UNIVERSIDADE ESTADUAL DO SUDOESTE DA BAHIA
CURSO DE CIÊNCIAS DA COMPUTAÇÃO

CONTROLE E MONITORAMENTO RESIDENCIAL UTILIZANDO REDES GPRS

ANDERSON JARBAS DE JESUS SOUZA

VITÓRIA DA CONQUISTA - BA
2011

ANDERSON JARBAS DE JESUS SOUZA

CONTROLE E MONITORAMENTO RESIDENCIAL UTILIZANDO REDES GPRS

Monografia apresentada como requisito parcial para obtenção do título de Bacharel do curso de Ciências da Computação, Universidade Estadual do Sudoeste da Bahia (UESB).

ORIENTADOR: PROF. ADILSON DE LIMA PEREIRA.

VITÓRIA DA CONQUISTA - BA
2011

ANDERSON JARBAS DE JESUS SOUZA

CONTROLE E MONITORAMENTO RESIDENCIAL UTILIZANDO REDES GPRS

Aprovada em ____/____/____

VITÓRIA DA CONQUISTA, BAHIA.

BANCA EXAMINADORA/ COMISSÃO AVALIDADORA

Me. Adilson de Lima Pereira - Orientador

Universidade Estadual do Sudoeste da Bahia

Esp. Oliver Tompson Lessa – Co-Orientador

Faculdade Independente do Nordeste

Esp. Charles Miranda Fróes

Faculdade Independente do Nordeste

COORDENAÇÃO

Dr. Fábio Moura

Universidade Estadual do Sudoeste da Bahia

DEDICATÓRIA

Dedico este trabalho aos meus pais, Roque José e Maria Lourdes, meus alicerces de minha vida.

AGRADECIMENTOS

Concluir uma meta de minha vida, trás a sensação de um sentimento muito grande de reconhecimento e agradecimento a Deus, aos meus pais Roque José e Maria Lourdes, pela presença constante, ao meu irmão Jackson, aos meus primos, a todos meus familiares, ao meu orientador Adilson, a todos os professores da UESB, aos professores convidados para a banca examinadora e amigos Charles, Oliver, Flávio, Jackson, Nicássio Robéria, e a todos os colegas da UESB.

“Sábio é o ser humano que tem coragem de ir diante do espelho da sua alma para reconhecer seus erros e fracassos e utilizá-los para plantar as mais belas sementes no terreno de sua inteligência.”

Augusto Cury

RESUMO

Este trabalho apresenta o estudo e o método de desenvolvimento de um sistema de monitoramento e controle residencial. A unidade de processamento é composta por um microcontrolador que está conectado ao modem GSM (Sistema Global de Comunicação Móvel) sendo este, programado com algoritmo de decisão de acordo aos sensores e dispositivos instalados por toda a residência e comandos disponibilizados pelo modem. O sistema é capaz de monitorar e controlar dispositivos presentes na residência através do serviço GPRS (General Packet Radio Service). No trabalho é utilizada uma pesquisa experimental, sendo que na primeira etapa foram realizadas consultas a diversas fontes bibliográficas que tratam do tema escolhido para investigação. Os resultados obtidos do sistema de monitoramento e controle residencial foram satisfatórios com um bom desempenho e um custo acessível.

Palavras-chave: GPRS. Microcontrolador. Monitoramento e Controle Residencial.

ABSTRACT

This work presents the study, development, construction and tests of a system for monitoring and home control. The processing unit comprises a microcontroller that is connected to the GSM (Global System for Mobile Communications) modem, and this decision algorithm programmed according sensors and devices installed throughout the residence, and command decision-making provided by the modem. The system is able to independently monitor and control devices in the residence, via service GPRS (General Packet Radio Service). Was used an experimental research, and that in the first stage were held at various library resources that address the theme chosen for investigation. The results goals can be considered satisfactory, since the tool was tested with a good performance and an affordable cost.

Key words: GPRS, Microcontroller, Monitoring and Home Control.

LISTA DE FIGURAS

Figura 1 - As freqüências não são reutilizadas nas células adjacentes e para acrescentar mais usuários, podem ser utilizadas células menores. 19
Figura 2 - Arquitetura de uma rede GSM. 20
Figura 3 - A plataforma Java. 25
Figura 4 - Arquitetura JME. 26
Figura 5 - Ciclo e estados de um MIDlet. 27
Figura 6 - Composição do MIDP. 30
Figura 7 - Detalhamento de um circuito integrado. 32
Figura 8 - Tipos de encapsulamento. 33
Figura 9 - Circuito Impresso na Placa Fenolítica. 33
Figura 10 - Ambiente do esquemático do Eagle 5.7. 34
Figura 11 - Ambiente do *board* no Eagle (layout). 34
Figura 12 - Pinagem do ATmega 8515. 38
Figura 13 - Diagrama de blocos do Microcontrolador. 40
Figura 14 - Diagrama de bloco, gerador de clock. 41
Figura 15 - Modem GSM. 42
Figura 16 - Módulo SIM340D. 43
Figura 17 – Diagrama representativo do sistema. 49
Figura 18 – Diagrama de sequência do sistema. 51
Figura 19 – Socket é inicializado com uma porta pré-definida 7979. 51
Figura 20 - Threads para executar clientes em paralelo. 52
Figura 21 - Sensores e dispositivos de controle. 53
Figura 22 - Protótipo para a comunicação com o módulo. 54
Figura 23 - Conversor de sinais RS232/TTL para sinais do modulo. 54
Figura 24 - Placa dos sensores e dos relés. 56
Figura 25 - Maquete, visão do interior da residência. 56
Figura 26 - Maquete, posicionamento do hardware e do modem GSM. 57
Figura 27 - Codificação responsável por verificar o estado dos sensor. 58
Figura 28 - Codificação responsável para executar o controle solicitado. 58
Figura 29 - Imagem do ambiente de programação AVR Studio 4® 59
Figura 30 - Estratégia de um loop para receber mesagens. 60

