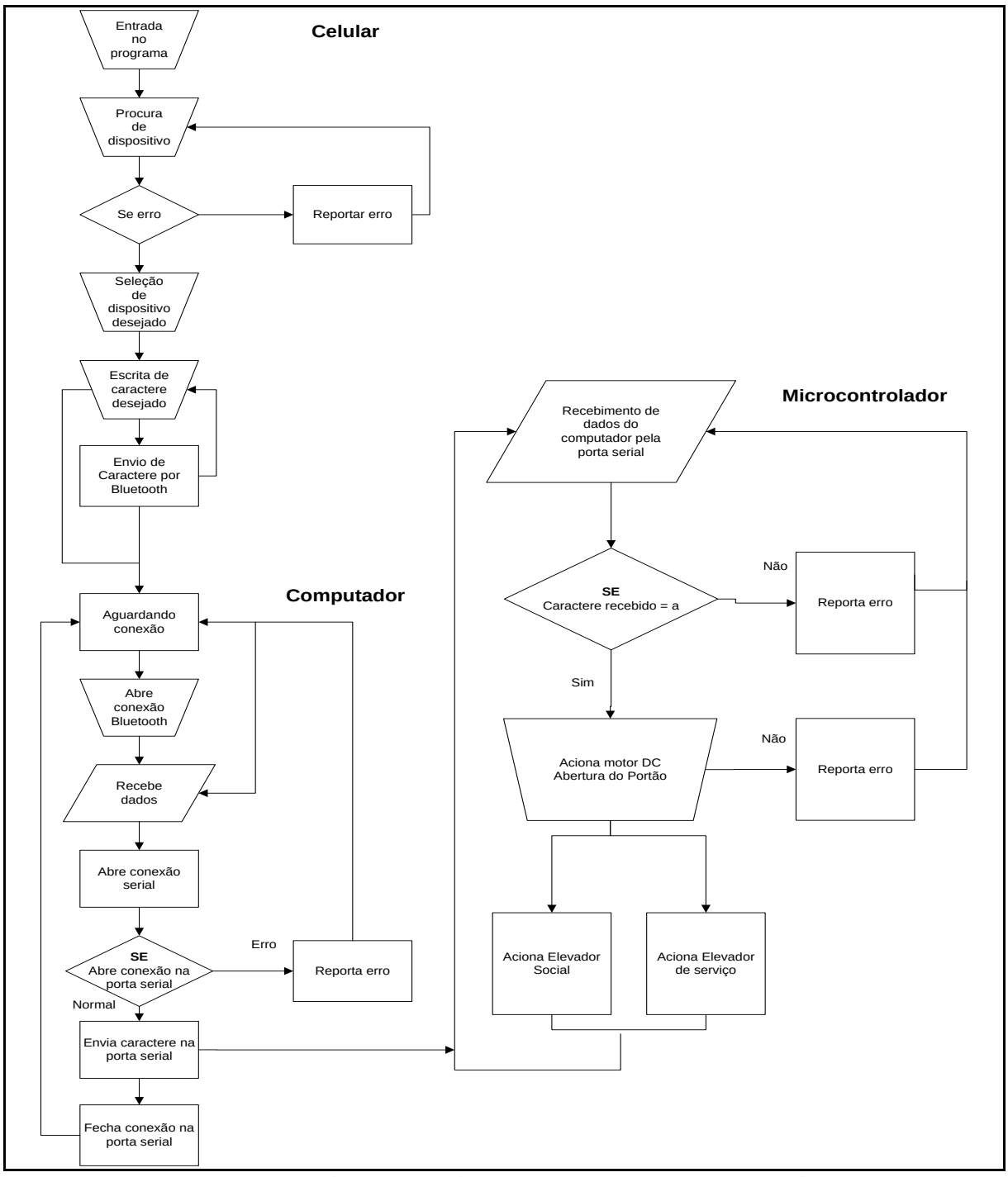

Figura 3.5 - Fluxograma dos equipamentos se comunicando

CAPÍTULO 4 - DESENVOLVIMENTO DO TRABALHO

O Sistema de Automação Predial utilizando celular com tecnologia *Bluetooth* foi desenvolvido utilizando componentes de comunicação, celular, controladores, motor etc.

As macro etapas do desenvolvimento foram:

- Software de comunicação entre o celular e o servidor.
- Software de comunicação entre o servidor e o microcontrolador.
- Desenvolvimento no software Keil que foi utilizado para compilar o programa para gravação no microcontrolador.
- Confecção do Driver.
- Integração dos componentes anteriormente descritos, para verificar a comunicação de todos.

4.1 Descrição das Etapas do Trabalho

4.1.1 Hardware

- 1 computador.
- 1 Adaptador *Bluetooth* para o computador.
- 1 celular com *Bluetooth*.
- 1 kit de microcontrolador 8051.
- 1 cabo serial DB9 padrão RS-232.
- 1 cabo *flat* de 10 vias.
- 1 cabo para gravação.
- *Protoboard*.
- Motor DC.
- *Driver* para acionamento do motor.

Os equipamentos utilizados no projeto estão ilustrados na Figura 4.1 a seguir:

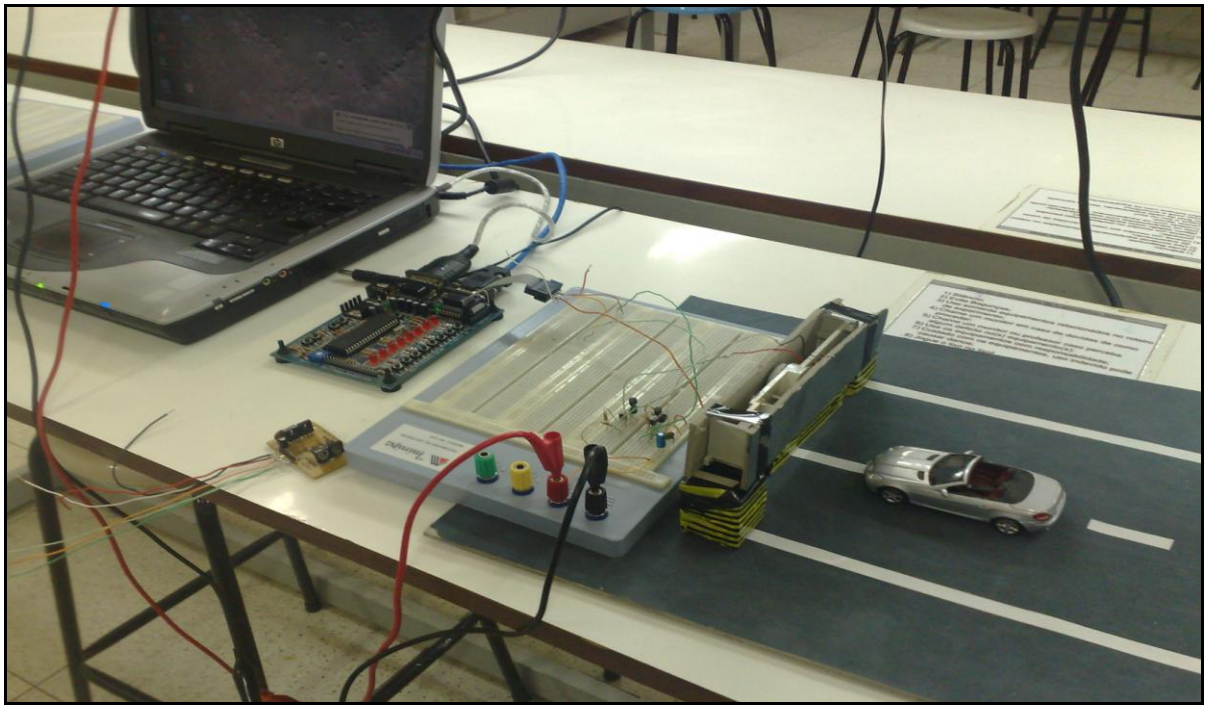

Figura 4.1 - Ligações entre dispositivos (Foto)