Figura 31 - Rotina para a composição de mensagens..... 61
Figura 32 - A classe *Despachante*. 62
Figura 33 - NetBeans IDE. 63
Figura 34 - Teste 1 - Porta da frente aberta. 65
Figura 35 - Esquemático do circuito principal. 71
Figura 36 – Layout da parte superior da placa principal. 72
Figura 37 - Layout do circuito impresso da placa principal. 72
Figura 38 - Esquemático do circuito da placa de conversão dos sinais RS232. 72
Figura 39 - Layout da parte superior da placa de conversão dos sinais RS232. 73
Figura 40 - Layout da placa de conversão dos sinais RS232. 73
Figura 41 - Esquemático do circuito dos sensores e atuadores. 74
Figura 42 - Layout da parte superior da placa de sensores e atuadores..... 75
Figura 43 - Layout do circuito imresso da placa de sensores e atuadores. 75

LISTA DE QUADROS

Quadro 1: Características do microcontrolador ATmega 8515. 39
Quadro 2: Equações para calcular a taxa de transmissão em bits por segundo. 41
Quadro 3 - Características do módulo SIM340D. 44
Quadro 4 - Comandos AT. 47

LISTA DE ABREVIAÇÕES E SIGLAS

CI: Circuito Integrado

CISC: Complex Instruction Set Computer (Computador com um Conjunto Complexo de Instruções)

DSP: Processadores Digitais de Sinais

EPROM: Eletrically Programmable Read Only Memory (Memória programável exclusivamente para leitura)

EEPROM: Electrically Erasable Programmable Read-Only Memory (Memória programável apagável eletricamente exclusivamente para leitura)

EPP: Enhanced Parallel Port

GPRS: General Packet Radio Service (Serviço de Radio Geral por Pacotes)

GSM: Global System for Mobile Communications

I/O: Entrada e saída LED: Light Emitting Diode (Diodo Emissor de Luz)

MTSO: Mobile Telephne Switching Oflice (Estação De Comutação de Telefonia Móvel)

pF: pico Faraday

PROM: Programmable Read Only Memória (Memória programável de exclusivamente para leitura)

PIN: Personal Identificação Código

PUK: Personal Unlock Code

RAM: Random Access Memory (Memória de Acesso Aleátorio)

ROM:Read Only Memory (Memória Exclusivamente de Leitura)

RISC: Reduced Instruction Set Computer (computador com um conjunto reduzido de instruções)

RX: Recepção

RF:Rádio Freqüência

RC: Resistor capacitor

SIM: Subscriber Identity Module

SMS: Short Message System

SPP: Standard Parallel Port

TX: Transmissão

USB: Universal Serial Bus ()

uC: Microcontrolador

ÍNDICE

Capítulo 1.INTRODUÇÃO 15
1.1. Trabalhos relacionados 16
1.2. Problema de pesquisa 16
1.3. Justificativa 16
1.4. Objetivos 17
1.5. Metodologia 17
1.6. Estrutura da monografia 18
Capítulo 2.REDE DE DISPOSITIVOS MOVEIS-GSM 19
2.1. Rede GSM 19
Capítulo 3.IMPLEMENTAÇÃO PARA DISPOSITIVOS MÓVEIS 24
3.1. Por que Java! 24
3.2. A plataforma Java 24
3.3. Java ME 25
Capítulo 4.SISTEMAS EMBARCADOS 32
4.1. Circuito integrado 32
4.2. Encapsulamento 32
4.3. Circuito impresso 33
4.4. Arquitetura geral de um sistema embarcado 35
4.5. Comunicação 36
4.5.1. Comunicação serial 36
4.6. Microcontrolador 37
4.6.1. Microcontrolador atmel AVR atmega8515 37
4.7. Modem GSM 42
Capítulo 5.PROJEÇÃO DE DESENVOLVIMENTO DO TRABALHO 49
5.1. Protocolo e troca de mensagens 49
5.2. Servidor 51
5.3. Sistema controlador residencial 52
5.3.1. Hardware 53
5.3.2. Maquete 56
5.3.3. Firmware 57
5.3.4. Ferramentas de desenvolvimento 59
5.4. Aplicação do celular 59
5.5. Ferramentas de desenvolvimento 63
5.5.1. Netbeans IDE 643
5.5.2. Mobility pack 64

Capítulo 6.TESTES E ANÁLISES 65
6.1. Teste e análise I – monitoramento 65
6.2. Teste e análise II – controle 66
Capítulo 7.CONCLUSÃO 67
7.1. Sugestão para trabalhos futuros 68
REFERÊNCIAS 69
APÊNDICE A. 71

Capítulo 1. INTRODUÇÃO

Algo que está modificando a vida cotidiana de todos, é o progresso dos computadores, não só os do tipo computador pessoal, mas todos os *chips* de silício que estão embutidos nos eletro-eletrônicos em geral. Atualmente, observa-se que todos estes aparelhos trabalham independentes e isolados em suas funções. A revolução das redes domésticas e, por conseqüência, a automação residencial, estão baseadas no fato de permitir a comunicação entre estes dispositivos e controlá-los através de um gerenciador central. A automação permite controlar a residência remotamente, poupar tempo com tarefas repetitivas, economizar energia, dinheiro e aumentar o conforto e segurança (BOLZANI, 2004).

Atualmente, a insegurança em residências tem agravado esporadicamente, decorrente da grande banalização da população. O número de roubos a residências cariocas; por exemplo, durante os meses de junho, julho e agosto de 2009, subiu 12% em relação ao ano de 2008 durante o mesmo período, levando a sociedade a se precaver de inseguranças nas residências AUTRAN (2009).

Outro relato sobre a insegurança, é nas residências nobres de São Paulo e de acordo com Sidiconet (2009), na Rua João Della Manna, no bairro de Rolinópolis, que concentra casas de alto padrão na Zona Oeste de São Paulo, tem sido alvo de uma onda de assaltos, furtos e até seqüestros. Em comum, quase 90 % das residências dessa rua já foram roubados.

Levando em consideração as empresas já existentes, responsáveis pelo sistema de monitoramento remoto, como a TECNOGUARD, que utilizam o Sistema de Posicionamento Global (GPRS), os sinais da residência são recebidos via satélite na Central de Monitoramento Tecnoguard em que é monitorado em tempo real, 24 horas por dia, assim como o sistemas de monitoramento da Siemens que utilizam mesma tecnologia via GPRS ou por linha telefônica para fazer o monitoramento.

Este sistema utilizado na TECNOGUARD é bom para fazer o monitoramento, mas levando em conta por ser um sistema de grande porte é mais adequado para residências maiores. Para usuários que possuem residências menores e desejam de um sistema simples e funcional uma boa opção é utilizar o sistema proposto neste trabalho, que é capaz de fazer o monitoramento e controle remoto por mensagens que são transmitidas via serviço GPRS.

Os sistemas citados anteriormente são considerados como sistemas domóticos. Termo domótico é usado para designar residências que empregam serviços

automatizados. Tecnicamente falando, uma rede domótica pode ser representada por um conjunto de serviços interligados que realizam diversas funções de monitoramento e controle, podendo estar conectados entre si por meio de uma rede de comunicação interna e/ou externa (BOLZANI, 2004).

1.1.Trabalhos relacionados

Os seguintes trabalhos moderadamente relacionados foram: (a) Automação Residencial (OLIVEIRA, 2005) trabalho proposto que faz o uso da automatização utilizando computador pessoal, todo o sistema é monitorado a partir deste computador, faz o uso de porta paralela, para a comunicação dos dados entre circuito, e faz o uso, câmeras para fazer vídeo conferências, utiliza a programação assembler no circuito e utiliza um software *KeyDigital* com uma interface bastante simples somente para demonstração, este sistema desenvolvido é desktop não utiliza uma tecnologia remota. (b) Projeto de um Controlador de Alarme de Carro via SMS (ARTHUR, 2007), faz o uso do um módulo GSM da Motorola, para fazer o monitoramento de um sistema de alarme automotivo, utiliza o microcontrolador Atmel 8051. Em relação ao tema proposto é semelhante, porém utilizado em automóveis.

1.2. Problema de pesquisa

Decorrente da falta de segurança nas residências, principalmente quando se trata de residências nobres, a sociedade no geral tem a necessidade de um sistema que possa conceder segurança, confiança e comodidade. Diante dessa situação, como desenvolver um sistema, utilizando a tecnologia GSM (Global System for Mobile Communications), sistema global de comunicação móvel, por mensagens via serviço GPRS, para controle e monitoramento residencial, procurando dar eficiência e comodidade para o usuário?

1.3. Justificativa

O dicionário Aurélio define segurança como um lugar, espaço que está seguro; afastamento de todo perigo: viajar com segurança, viver em sua residência com segurança. A segurança na sociedade está sendo uma questão primordial, isto decorrente de ações dos agentes maliciosos ativos que tentam causar destruição, para se apoderar de algum bem material que não é de seu pertence. Decorrente agente maliciosos, pessoa que querem possuir algum bem material por meio de furto, as vítimas procuram se

precaver com medidas de segurança no ambiente que elas convivem, desde suas residências, ambiente de trabalho, ambiente de lazer, etc. (MEDINA, 2008).

Com o auxilio da tecnologia certas medidas de segurança tornam-se acessíveis. O monitoramento e controle em residências é uma medida de segurança viável, o termo "monitorar" acompanhar sistematicamente a residência, registrando algo indevido e o termo "controle" tomar uma decisão e verificar se a mesma está sendo bem sucedida.

Como qualquer novidade, a automação residencial, inicialmente, é percebida pelo cliente como um símbolo de status e modernidade. No momento seguinte, o conforto, a conveniência e a segurança por ela proporcionados, passam a ser decisivos. E, por fim, ela se tornará uma necessidade e um fator de economia (AURESIDE, 2000).

A partir do problema surgiu á idéia de criar um sistema de monitoramento e controle residencial utilizando a tecnologia GPRS na qual o usuário poderá monitorá-lo e controlá-lo a uma longa distância, tendo eficiência e comodidade para si próprio. Este sistema será composto de *hardware* e *software* com lógica capaz de fazer todo o acompanhamento residencial e informar ao usuário alguma irregularidade. Posteriormente o usuário irá responder ao sistema, referente a algum tipo de controle.

1.4. Objetivos

O objetivo geral deste trabalho científico será desenvolver um sistema utilizando GPRS para controle e monitoramento residencial através de telecomandos, procurando dar eficiência e comodidade para o usuário.

Os objetivos específicos são os seguintes:

- Solucionar o problema verificando se há a necessidade e adequação ao modelo proposto para projetar o sistema que se torne prático e acessível ao mercado;
- Conhecer alguns princípios da tecnologia GSM e GPRS;
- Observar a aplicação prática do uso da tecnologia nas áreas de Redes e Circuitos Digitais;
- Identificar, entre os princípios, os pontos a serem utilizados para desenvolver um sistema de acionamento por telecomandos.

1.5. Metodologia

Este trabalho se baseia em uma pesquisa aplicada, pois objetiva gerar conhecimentos para aplicação prática. Quanto à forma de estudo, trata-se de uma

pesquisa exploratória. Quanto ao objeto, este trabalho se enquadra na pesquisa experimental, já que o pesquisador é um agente passivo, que manipula uma ou mais variáveis independentes sob um controle adequado, objetivando observar e interpretar as reações e modificações acontecidas no objeto de pesquisa (PRESTES, 2002). O tipo de procedimento deste trabalho se vale do método dedutivo, uma vez que este se propõe a criar algo partindo do raciocínio geral para o particular.

1.6. Estrutura da monografia

A estrutura da monografia considera o primeiro capítulo como parte de contextualização do tema.

O segundo capítulo refere-se aos conceitos e informações levantadas sobre o tema em relação à tecnologia celular, abordado pela visão de outros autores e do próprio autor deste trabalho.

O terceiro capítulo descreve qual tecnologia de desenvolvimento é aconselhada para a criação das aplicações nos dispositivos moveis, destacando suas características, funcionalidades, facilidades e uso da tecnologia.

O quarto capítulo apresenta os principais componentes constituintes envolvidos neste estudo que se relaciona com a parte de sistemas embarcados, bem como seu princípio de funcionamento.