4.1.2 Aplicação do software no celular

A tecnologia Java, utilizada no desenvolvimento do projeto, é muito utilizada em dispositivos móveis. Além de fornecer os recursos de conectividade, persistência de dados, a portabilidade foi o motivo principal para a escolha desta tecnologia, fazendo com que a aplicação funcione independentemente da plataforma, bastando para isso, utilizar um celular.

Para um melhor entendimento desse projeto, é importante conhecer alguns conceitos:

- Lista: É a tela que é mostrada no display do celular que lista opções. No caso, a lista que será mostrada ao usuário é de aparelhos disponíveis para acesso.
- Textbox: É a tela que é mostrada no display onde o usuário deve escrever o que é pertinente à aplicação. O textbox é mostrado ao usuário para digitar o caractere correspondente à opção que ele deseja.
- startApp: É uma função utilizada na inicialização do programa.

- CommandAction: Programa que aguarda a execução dos eventos. Possui 4 tipos de comandos internamente: o comando saída, o comando “ok” e dois comandos de seleção.
- CommandListener: Detecta o comando e inicia o commandAction.
- acharDispositivos: Procura por dispositivos.
- acharServicos: Procura por serviços na sua área de cobertura.
- do_alert: Apresenta no display alguns avisos.
- destroyApp: Utilizado para finalizar o programa.

O detalhamento de todos os códigos utilizados neste trabalho está descrito no Apêndice I.

4.1.3 Aplicação do software no computador

Para que o software seja executado no computador conforme esperado, foi necessário desenvolver outras classes para conexão com o *Bluetooth*. A primeira classe possui a execução da conexão *Bluetooth* e chama as funções presentes na segunda classe, que por sua vez realiza a comunicação com a porta serial.

Alguns conceitos que precisam ser conhecidos:

- Server: Inicia o processo de comunicação.
- executaConfiguracoesIniciais: Configura o dispositivo *Bluetooth* com os parâmetros iniciais recebidos.
- processaDados: Executa as funções que estão na classe Serial.java. Possui somente funções que trabalham com a conexão serial.
- ObterIdDaPorta: Verifica se a porta COM está operante.
- EnviarUmaString: Envia o caractere recebido pelo *Bluetooth*.
- FecharCom: Finaliza a conexão serial.

4.1.4 Aplicação do software no microcontrolador

A aplicação desenvolvida para o microcontrolador tem o objetivo de receber dados da porta serial e comparar com caracteres pré-definidos. Se o dado recebido for igual aos caracteres que ele espera receber, então, o microcontrolador abre o portão conforme caractere recebido.

O *software* possui rotinas definidas para a execução de funções de acordo com caracteres recebidos que são predefinidos. As rotinas abaixo executam a mesma ação "Abrir Portão":

- Seri: Verificar se recebeu o caractere "a" e executa a ação.
- Limpa: Verificar se recebeu o caractere "s" e executa a ação.
- Limpa1: Verificar se recebeu o caractere "d" e executa a ação.
- Limpa2: Verificar se recebeu o caractere "z" e executa a ação.
- Limpa3: Verificar se recebeu o caractere "x" e executa a ação.
- Limpa4: Verificar se recebeu o caractere "c" e executa a ação.

Por exemplo: se o microcontrolador receber um caractere "z", inicialmente a rotina "seri" é executada, que o compara com o caractere "a". Como o caractere não é "a", então ele entra na próxima rotina até que haja igualdade entre o caractere recebido e o caractere da rotina. Caso o caractere recebido não seja o esperado em nenhuma das funções, o *software* não abre o portão.

4.1.5 Execução do software no celular

Uma vez conectados todos os cabos e aparelhos, inicia-se a execução do projeto. Primeiramente o dispositivo *Bluetooth* do celular e do computador são ativados. Após garantir que a conexão *Bluetooth* está funcionando o aplicativo presente no celular chamado Controle Universal deve ser iniciado, conforme apresentado na Figura 4.2.

Figura 4.2 - Aplicativos disponíveis no celular

Quando o aplicativo é iniciado, é apresentada uma opção em que o celular inicia a busca por dispositivos que possuam *Bluetooth* e forneçam o serviço desenvolvido no projeto. A opção é ilustrada na Figura 4.3

Figura 4.3 - Imagem ao entrar no aplicativo desenvolvido

Para que o usuário seja informado que o aparelho celular está procurando dispositivos, é exibido um alerta no visor do aparelho informando: “Procurando dispositivos...”, como na figura 4.4.

Figura 4.4 - Procurando dispositivos

Finalizada a pesquisa por dispositivos que se enquadram nos requisitos, ou seja, equipamentos que forneçam o serviço solicitado e que possuam a tecnologia *Bluetooth*, uma lista é mostrada ao usuário. O usuário deve, então, selecionar o dispositivo desejado. No exemplo da figura 4.5, só existe uma opção: HUGO-PC. Essa é a opção que deve ser escolhida, pois é esse o nome atribuído ao computador servidor.

Figura 4.5 - Dispositivos que oferecem o serviço

Após a seleção do servidor ao qual o usuário deve se conectar, o celular apresenta em seguida, uma lista de opções onde deverá ser escolhido o que será transmitido para o computador. No exemplo, os caracteres “a”, “s”, “d”, “z”, ”x” e ”c” abrem o portão.

Quando o aparelho celular entra em contato, o servidor exibe outra mensagem relatando que há um cliente conectado. Com a conexão concluída, o cliente está pronto para o envio do caractere.

4.1.6 Execução do software no computador servidor

A ferramenta Eclipse para desenvolvimento em Java é usada para que o código seja executado e mostre no console seus resultados.

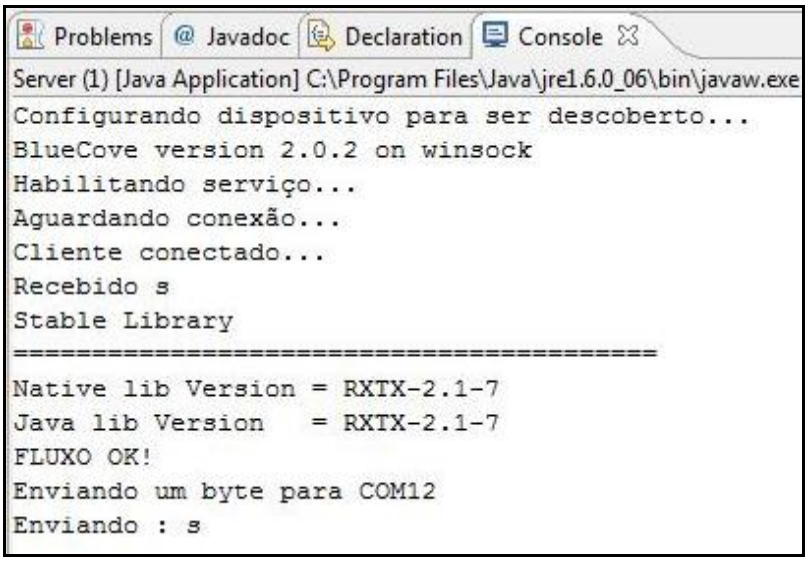

Figura 4.6 – Execução da ação no servidor

As quatro primeiras linhas apresentadas na figura 4.6 representam a iniciação do serviço. Primeiramente, o código configura o dispositivo para ser encontrado por outros aparelhos. A segunda linha mostra a API usada para a conexão *Bluetooth*. Na terceira e quarta, o programa exibe mensagens informando que o servidor está começando a executar o serviço e está pronto esperando alguma conexão cliente.

Quando o aparelho celular se conecta ao servidor este exibe uma mensagem relatando que há um cliente conectado. Com a conexão concluída, o cliente está pronto para o envio do caractere. A figura 4.6 apresenta que o caractere enviado foi o “s”.

Seguindo a ordem, as linhas de 5 à 9 dizem respeito à outra API utilizada na codificação do trabalho. Ela é a API de comunicação serial, indispensável para a conexão com o microcontrolador. O restante das linhas diz respeito ao envio do caractere pela porta serial que tem do outro lado o microcontrolador esperando o caractere em questão.

Caso o cabo serial não esteja conectado ao computador, serão apresentadas as mensagens: “Não foi possível estabelecer uma conexão com a porta selecionada” e “Ocorreu um erro na tentativa de envio do comando para o Microcontrolador.” Conforme apresentado na figura 4.7.

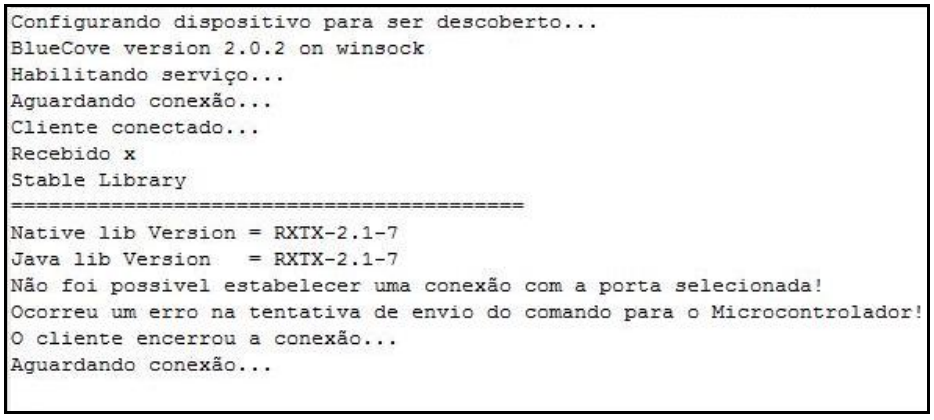

Figura 4.7 – Execução da ação no servidor

4.1.7 Execução do software no microcontrolador

Este programa faz comunicação serial entre o computador servidor e o microcontrolador. Os parâmetros da conexão escolhidos são os seguintes:

- Envio de bits = 4800bps.
- Bits de dados = 8bits de dados.
- Paridade = sem paridade.
- Bits de parada = 1 bit de parada.
- Controle de fluxo = nenhum.

Esses parâmetros devem ser os mesmos nas duas pontas do cabo serial para que a comunicação seja efetuada com sucesso.

O programa no microcontrolador, com o auxílio do *Driver* desenvolvido e o motor DC são responsáveis pela abertura do portão.

O programa estabelece uma conexão serial *full duplex* com o computador, ou seja, permite o recebimento e envio de informações simultaneamente. O microcontrolador espera o recebimento dos caracteres predefinidos. Após o recebimento dos caracteres específicos, o microcontrolador enviará comandos de nível alto e baixo (10-00-01), fazendo com que o portão seja aberto e fechado. Posteriormente esse comando aciona o elevador mais próximo do subsolo.

CAPÍTULO 5 - CONCLUSÃO

A tendência de aplicar controle em edifícios é cada vez mais presente, independentemente de seus coeficientes de inteligência, pois o que varia é o grau de sofisticação do controle.

Este projeto apresenta uma proposta de automação de garagens de prédios residenciais usando a tecnologia *Bluetooth*. Para isso, foi construído um protótipo em baixa escala como definido no trabalho. Para utilização desse projeto na vida real basta adequá-lo à realidade substituindo os materiais de baixa escala. Exemplo, a troca do motor DC para um motor de indução.

As tarefas propostas para a realização deste trabalho foram executadas com sucesso na montagem da maquete. As informações do celular foram capturadas pelo sistema onde o computador fez a comunicação com a porta serial. A abertura do portão eletrônico foi realizada através do comando enviado pela porta serial onde o microcontrolador foi programado para verificar qual o elevador está mais próximo para ser acionado e direcionado para o subsolo. A programação do sistema foi feita através dos programas desenvolvidos em linguagem JAVA integrando os sistemas mecânicos com o sistema do microcontrolador, que foi programado utilizando a linguagem “C”. Circuitos com chaves ópticas para garantir se o portão abriu corretamente foram implementados e funcionaram perfeitamente. Por fim, através de somente um comando, conseguimos fazer com que o portão eletrônico fosse aberto e o elevador direcionado para o subsolo aguardando o usuário. Alcançando assim todos os objetivos e tarefas propostas para este projeto.

Os resultados obtidos com o protótipo foram satisfatórios para os objetivos iniciais. A principal vantagem do projeto proposto é ter a comodidade de usar o celular como controle e assim que se estaciona já haverá um elevador aguardando no subsolo.