A seguir, o quinto capítulo apresenta a descrição, a projeção de desenvolvimento do trabalho teve como norteadores os requisitos identificados pelas especificações dos objetivos. Além disto, buscou-se, na medida do possível, manter simplicidade tanto na abordagem da solução quanto na implementação dos códigos.

O sexto capítulo foi realizado diversos testes presente na realidade sobre o monitoramento e controle residencial a partir disso retirou uma amostragem para fazer as devidas análises sobre a eficiência e qualidade do sistema.

No sétimo capítulo foram feitas as considerações finais e apontadas os pontos positivos e negativos do sistema de monitoramento e controle residencial, assim como perspectivas para trabalhos futuros.

Por último têm-se as referências bibliográficas, utilizadas para a criação desse trabalho.

Capítulo 2. REDE DE DISPOSITIVOS MOVEIS-GSM

Este capítulo descreve a rede de dispositivos moveis GSM, destacando suas características, funcionalidades e uso da tecnologia.

2.1. Rede GSM

A rede GSM foi criada pelos europeus a partir do sistema de tecnologia digital, PTTs (Push To Talk) estatais, que se juntaram e padronizaram um único sistema GSM (*Global System for Mobile Communications*). Esse esforço, em busca de um padrão, tornou o GSM o principal representante da segunda geração (SANTOS, 2008).

A rede é formada por interfaces abertas e padronizadas, seguindo sua principal intenção, montar uma arquitetura mais abrangente possível com diversos dispositivos de modens e celulares de diferentes fabricantes.

Assim como em outras tecnologias, o sistema composto de células geográficas que no geral, possuíam de 10 a 20 Km, foram aperfeiçoadas em células menores. Essas células utilizam um conjunto de freqüências não-utilizados por qualquer célula vizinha. Com isso, quanto menor a célula, maior a quantidade de freqüências por determinada localidade. Além de células menores, significam menor a necessidade de energia, que possibilita a existência de dispositivos transmissores e receptores menores e mais econômicos.

Na figura 1 está a representação das células em forma de hexágonos, que poderão ser divididas conforme a quantidade de usuários.

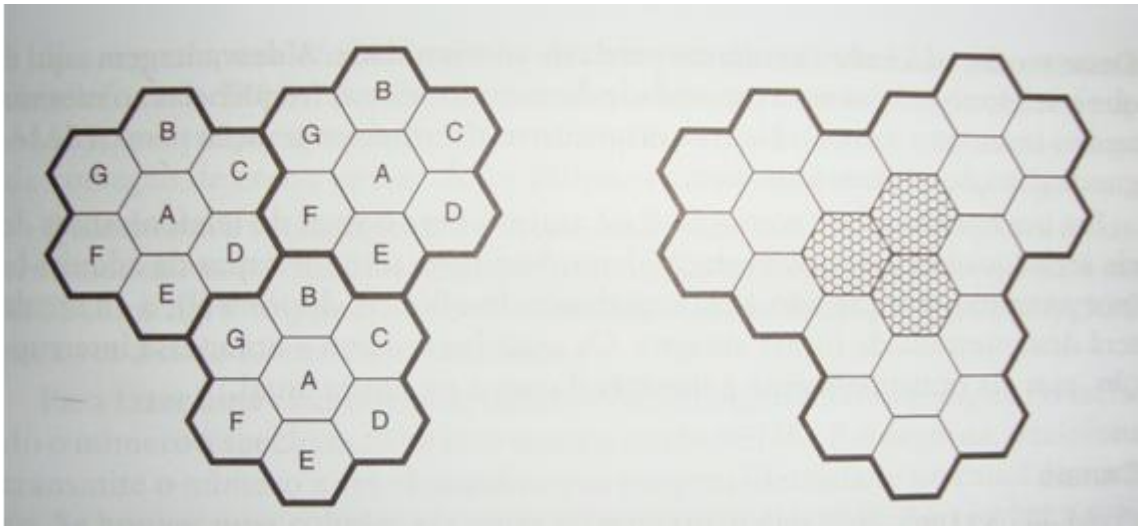

Figura 1 - As freqüências não são reutilizadas nas células adjacentes e para acrescentar mais usuários, podem ser utilizadas células menores.

Fonte: TANENBAUM (2003).

No centro de cada célula há uma estação-base que recebe as transmissões de todos os telefones presentes na célula. A estação base é composto por um sistema conectado a um transmissor/receptor conectados a uma antena, que transmite a uma MTSO (*Mobile Telephne Switching Oflice* – estação de comutação de telefonia móvel) usando uma rede de comutação de pacotes. (TANENBAUM, 2003).

Numa rede GSM, o terminal do utilizador chama-se estação móvel. Uma estação móvel é composta por uma cartão SIM (*Subscriber Identity Module*), permitindo identificar de maneira única este terminal móvel, maioria das vezes, é um telefone móvel (KIOSKEA, 2009).

Os terminais (aparelhos) são identificados por um número de identificação único de 15 números chamado **IMEI** (*International Mobile Equipment Identity*). Cada cartão SIM possui, igualmente, um número de identificação único (e secreto) chamado **IMSI** (*International Mobile Subscriber Identity*). Este código pode ser protegido com uma chave de 4 números chamada **código PIN.**

O cartão SIM permite assim identificar cada terminal, quando se faz a comunicação com uma estação básica. A comunicação entre uma estação móvel e a estação básica faz-se através de uma estação de rádio GSM.

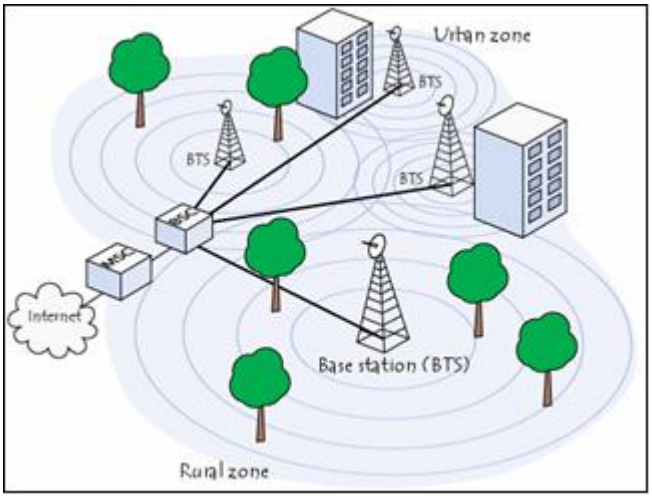

Figura 2 - Arquitetura de uma rede GSM.