5.1 Trabalhos Futuros

Com as pesquisas e estudos para confecção desse trabalho final surgiram idéias para futuros projetos. O *Bluetooth* é bastante utilizado em áreas diversas por trazer inúmeras vantagens. Alguns fatores presentes no projeto que podem ser alterados e melhorados são listados a seguir:

1. Cadastrar os endereços MAC (*Media Access Control*) dos celulares de cada morador do edifício no computador servidor.
2. Cadastrar as portarias de entrada de cada morador.
3. Ao receber o pedido de conexão de um celular, o computador servidor deverá identificar se aquele celular é o mesmo cadastrado, visando aumentar a segurança do prédio, e consequentemente de todos os condôminos.
4. Para as ações de erro ou sucesso, devem ser exibidas mensagens informativas na tela do celular.
5. Verificar qual a portaria do morador e acionar o elevador, mais próximo do subsolo (social ou serviço), desta portaria.
6. Apresentar na tela do celular a informação de qual elevador foi acionado e se este já se encontra no subsolo.
7. Desenvolver um atalho para iniciação do programa no celular.
8. Instalação de sensores no portão de entrada da garagem visando o não fechamento deste antes da passagem total do veículo.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] CORRADI JUNIOR. Microcontrolador 8051, Colégio Técnico de Campinas, 2006

[2] FOROUZAN, BEHROUZ A. Comunicação de Dados e Redes de Computadores, Ed Porto Alegre, 2006

[3] GIMENEZ, SALVADOR P. Microcontroladores 8051, Ed Prentice Hall, 2002

[4] HORSTMANN, CAY S. Core JAVA 2 Volume I – Fundamentos, Editora Makron, 2001

[5] JOHNSON, THIENNE M. Java para dispositivos móveis, Editora Novatec, 2008

[6] SOARES, M. J.. Saber Eletrônica Nº 424 Ed. Saber, 2008

[7] MILLER, MICHAEL. Descobrindo Bluetooth, Editora Campus, 2001

[8] MORIMOTO, Carlos; Bluetooth, disponível em http://www.guiadohardware.net/artigos/bluetooth/ - acessado em 05/06/2010

[9] OMEGA.COM; The RS-232 Standard, disponível em http://www.omega.com/TechRef/pdf/RS-232.pdf - acessado em 06/06/2010

[10] http://www.das.ufsc.br/~werner/eel7030/8051/Apostila8051Hari.pdf - Acessado em 24/08/10

[11] http://grenoble.ime.usp.br/movel/monografia_bluetooth.pdf - Acessado em 10/09/10

[12] http://pt.wikipedia.org/wiki/Bluetooth - Acessado em 10/09/10

[13] http://developer.java.sun.com/developer/onlineTraining - Acessado em 10/09/10

[14] http://java.web.cern.ch/java/tutorials.html - Acessado em 12/09/10

[15] http://www.java.com/pt_BR/download/faq/whatis_j2me.xml - Acessado em 14/09/10

[16] http://www.rogercom.com/ - Acessado em 14/09/10

[17] http://pt.wikipedia.org/wiki/Endere%C3%A7o_MAC - Acessado em 01/10/10

[18] www.proware.ind.br/Apostila%20Elet.%20Digital.pdf - Acessado em 24/08/10

APÊNDICE I

Aqui será disposto o código utilizado para realização das atividades previstas no projeto de conclusão do curso. Para a execução do código foi necessária a utilização de APIs (Application Program Interface) específicas para o tratamento da conexão serial e para a conexão Bluetooth. A API utilizada para comunicação serial foi a bluecove versão 2.0.2; e a API para comunicação serial foi a RXTXcomm.

Código do servidor

Em seguida será apresentado o código que está residindo e executando no computador:

Classe Serial.java

```
/**
 * Projeto Final - UNICEUB
 * Proposta de Automação Predia Utilizando Celular com Tecnologia Bluetooth
 * Paulo Henrique
 *
 */
import gnu.io.CommPortIdentifier;
import gnu.io.SerialPort;
import java.io.OutputStream;

import javax.swing.JOptionPane;
```

```
/**
 *
 * Classe representativa da porta Serial.
 *
 */
public class Serial {

	public String Dadoslidos;
	public int nodeBytes;
	private int baudrate;
	private int timeout;
	private CommPortIdentifier cp;
	private SerialPort porta;
	private OutputStream saida;
	private String Porta;
	protected String peso;

	public Serial(String p, int b, int t) {   //parâmetros para conexão serial
		this.Porta = p;
		this.baudrate = b;
		this.timeout = t;
	}
	/**
	 * Obtem o ID da porta.
	 * @return Boolean
	 */
```

```
public boolean ObterIdDaPorta() {    // Verifica se a porta está pronta
        try {
                cp = CommPortIdentifier.getPortIdentifier(Porta);
                if (cp == null) {
                        JOptionPane.showMessageDialog(null, "Erro na abertura da porta cp : "+cp==null);
                        System.out.println("Erro na abertura da porta!");
                        return false;
                }
        } catch (Exception e) {
                JOptionPane.showMessageDialog(null, "Erro na abertura da porta exception "+e.getMessage());
                System.out
                                .println("Não foi possivel estabelecer uma conexão com a porta selecionada!");
                return false;
        }
        return true;
}
/**
 * Abre a porta
 * @return boolean
 */
public boolean AbrirPorta() {           //Abre conexão na porta serial
        try {
                porta = (SerialPort) cp.open("SerialComLeitura", timeout);
                porta.setSerialPortParams(baudrate, porta.DATABITS_8,
```

```
                                porta.STOPBITS_1, porta.PARITY_NONE);
                porta.setFlowControlMode(SerialPort.FLOWCONTROL_NONE);
        } catch (Exception e) {
                System.out.println("Erro abrindo comunicação: " + e);
                return false;
        }
        return true;
}
/**
 * Envia mensagem
 * @param msg
 * @return boolean
 */
public boolean EnviarUmaString(String msg) {
//envia a string recebida pela conexão Bluetooth
        try {
                saida = porta.getOutputStream();
                System.out.println("FLUXO OK!");
        } catch (Exception e) {
                System.out.println("Erro.STATUS: " + e);
                return false;
        }
        try {
                System.out.println("Enviando um byte para " + Porta);
                System.out.println("Enviando : " + msg);
                saida.write(msg.getBytes());
```

```
                Thread.sleep(100);
                saida.flush();
        } catch (Exception e) {
                System.out.println("Houve um erro durante o envio. ");
                System.out.println("STATUS: " + e);
                return false;
        }
        return true;

}
/**
 * Método de execução
 */
public void run() {
        try {
                Thread.sleep(5);
        } catch (Exception e) {
                System.out.println("Erro de Thred: " + e);
        }
}
/**
 * Fecha a Porta
 * @return boolean
 */
public boolean FecharCom() {                    //Fecha comunicação na porta serial
        try {
```

```
                porta.close();
            } catch (Exception e) {
                System.out.println("Erro fechando porta: " + e);
                return false;
            }
            return true;
        }
        /**
         * Retorna a porta.
         * @return String
         */
        public String obterPorta() {            //retorna a porta

            return Porta;
        }
        /**
         * Retorna Baudrate (um parâmetro para conexão serial)
         * @return int
         */
        public int obterBaudrate() {
            return baudrate;
        }
}
```