Fonte: Kioskea (2009).

O conjunto das estações básicas de uma rede de dispositivos móveis ligadas a um controlador de estações (em inglês *Base Station Controller*, notado **BSC**), é encarregado de gerir a distribuição dos recursos. O conjunto constituído pelo controlador de estação e as estações básicas conectadas constitui o subsistema de rádio (em inglês **BSS** para *Base Station Subsystem*) (KIOSKEA, 2009).

Os controladores de estações estão, eles mesmos, ligados fisicamente ao centro de comutação do serviço móvel (em inglês MSC para Móvel Switching Center), gerido pelo operador telefônico, que o liga à rede telefônica pública e à Internet. O MSC pertence a um conjunto chamado sub-sistema rede (em inglês NSS para Network Estação Subsystem), encarregado de gerir as identidades dos utilizadores, a sua localização e o estabelecimento da comunicação com os outros assinantes (KIOSKEA, 2009).

O MSC está ligado geralmente a bases de dados que asseguram funções complementares:

- O registro dos assinantes locais (notado HLR para *Visitor Location Register* : trata-se de uma base de dados que contém informações (posição geográfica, informações administrativas, etc.) sobre os assinantes inscritos na zona do comutador MSC.
- O Registro dos assinantes visitantes (notado VLR para *Visitor Location Register*): trata-se de uma base de dados que contém informações sobre os outros utilizadores além dos assinantes locais. O VLR reenvia os dados sobre um novo utilizador a partir do HLR que corresponde à sua zona de assinatura. Os dados são conservados durante todo o tempo da sua presença na zona e suprimidos quando a deixa ou após um longo período de inatividade.
- O registro dos terminais (notado EIR para *Equipement Identity Register*): trata-se de uma base de dados que posiciona os terminais móveis.
- O Centro de autenticação (notado AUC para Autentication Center): trata-se de um elemento encarregado de verificar a identidade dos utilizadores.

A rede de dispositivos móveis, assim formada, foi concebida para suportar a mobilidade, à gestão do handover, ou seja, a passagem de uma célula à outra.

Por último, as redes GSM suportam deslocamento do dispositivo por lugares distintos, (em inglês roaming), ou seja, a passagem da rede de um operador a outro.

Cartão SIM.

O Cartão SIM contém as informações seguintes (SANTOS,2008):

- Número de telefone do assinante (MSISDN);
- Número de assinante internacional (<gras>IMSI</ital>, international mobile subscriber identity);
- Estado do mapa SIM;

- Código de serviço (operador);
- Chave de autenticação;
- Código PIN (Personal Identificação Código);
- Código PUK (Personal Unlock Code);
- Número de telefone do assinante (MSISDN);
- Número de assinante internacional;
- IMSI, internacional móvel subscriber identity) Estado do carta SIM;
- Código de serviço (operador);
- Chave de autenticação;

O padrão GPRS (*General Packet Radio Service*) é uma evolução da norma GSM, o que lhe vale, às vezes, a denominação GSM++ (ou GMS 2+). Dado que se trata de uma norma de telefonia de segunda geração que permite fazer a transição para a terceira geração (3G), fala-se geralmente de 2.5G para classificar o padrão GPRS (TANENBAUM, 2003).

O GPRS permite aumentar a arquitetura do padrão GSM, para autorizar a transferência de dados por pacotes, com débitos teóricos máximos de aproximadamente 171,2 kbit/s (na prática, até 114 kbit/s). Graças ao modo de transferência por pacotes, as transmissões de dados utilizam a rede apenas quando é necessário. O padrão GPRS permite, por conseguinte, faturar o utilizador ao volume trocado, em vez da duração de conexão, o que significa nomeadamente que pode ficar e ligar sem custo adicional (SANTOS, 2008).

Assim, o padrão GPRS utiliza a arquitetura da rede GSM para o transporte da voz, e propõe aceder a redes de dados (nomeadamente Internet) que utilizam o protocolo IP ou o protocolo X.25.

O GPRS permite novos usos que não permitia a norma GSM, geralmente classificados nas classes de serviços seguintes (TANENBAUM, 2003):

- Serviços ponto a ponto (PTP), ou seja, a capacidade de se ligar em modo cliente-servidor a uma máquina de uma rede IP.
- Serviços ponto a multipontos (PTMP), ou seja, a aptidão de enviar um pacote a um grupo de destinatários (Multicast).
- Serviços de mensagens curtas (SMS).

A Arquitetura da rede GPRS é composta pela integração do GPRS numa arquitetura GSM e necessita a adição de novos nós rede chamados GSN (*GPRS support nodes*) situados numa rede federativa (backbone) (TANENBAUM, 2003):

- O SGSN (*erving GPRS Support Node*, ou em português "Nó de apoio GPRS de serviço"), switch que permite gerir as coordenadas dos terminais da zona e realizar o interface de trânsito dos pacotes com a ponte estreita GGSN.
- O GGSN (*Gateway GPRS Support Node*, ou em português "Nó de apoio GPRS ponte estreita"), ponte estreita que se converte com as outras redes de dados (Internet). O GGSN é encarregado nomeadamente de fornecer um endereço IP aos terminais móveis durante toda a duração da conexão.

Concluindo a tecnologia GSM, o seu principal foco era oferecer telefonia digital móvel. Por isso, os protocolos de transmissão de dados se esforçam em adaptar o canal de voz para a transferência de bits de dados. O resultado é que o tráfego máximo de bits obtido pelo GSM foi inicialmente projetado para atender ao tráfego de bits gerado por conversações telefônicas. Para fazer um celular acessar a internet, por exemplo, seria necessária uma taxa bem maior de tráfego, e os projetistas do GSM inicialmente não se preparam para isso, depois começou a desenvolver um sistema que atendia a necessidade da transferência rápida de dados que foram aperfeiçoadas na tecnologia GPRS.

Capítulo 3. IMPLEMENTAÇÃO PARA DISPOSITIVOS MÓVEIS

Este capítulo descreve qual tecnologia de desenvolvimento é aconselhada para a criação das aplicações nos dispositivos móveis, destacando suas características, funcionalidades, facilidades e uso da tecnologia.

3.1. Por que Java!

Desde o seu surgimento em 1991 até os dias de hoje, Java evoluiu de tal maneira, que passou de uma linguagem de programação para uma plataforma que pode estar presente em, praticamente, todos os segmentos dos dispositivos computacionais existentes (MUCHOW, 2006).

Podem-se encontrar várias motivações que levaram à escolha de Java para o desenvolvimento do projeto, assunto deste relatório. Essas motivações vão desde aspectos técnicos da plataforma, linguagem de programação e ferramentas de desenvolvimento, até a aspectos de mercado, forma de distribuição, facilidade na obtenção de informações e popularização no meio dos desenvolvedores. Alguns números e estatísticas colaboram para estas conclusões:

Estimativas (JavaOne 2012):

• 95% dos celulares

• 99% dos computadores (desktop e portáteis)

• 100% dos PlayStation 3

Números (JavaOne 2010):

• 1,5 bilhões de celulares

• 99% dos celulares com suporte a Java

• 6 milhões de desenvolvedores Java.

3.2. A plataforma Java

Em virtude do seu crescimento e expansão, Java se tornou uma plataforma constituída por uma série de tecnologias. Cada uma delas é responsável por determinada parte do que compõe todo o ambiente de desenvolvimento e execução de software. Estas tecnologias foram, também, desenvolvidas para serem aplicadas a alguns segmentos específicos. Assim, a plataforma Java é composta da seguinte forma:

Java SE: Contém a base da plataforma com o ambiente de execução e bibliotecas mais comuns. Aplicada, normalmente, no desenvolvimento de aplicações para desktop.

Java EE: Aplicada ao desenvolvimento de aplicações corporativas seguindo padrões e modelo *server-side*.

JavaME: Desenvolvimento de aplicações para dispositivos móveis e para a maior parte de sistemas embarcados. Esse tipo de tecnologia, utilizado neste trabalho no intuito de desenvolver a aplicação para fazer a comunicação com o usuário.

Java Card: Orientada para o emprego em dispositivos embarcados com rígidas limitações de espaço para armazenamento e poder de processamento como o *Smart Card*.

JavaFX: Específica para o desenvolvimento de aplicações multimídia e se aplica a outros segmentos da plataforma (desktop, web e móveis). A figura 3 ilustra a arquitetura da plataforma Java de uma forma geral.

3.3.Java ME

Assim como em todos os outros segmentos da plataforma, na Java ME todas as classes são interpretadas por uma JVM (Maquina Virtual Java). Isto proporciona a condição de a aplicação não precisar ser recopilada para ser executada em outros aparelhos, pois as classes não sofrem influência de características e condições particulares de cada dispositivo.

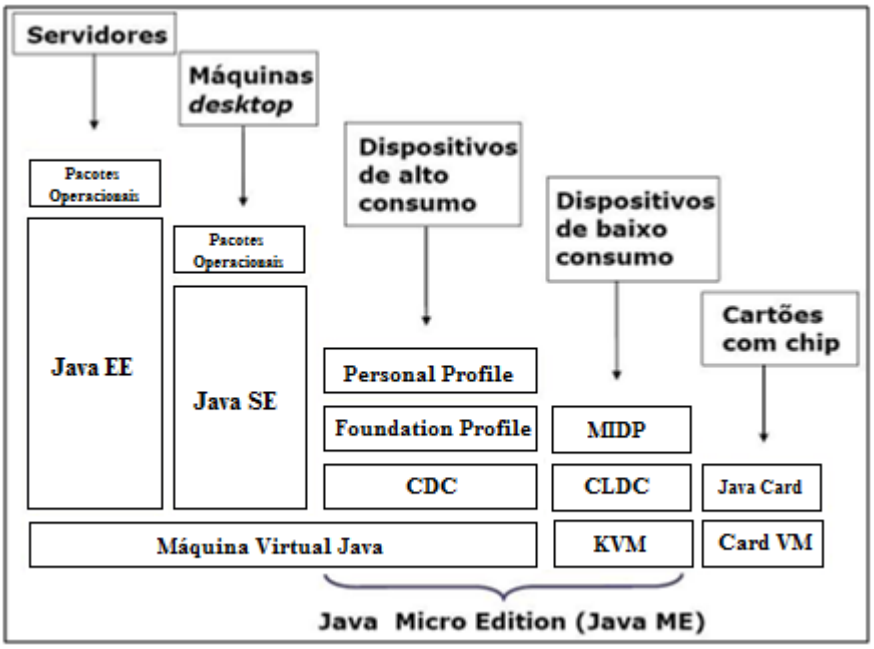

Figura 3 - A plataforma Java.

A arquitetura da JME (Java Micro Edition) é estruturada de modo que o seu núcleo é composto por *configurações* e por *perfis* (MUCHOW, 2006), figura 4.

Configurações

A configuração provê a estrutura básica de um ambiente Java. Isto é, a linguagem de programação, a máquina virtual Java, bibliotecas básicas da plataforma, algumas caractéristicas de segurança e a parte inicial das funcionalidades de comunicação. Existem dos tipos de configurção:

Connected Limited Device Configuration (CLDC) e Connected Device Configuration (CDC).

A CLDC é normalmente direcionada a dispositivos, teoricamente, mais limitados como um telefone celular ou um PDA com funcionalidades mais restritas. Isto exige que ela seja a mais leve das cofigurações. Os dispositivos devem apresentar pouco consumo de energia e ter, pelo menos, 128k de memória para a máquina virtual (VM).