Classe Server.java

```
        /**
```

```
 * Projeto Final - UNICEUB
 * Proposta de Automação Predia Utilizando Celular com Tecnologia Bluetooth
 * Paulo Henrique
 *
 */
import java.util.Enumeration;

import javax.swing.JOptionPane;

import gnu.io.CommPortIdentifier;
/**
 * Classe de comunicação com porta serial
 *
 *
 */
public class ComunicacaoSerial {

	/**
	 * Lista as portas disponíveis
	 */
	static void listPorts() {
		Enumeration portEnum = CommPortIdentifier.getPortIdentifiers();
		while (portEnum.hasMoreElements()) {
			CommPortIdentifier portIdentifier = (CommPortIdentifier)
portEnum
					.nextElement();
```

```
                        System.out.println(portIdentifier.getName() + " - "
                                                +
getPortTypeName(portIdentifier.getPortType()));
                    }
                }
                /**
                 * Obtem porta por tipo
                 * @param portType
                 * @return String
                 */
                static String getPortTypeName ( int portType )
                   {
                      switch ( portType )
                      {
                         case CommPortIdentifier.PORT_I2C:
                            return "I2C";
                         case CommPortIdentifier.PORT_PARALLEL:
                            return "Parallel";
                         case CommPortIdentifier.PORT_RAW:
                            return "Raw";
                         case CommPortIdentifier.PORT_RS485:
                            return "RS485";
                         case CommPortIdentifier.PORT_SERIAL:
                            return "Serial";
                         default:
                            return "unknown type";
```

```
        }
    }
    /**
     * Método principal
     * @param args
     */
    public static void main(String[] args) {
        // Para saber qual porta a se conectar...
        //listPorts();
        try {
            //Adicionar a porta...
            String PORTA =  JOptionPane.showInputDialog(null, "Digite a porta exemplo: COM11 "); //"COM11";
            int BAUDRATE = new Integer( JOptionPane.showInputDialog(null,"Digite o BAUDRATE exemplo: 4800 ") ) ;//4800;
            int TIMEOUT = 0;
            Serial serial = new Serial(PORTA, BAUDRATE, TIMEOUT);
            if(serial.ObterIdDaPorta()){
                serial.AbrirPorta();
                String msn = JOptionPane.showInputDialog(null, "Digite a Mensagem : ");
                boolean retorno = serial.EnviarUmaString(msn);
                if(retorno)
                    JOptionPane.showMessageDialog(null, "MENSAGEM ENVIADA COM SUCESSO...!!");
            }else{
```

```
                                    JOptionPane.showMessageDialog(null, "Porta não foi aberta...!!! erro verifacar uso da porta...");
                                }

                            } catch (Exception e) {
                                e.printStackTrace();
                                JOptionPane.showMessageDialog(null, "ERRO "+e.getMessage());
                            }

                        }

                }
```

Código do aparelho celular

```
/**
 * Projeto Final - UNICEUB
 * Proposta de Automação Predia Utilizando Celular com Tecnologia Bluetooth
 * Paulo Henrique
 *
 */
package br.com.automacao.jme;

import java.io.DataOutputStream;

import javax.bluetooth.DeviceClass;
```

```
import javax.bluetooth.DiscoveryAgent;
import javax.bluetooth.DiscoveryListener;
import javax.bluetooth.LocalDevice;
import javax.bluetooth.RemoteDevice;
import javax.bluetooth.ServiceRecord;
import javax.bluetooth.UUID;
import javax.microedition.io.Connector;
import javax.microedition.io.StreamConnection;
import javax.microedition.lcdui.Alert;
import javax.microedition.lcdui.Choice;
import javax.microedition.lcdui.Command;
import javax.microedition.lcdui.CommandListener;
import javax.microedition.lcdui.Display;
import javax.microedition.lcdui.Displayable;
import javax.microedition.lcdui.List;
import javax.microedition.lcdui.TextBox;
import javax.microedition.lcdui.TextField;
import javax.microedition.midlet.MIDlet;
/**
 *
 * Midlet, responsável por achar dispositivos bluetooth.
 *
 */
public class ProjetoFinal extends MIDlet implements CommandListener,
                    DiscoveryListener {
          List lista_princ, lista_disp;
```

```
Command saida, ok;
TextBox cmd;
Display display;
java.util.Vector dispositivos, servicos;
LocalDevice local;
DiscoveryAgent agente;
DataOutputStream dout;
int currentDevice = 0;

private static Object lock = new Object();

// Usado como indicador para o dispositivo contactado pelo servidor

public void startApp() {
        lista_princ = new List("Automacao residencial", Choice.IMPLICIT);
        // Menu principal
        lista_disp = new List("Selecione o dispositivo", Choice.IMPLICIT);
        // lista de dispositivos
        cmd = new TextBox("a,s,d ligam z,x,c desligam", "", 1, TextField.ANY);
        saida = new Command("Sair", Command.EXIT, 1);
        ok = new Command("Enviar", Command.OK, 1);
        display = Display.getDisplay(this);

        lista_princ.addCommand(saida);
        lista_princ.setCommandListener(this);
        lista_disp.addCommand(saida);
```

```
        lista_disp.setCommandListener(this);
        cmd.addCommand(ok);
        cmd.addCommand(saida);
        cmd.setCommandListener(this);

        lista_princ.append("Procurar servidor", null);
        display.setCurrent(lista_princ);
}

public void commandAction(Command com, Displayable dis) {
        if (com == saida) { // Botão saida apertado
                destroyApp(false);
                notifyDestroyed();
        }
        if (com == List.SELECT_COMMAND) {
                if (dis == lista_princ) { // Após selecionar o celular procura
                        if (lista_princ.getSelectedIndex() >= 0) { // dispositivos
                                acharDispositivos();
                                //new BluetoothDeviceDiscovery().start();
                                do_alert("Procurando dispositivos...", Alert.FOREVER);
                        }
                }
                if (dis == lista_disp) { // Seleção de dispositivo
                        StreamConnection con = null;
                        ServiceRecord servico = (ServiceRecord) servicos
                                        .elementAt(lista_disp.getSelectedIndex());
```

```
                    String url = servico.getConnectionURL(