CDC é uma configuração destinada a dispositivos com maiores capacidades tais como TVs, Set-top boxes ou PDAs e telefones celulares com maiores recursos de processamento armazenamento e conectividade. Na CDC, são encontradas todas as APIs que compõem a CLDC somadas a algumas outras APIs. Em outras palavras a CDC é um superconjunto da CLDC (MUCHOW, 2006).

Figura 4 - Arquitetura JME.

Aqui vale lembrar que, hoje em dia, muitos dos telefones celulares disponíveis no mercado, com preço relativamente acessível, já possuem funcionalidades e capacidades suficientes para utilizarem com facilidade a CDC como configuração preferencial.

Perfis

O perfil é o complemento da configuração contendo as APIS que vão possibilitar, de fato, a implementação de alguma aplicação. No perfil, é que certas características de um dispositivo serão exploradas no que diz respeito a interface, conectividade (rede), armazenamento, etc.

Mobile Information Device Profile (MIDP) é o nome dado ao perfil. Existe uma comunidade que especifica a arquitetura do MIDP e de várias APIs. Isto, para que se consiga padronização no desenvolvimento da plataforma buscando o máximo possível de suporte em todos os dispositivos com suporte à tecnologia.

Conforme a MIDP evolui, ganha funcionalidades e interação com vários aparelhos, novas versões são editadas e lançadas no mercado. Para cada versão, temos especificações diferentes de tela, entrada de dados, capacidade de armazenamento, forma de conectividade, etc. Isto oferece ao desenvolvedor uma maior facilidade na escolha da versão para o seu projeto de acordo com as capacidades do dispositivo em que se pretende executar a aplicação (MUCHOW, 2006).

A figura 5 ilustra a arquitetura da JME.

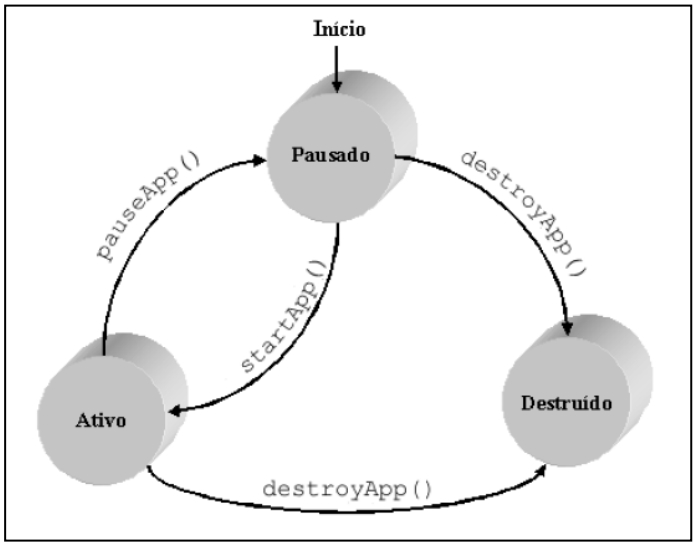

Figura 5 - Ciclo e estados de um MIDlet.

3.4.Midlet

Uma aplicação em Java para dispositivos móveis utilizando a plataforma JME recebe o nome de MIDlet. Seu nome denuncia obviamente sua relação com um perfil MIDP, pois é executada sobre a MIDP. Logo, podem dizer que uma aplicação MIDP é um MIDlet.

Na JME existe uma classe chamanda *MIDlet* que está alocada dentro do pacote *javax.microedition.midlet*. Assim, em termos mais técnicos de implementação e da linguagem, ao se escrever uma aplicação móvel é necessário estender a classe *MIDlet*.
Em todo dispositivo móvel, é instalado um AMS *(Application Management Software)* que interage e, em alguns aspectos, controla o comportamento do MIDlet. A interação pode ser provocada pelo próprio MIDlet, solicitando acesso ou informações do dispositivo e o controle é a ação do AMS sobre os estados do MIDlet.

Ciclo de vida do MIDlet

O AMS atua no MIDlet criando uma nova instância dele. Isto inicia o ciclo de vida do MIDlet que entra, no primeiro momento no estado **pausado**. O MIDlet possui um método chamado *startApp()* que, após ser chamado pelo AMS, coloca o MIDlet no estado **ativo**. Um outro estado em que o MIDlet pode estar por ação do EMS é o de **destruído**, onde sua atividade é encerrada. Isto ocorre com a chamada do método *destroyApp()* e o método *notifyDestroyed()* retorna a informação de sucesso da operação. Como vimos, um MIDlet pode se encontrar em três condições ou estados por influência, ou não, do AMS. A figura 5 ilustra os estados possíveis de um MIDlet e os métodos envolvidos em cada transição de estado.

Estrutura de um MIDlet

Abaixo se tem estrutura do código de um MIDlet com seus métodos principais. Para melhor visualização e entendimento, foi usado o código de um pequeno MIDlet como objeto de análise.

```
1 import javax.microedition.midlet.* ;
2 import javax.microedition.lcdui.*;
3
4 public class TesteMidlet extends MIDlet implements CommandListener {
5      Display display ;
6     Command exitCommand = new Command ( "Exit", Command . EXIT, 1 )
;
7     Alert testeAlert ;
8
9     public TesteMidlet( ) {
10          testeAlert = new Alert(
11                "Teste MIDlet ", "Teste, Teste OK! ",
12                null, AlertType.INFO
```

```
13          ) ;
14          testeAlert.setTimeout(Alert.FOREVER) ;
15          testeAlert.addCommand ( exitCommand ) ;
16          testeAlert.setCommandListener (this) ;
17    }
18
19    public void startApp( ) {
20          if(display == null) {
21                display = Display.getDisplay(this) ;
22          }
23          display.setCurrent(testeAlert) ;
24    }
25
26    public void pauseApp( ) { }
27
28    public void dest royApp (boolean unconditional) { }
29
30    public void commandAction (Command c , Displayable d ) {
31          i f (c == exitCommand) {
32                destroyApp (true) ;
33                notifyDestroyed( ) ;
34                / / Saída
35          }
36    }
37 }
16
```

Na linha de número 4, no código listado, é criada uma classe com o nome *TesteMidlet*, que é a nossa pequena aplicação exemplo. Como se pode obervar nessa linha, a classe criada estende uma classe *MIDlet*. Em outras palavras, a classe *TesteMidlet* é uma subclasse *MIDlet*. A classe *TesteMidlet* é criada de forma que também deverá implementar métodos da classe *CommandListener*, o que possibilitará o tratamento de alguns eventos.