ServiceRecord.NOAUTHENTICATE_NOENCRYPT, false);
                    try {
                        con = (StreamConnection) Connector.open(url);
                        // Estabelece a conexão
                        dout = new
DataOutputStream(con.openOutputStream());
                        // Pega o output stream para ser enviado
                        display.setCurrent(cmd);
                        // Mostra a TextBox
                    } catch (Exception e) {
                        this.do_alert("Erro na conexao", 4000);
                    }

                }
            }
            if (com == ok) { // usuário mandando comando
                try {
                    dout.writeChars(cmd.getString());
                    dout.flush();
                    cmd.setString("");
                } catch (Exception e) {
                    this.do_alert("Erro no envio de dados", 4000);
                }
            }
        }
```

```
public void acharDispositivos() {
        // procura dispositivos
        try {
                dispositivos = new java.util.Vector();
                LocalDevice local = LocalDevice.getLocalDevice();
                System.out.println("Address: " + local.getBluetoothAddress());

                System.out.println("Name: " + local.getFriendlyName());

                DiscoveryAgent agent = local.getDiscoveryAgent();
                agent.startInquiry(DiscoveryAgent.GIAC, this);

                try {

                        synchronized (lock) {

                                lock.wait();

                        }

                }

                catch (InterruptedException e) {

                        e.printStackTrace();
```

```
                }

        } catch (Exception e) {
                this.do_alert("Erro no inicio da procura", 4000);
        }
}

public void acharServicos(RemoteDevice device) {
        // procura serviços
        try {
                UUID[] uuids = new UUID[1];
                uuids[0] = new UUID("27012f0c68af4fbf8dbe6bbaf7aa432a", false);
                // UUID de cada serviço
                local = LocalDevice.getLocalDevice();
                agente = local.getDiscoveryAgent();
                agente.searchServices(null, uuids, device, this);
        } catch (Exception e) {
                this.do_alert("Erro no inicio da procura", 4000);
        }
}

/**
 *
 * This call back method will be called for each discovered bluetooth
 * devices.
```

```
 */

        public void deviceDiscovered(RemoteDevice btDevice, DeviceClass cod) {

                System.out.println("Device discovered: "
                                + btDevice.getBluetoothAddress());

                // add the device to the vector

                if (!dispositivos.contains(btDevice)) {

                        dispositivos.addElement(btDevice);

                }

        }

        public void servicesDiscovered(int transID, ServiceRecord[] serviceRecord) {
                // Método chamado pela interface DiscoveryListener
                // Chamado quando é encontrado serviço durante a procura
                for (int x = 0; x < serviceRecord.length; x++)
                        servicos.addElement(serviceRecord[x]);
                try {
                        lista_disp.append(((RemoteDevice) dispositivos
                                        .elementAt(currentDevice)).getFriendlyName(false),
null);
```

```
        } catch (Exception e) {
                this.do_alert("Erro no inicio da procura", 4000);
        }
}

public void inquiryCompleted(int param) {
        synchronized (lock) {

                lock.notify();

        }
        // Método chamado pela interface DiscoveryListener
        // Chamado quando a inquiry está completa
        switch (param) {
        case DiscoveryListener.INQUIRY_COMPLETED:
                // Inquiry completada normalmente
                if (dispositivos.size() > 0) {
                        // Pelo menos um dispositivo foi encontrado
                        servicos = new java.util.Vector();
                        this.acharServicos((RemoteDevice) dispositivos.elementAt(0));
                        // verifica se o primeiro dispositivo oferece o serviço
                } else
                        do_alert("Dispositivos nao encontrados", 4000);
                break;
        }
}
```

```
public void serviceSearchCompleted(int transID, int respCode) {
        // Método chamado pela interface DiscoveryListener
        // Chamado quando a procura por serviços foi completada
        switch (respCode) {
        case DiscoveryListener.SERVICE_SEARCH_COMPLETED:
                if (currentDevice == dispositivos.size() - 1) {
                        // Todos os dispositivos foram procurados
                        if (servicos.size() > 0) {
                                display.setCurrent(lista_disp);
                        } else
                                do_alert("O servico nao foi encontrado", 4000);
                } else { // Procura o próximo dispositivo
                        currentDevice++;
                        this.acharServicos((RemoteDevice) dispositivos
                                        .elementAt(currentDevice));
                }
                break;
        case DiscoveryListener.SERVICE_SEARCH_DEVICE_NOT_REACHABLE:
                this.do_alert("Dispositivo nao alcancavel", 4000);
                break;
        case DiscoveryListener.SERVICE_SEARCH_ERROR:
                this.do_alert("Erro na procura de servico", 4000);
                break;
        case DiscoveryListener.SERVICE_SEARCH_NO_RECORDS:
                this.do_alert("Registros nao encontrados", 4000);
```

```
                break;
        case DiscoveryListener.SERVICE_SEARCH_TERMINATED:
                this.do_alert("Inquiry Cancelada", 4000);
                break;
        }
}

public void do_alert(String msg, int time_out) {
        // Emite alertas no display do celular
        if (display.getCurrent() instanceof Alert) {
                ((Alert) display.getCurrent()).setString(msg);
                ((Alert) display.getCurrent()).setTimeout(time_out);
        } else {
                Alert alert = new Alert("Bluetooth");
                alert.setString(msg);
                alert.setTimeout(time_out);
                display.setCurrent(alert);
        }
}

public void pauseApp() {
}

// Aplicação pausada
public void destroyApp(boolean unconditional) {
}
```

```
        // Aplicação finalizada

}
```

Código do microcontrolador

```
#include <REGX52.H>          // Biblioteca com os mnemonicos dos microcontrolador ATMEL
                                da familia 8051

                        // Prototipos das funcoes
                    void temp_seg(unsigned char segundos);
                    void chama_elevador(void);
                    void motor(void);
                    void verifica_andar(void);

                        // Variaveis globais
                    unsigned char elevA, elevB;

                    void main(void)          // Funcao principal.
                    {

        IE = 0x90;               // Habilita a chave geral (EA) das interrupcoes e a interrupcao
                                    serial.
        SCON = 0x50;             // Modo 1 - 8 bits UART, assincr., full-duplex, freq. var., recep.
                                    hab.
```

```
        TMOD |= 0x20;          // Timer 1 - Gate 0, Modo 2 - 8 bits com recarga -
                               // Usado para gerar o Baud Rate.
                               // Configura o Timer 0 no modo 1 16bits
        TH1 = 0xFD;            // Valor a ser inserido no registrador TH1
                               // para a obtencao do Baud Rate de 9.600bps,
                               // com um cristal de 11,0592MHz.
        TR0 = 0;               // Desliga o Timer 0
        TF0 = 0;               // Zera o Flag do Timer 0

        TR1 = 1;               // Liga o Timer 1

        chama_elevador();
        }

    void chama_elevador(void)  // Funcao para verificar se alguma tecla foi pressionada,
{                              // tendo como resultado o acendimento do
                               led associado a
                               // respectiva tecla.

        while(1)               // Funcao em loop para varredura do teclado.
                               {
        switch(P0)             // Verifica o a tecla que foi pressionada
                               {
                case 0xFE:     // Tecla pressionada - P0_0
                               P2_0 = 1;       // Led desligado
                               P2_1 = 1;       // Led desligado
```

```
                P2_2 = 1;       // Led desligado
                P2_3 = 1;       // Led desligado
                P2_0 = 0;       // Led ligado
        elevA = 0;      // Variavel armazena o valor 0 para posterior comparacao
                        na funcao
                                // verifica_andar().
                break;
        case 0xFD:      // Tecla pressionada - P0_1
                P2_0 = 1;       // Led desligado
                P2_1 = 1;   // Led desligado
                P2_2 = 1;       // Led desligado
                P2_3 = 1;       // Led desligado
                P2_1 = 0;       // Led ligado
        elevA = 1;      // Variavel armazena o valor 1 para posterior comparacao
                        na funcao
                                // verifica_andar().
                break;
        case 0xFB:      // Tecla pressionada - P0_2
                P2_0 = 1;       // Led desligado
                P2_1 = 1;       // Led desligado
                P2_2 = 1;       // Led desligado
                P2_3 = 1;       // Led desligado
                P2_2 = 0;       // Led ligado
        elevA = 2;      // Variavel armazena o valor 2 para posterior comparacao
                        na funcao
                                // verifica_andar().
                break;
```

```
                case 0xF7:      // Tecla pressionada - P0_3
                        P2_0 = 1;       // Led desligado
                        P2_1 = 1;       // Led desligado
                        P2_2 = 1;       // Led desligado
                        P2_3 = 1;       // Led desligado
                          P2_3 = 0;       // Led ligado
        elevA = 3;      // Variavel armazena o valor 3 para posterior comparacao
                        na funcao
                                        // verifica_andar().
                                break;
                case 0xEF:      // Tecla pressionada - P0_4
                        P2_4 = 1;       // Led desligado
                        P2_5 = 1;       // Led desligado
                        P2_6 = 1;       // Led desligado
                        P2_7 = 1;       // Led desligado
                          P2_4 = 0;       // Led ligado
        elevB = 0;      // Variavel armazena o valor 0 para posterior comparacao
                        na funcao
                                        // verifica_andar().
                                break;
                case 0xDF:      // Tecla pressionada - P0_5
                        P2_4 = 1;       // Led desligado
                        P2_5 = 1;       // Led desligado
                        P2_6 = 1;       // Led desligado
                        P2_7 = 1;       // Led desligado
                          P2_5 = 0;       // Led ligado
```

```
elevB = 1;       // Variavel armazena o valor 1 para posterior comparacao
                    na funcao
                                        // verifica_andar().
                                    break;
                            case 0xBF:      // Tecla pressionada - P0_6
                                P2_4 = 1;       // Led desligado
                                P2_5 = 1;       // Led desligado
                                P2_6 = 1;       // Led desligado
                                P2_7 = 1;       // Led desligado
                                P2_6 = 0;       // Led ligado
elevB = 2;       // Variavel armazena o valor 2 para posterior comparacao
                    na funcao
                                        // verifica_andar().
                                    break;
                            case 0x7F:      // Tecla pressionada - P0_6
                                P2_4 = 1;       // Led desligado
                                P2_5 = 1;       // Led desligado
                                P2_6 = 1;       // Led desligado
                                P2_7 = 1;       // Led desligado
                                P2_7 = 0;       // Led ligado
elevB = 3;       // Variavel armazena o valor 3 para posterior comparacao
                    na funcao
                                        // verifica_andar().
                                    break;
                            }
                            }
                        }
```

```
void motor(void)        // Funcao que ira enviar os sinais para controle do motor.
                                {
        P3_4 = 1;       // Nivel alto 1 e  baixo 0 nos pinos 3.4 e 3.5 - rotacao do motor
                                no sentido horario
                                P3_5 = 0;
        temp_seg(10); // Chama a funcao para contagem do tempo que o motor devera
                                permanecer ligado

        P3_4 = 0;       // Nivel baixo 0 nos pinos 3.4 e 3.5 - parada do motor
                                P3_5 = 0;
        temp_seg(50); // Chama a funcao para contagem do tempo que o motor
permanecera parado

        P3_4 = 0;               // Nivel baixo 0 e alto 1 nos pinos 3.4 e 3.5 - rotacao do
                                moto o sentido reverso
                                P3_5 = 1;
        temp_seg(10); // Chama a funcao para contagem do tempo que o motor devera
                                permanecer ligado

        P3_4 = 0;       // Nivel baixo 0 nos pinos 3.4 e 3.5 - parada do motor
                                P3_5 = 0;
        temp_seg(10);  // Chama a funcao para contagem do tempo que o motor
                                permanecera parado
                                }

void verifica_andar(void)       // Funcao que ira verificar quais led`s estao ligados.
```

```
                                        // Comparacao com relacao ao pino menos
                                    significativo
                                        // dentre os pinos 2_0 e 2_3 e 2_4 e 2_5.
                                Ao pino menos significativo
                                        // ativar o led no pino 2_0, ou entao, no
                                    pino 2_4.
                                    {
                                unsigned char contador1;

if(elevA <= elevB)     // Compara os valores armazenados nas variaveis elevA e elevB.
                                    {
                            P2_0 = 1;          // Desliga os led`s ligados aos pinos P2_0 a P2_3
                                    P2_1 = 1;
                                    P2_2 = 1;
                                    P2_3 = 1;

                            switch(elevA) // Verifica o valor armazenado na variavel elevA.
                                    {
                                     case 0:
                                P2_0 = 0;        // Liga o led ligado ao pino P2_0
                            for(contador1 = 0; contador1<=10;contador1++)     // Laco para
                                    manter o led
                                        // que esta ligado ao pino
                                        // P2_0, piscando.
                                        {
                                        P2_0=~P2_0;
```

```
                temp_seg(5);
                }
        P2_0 = 0;       // Coloca o estado alto no pino P2_0, o que
            mantem o led ligado.
                break;
              case 1:
            P2_1 = 0;       // Liga o led ligado ao pino P2_1
                temp_seg(10);
            P2_1 = 1;       // Desliga o led ligado ao pino P2_1
                temp_seg(10);
            P2_0 = 0;       // Liga o led ligado ao pino P2_0
    for(contador1 = 0; contador1<=10;contador1++)    // Laco para
            manter o led

                // que esta ligado ao pino

                // P2_0, piscando
                {
                P2_0=~P2_0;
                temp_seg(5);
                }
        P2_0 = 0;       // Coloca o estado alto no pino P2_0, o que
            mantem o led ligado.
                break;
              case 2:
            P2_2 = 0;       // Liga o led ligado ao pino P2_2
                temp_seg(10);
```

```
P2_2 = 1;        // Desliga o led ligado ao pino P2_2
                    temp_seg(10);
P2_1 = 0;        // Liga o led ligado ao pino P2_1
                    temp_seg(10);
P2_1 = 1;        // Desliga o led ligado ao pino P2_1
                    temp_seg(10);
P2_0 = 0;        // Liga o led ligado ao pino P2_0
for(contador1 = 0; contador1<=10;contador1++)     // Laco para
    manter o led

                    // que esta ligado ao pino

                    // P2_0, piscando
                    {
                    P2_0=~P2_0;
                    temp_seg(5);
                    }
P2_0 = 0;        // Coloca o estado alto no pino P2_0, o que
    mantem o led ligado.
                    break;
                    case 3:
P2_3 = 0;        // Liga o led ligado ao pino P2_3
                    temp_seg(10);
P2_3 = 1;        // Desliga o led ligado ao pino P2_3
                    temp_seg(10);
P2_2 = 0;        // Liga o led ligado ao pino P2_2
                    temp_seg(10);
```

```
                P2_2 = 1;       // Desliga o led ligado ao pino P2_2
                        temp_seg(10);
                P2_1 = 0;       // Liga o led ligado ao pino P2_1
                        temp_seg(10);
                P2_1 = 1;       // Desliga o led ligado ao pino P2_1
                        temp_seg(10);
                P2_0 = 0;       // Liga o led ligado ao pino P2_0
            for(contador1 = 0; contador1<=10;contador1++)     // Laco para
                manter o led

                        // que esta ligado ao pino

                        // P2_0, piscando
                                {
                                P2_0=~P2_0;
                                temp_seg(5);
                                }
            P2_0 = 0;       // Coloca o estado alto no pino P2_0, o que
                mantem o led ligado.
                                break;
                        }

        elevA = 0;      // Armazena o valor 0 na variavel elevA.
                }
                else
                {
    P2_4 = 1;       // Desliga os led`s ligados aos pinos P2_4 a P2_7
```

```
P2_5 = 1;
P2_6 = 1;
P2_7 = 1;

switch(elevB) // Verifica o valor armazenado na variavel elevB.
{
 case 0:
P2_4 = 0;       // Liga o led ligado ao pino P2_4
for(contador1 = 0; contador1<=10;contador1++)     // Laco para
    manter o led

// que esta ligado ao pino

// P2_4, piscando
{
P2_4=~P2_4;
temp_seg(5);
}
P2_4 = 0;       // Coloca o estado alto no pino P2_4, o que
   mantem o led ligado.
break;
 case 1:
P2_5 = 0;       // Liga o led ligado ao pino P2_5
temp_seg(10);
P2_5 = 1;       // Desliga o led ligado ao pino P2_5
temp_seg(10);
P2_4 = 0;       // Liga o led ligado ao pino P2_4
```

```
for(contador1 = 0; contador1<=10;contador1++)     // Laco para
        manter o led

                    // que esta ligado ao pino

                    // P2_4, piscando
                        {
                    P2_4=~P2_4;
                    temp_seg(5);
                        }
    P2_4 = 0;       // Coloca o estado alto no pino P2_4, o que
      mantem o led ligado.
                        break;
                    case 2:
      P2_6 = 0;     // Liga o led ligado ao pino P2_6
                    temp_seg(10);
      P2_6 = 1;     // Desliga o led ligado ao pino P2_6
                    temp_seg(10);
      P2_5 = 0;     // Liga o led ligado ao pino P2_5
                    temp_seg(10);
      P2_5 = 1;     // Desliga o led ligado ao pino P2_5
                    temp_seg(10);
      P2_4 = 0;     // Liga o led ligado ao pino P2_4
for(contador1 = 0; contador1<=10;contador1++)     // Laco para
        manter o led

                    // que esta ligado ao pino
```

```
                    // P2_4, piscando
                    {
                    P2_4=~P2_4;
                    temp_seg(5);
                    }
P2_4 = 0;       // Coloca o estado alto no pino P2_4, o que
    mantem o led ligado.
                    break;
                case 3:
    P2_7 = 0;       // Liga o led ligado ao pino P2_7
                    temp_seg(10);
    P2_7 = 1;       // Desliga o led ligado ao pino P2_7
                    temp_seg(10);
    P2_6 = 0;       // Liga o led ligado ao pino P2_6
                    temp_seg(10);
    P2_6 = 1;       // Desliga o led ligado ao pino P2_6
                    temp_seg(10);
    P2_5 = 0;       // Liga o led ligado ao pino P2_5
                    temp_seg(10);
    P2_5 = 1;       // Desliga o led ligado ao pino P2_5
                    temp_seg(10);
    P2_4 = 0;       // Liga o led ligado ao pino P2_4
    for(contador1 = 0; contador1<=10;contador1++)
                    {
                    P2_4=~P2_4;
                    temp_seg(5);
```

```
                                        }
                P2_4 = 0;       // Coloca o estado alto no pino P2_4, o que
                    mantem o led ligado.
                                        break;
                                    }

            elevB = 0;      // Armazena o valor 0 na variavel elevB.
                                }
                                    }

        void int_serial(void) interrupt 4 // Funcao de interrupcao serial
                                    {

        char confirma[] = {"OK"}, i; // Variavei com mensagem OK e contador,
                                respectivamente.
int opcao;                  //Variavel que ira receber o dado do registrador SBUF

 opcao = SBUF;              // A variavel opcao recebera o dado armazenado no
                                registrador
                                    // SBUF - recebido serialmente atraves do cabo
                                RS232.
while(RI == 0);             // Espera que o flag de recepcao seja 1, ou seja, fim da
                                                    // recepcao.
    RI = 0;                 // Zera o flag, permitindo nova recepcao.

switch(opcao) // Verifica o caractere armazenado na variavel opcao e, caso sejam
                                alguns
```

```
// dos caracteres a, s, d, z, x e c, envia uma mensagem
  // de confirmacao de recebimento "OK" e chama as
funcoes motor() e
                    // verifica_andar().
                {
                 case 'a':
  for(i=0;confirma[i]!='\0';i++)
                      {
                SBUF=confirma[i];

                 while(TI==0);

                   TI=0;
                  }

                motor();

              verifica_andar();
                 break;

                 case 's':
  for(i=0;confirma[i]!='\0';i++)
                      {
                SBUF=confirma[i];

                 while(TI==0);
```

```
TI=0;
}

motor();

verifica_andar();
break;

case 'd':
for(i=0;confirma[i]!='\0';i++)
{
SBUF=confirma[i];

while(TI==0);

TI=0;
}

motor();

verifica_andar();
break;

case 'z':
for(i=0;confirma[i]!='\0';i++)
```

```
{
SBUF=confirma[i];

while(TI==0);

TI=0;
}

motor();

verifica_andar();
break;

case 'x':
for(i=0;confirma[i]!='\0';i++)
{
SBUF=confirma[i];

while(TI==0);

TI=0;
}

motor();

verifica_andar();
```

```
                    break;

                    case 'c':
                for(i=0;confirma[i]!='\0';i++)
                    {
                SBUF=confirma[i];

                    while(TI==0);

                      TI=0;
                    }

                motor();

                verifica_andar();
                    break;
                }
            }

void temp_seg(unsigned char segundos)       // Funcao de geracao de tempo - base 1s
                {
// A variavel segundos e a entrada do parametro de quantos segundos serao gerados

      unsigned int carga_inicial=(65535-49999);  // Variavel de carga inicial do Timer
    unsigned char vezes;                                    // Variavel multiplicativa
                        para geracao
```

```
                                                                    // de
1 segundo.

// Loop para contagem de SEGUNDOS x VEZES x 50.000us
while(segundos)
{
// Loop para contagem de VEZES x 50.000us
vezes=20;

while(vezes)
{
TL0 = carga_inicial;   // Carregamento da parte baixa do valor_inicial
de
                                                                    // contagem.
TH0 = carga_inicial>>8; // Carregamento alta do valor inicialde
contagem.

TR0 = 1;                                    // Ligamento do Timer 0.

while(!TF0);                    // Verificacao do fim da contagem a partir
do estouro
                                                                    // do flag, necessario
apenas quando nao usamos a
                                                                    // interrupcao do Timer.
TR0 = 0;                                    // Desliga o Timer 0.
TF0 = 0;                                    // Zera o flag.
```

```
                vezes--;                                    // Decrementa a variavel.
                                }


                                segundos--;
                                }
                        }
```