Com uma rápida obervação podemos verificar qual será o comportamento da aplicação em questão. Assim que iniciada, oferecerá ao usuário uma opção (podendo aparecer em ítem de menu ou associada diretamente a algum botão de controle) que, quando disparada, mostrará na tela um pequeno painel de alerta com uma mensagem curta.

Além do atributo que proporciona a tela de alerta, temos a presença de um objeto do tipo *Display*. A partir dele é que podemos adicionar todos os outros elementos de comunicação com o usuário como formulários, campos de texto listas, etc. Em cada MIDlet só pode existir um único objeto *display*. Na figura 6 está demonstrada essa estrutura, os objetos, bem como a relação e dependência entre eles.

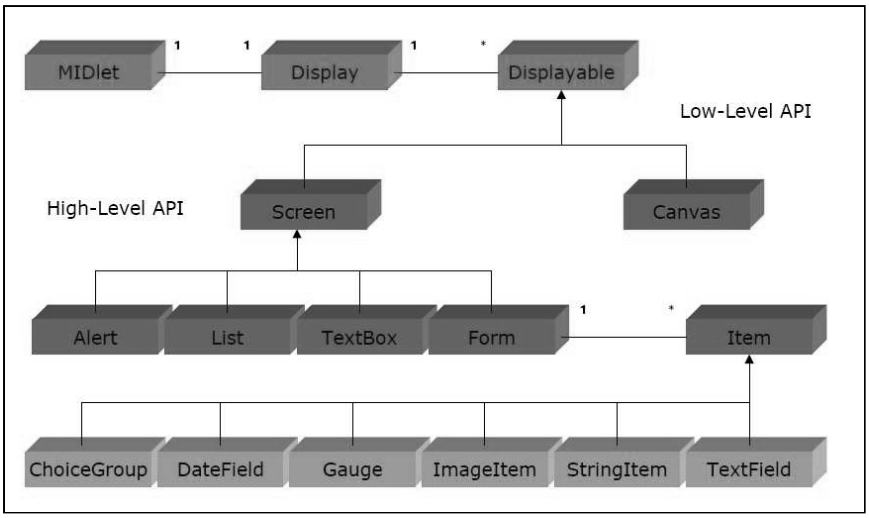

Figura 6 - Composição do MIDP.

O código, também, implementa uma opção de encerramento da aplicação. Na linha 30 o tratamento deste evento onde o método *destroyApp(true)* é invocado para acionar a mudança de estado da aplicação e, logo em seguida, o médoto *notifyDestroyed()* é chamado para que o AMS seja notificado do encerramento da aplicação. Para o tratamento do evento foi criado um obejto do tipo *Command* que recebeu o nome de *exitCommand* que é comparado a um outro objeto do mesmo tipo para verificar a ação do usuário que provocou o evento.

Em resumo, nesta pequena aplicação exemplo, a implementação mais básica de um MIDlet com os métodos necessários para o controle do estado da aplicação e um método responsável pelo tratamento de alguns eventos que dispara as ações apropriadas a cada um deles.

O que se pode notar, também, é que não existe, no código, a implementação de um método *main()* que seria comum e necessário em um outro programa escrito em Java. Para um MIDlet este método não é necessário e, se presente, não seria considerado para o funcionamento do programa. Para o MIDlet o método essencial que cumpre, superficialmente falando, o mesmo papel, é o *startApp()*.

O código apresentado é bem pequeno e não demonstra tudo o que se pode fazer em um MIDlet. Entretanto, nos permite visualizar a implementação do que é necessário para cumprir o ciclo de vida de uma aplicação e, também, dá uma idéia de como as ações e eventos são tratados. Em fim, o código permite visualizar de maneira geral a anatomia de um MIDlet.

A tecnologia JME possui aspectos técnicos que facilitam o desenvolvimento da aplicação, pois a linguagem de programação e a ferramentas de desenvolvimento possui facilidade na obtenção de informações, tornando acessível para a criação do sistema proposto.

Capítulo 4. SISTEMAS EMBARCADOS

Neste tópico serão abordados os principais componentes eletrônicos, assim como o seu principio de funcionamento. Aborda como acoplar dispositivo externo de I/O (entrada/saída) de sinais aos componentes, e os principais componentes e conceitos referentes a esse sistema.

Será apresentado os principais tipos de sistemas embarcados, suas diferenças, as formas de comunicação e suas áreas de aplicações.

4.1. Circuito integrado

Com a invenção dos transistores, as implementações de circuitos digitais tornaram-se mais complexas, devido à quantidade destes componentes, a partir disso teve a necessidade de criar chip's chamados de circuito integrado (CI), composto de milhares de pequenos transistores em um encapsulamento, conforme a figura 7.

A cada novo CI lançado, sua complexidade aumentava, possibilitando aplicações que antes eram impossíveis e inviáveis de serem implementadas, dando espaço para, logo depois, surgirem os primeiros microprocessadores, capazes de executar milhares de instruções por segundo.

Atualmente, a presença dos circuitos integrados está cada vez mais presente em atividades humanas, principalmente em equipamentos eletrônicos. (OLIVEIRA, 2006):

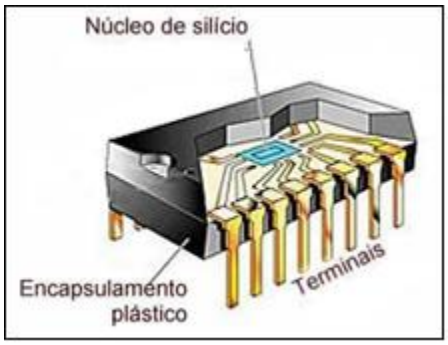

Figura 7 - Detalhamento de um circuito integrado.

Fonte: OLIVEIRA (2006).

4.2. Encapsulamento

Existem diversos formatos na indústria eletrônica denominados de *encapsulamentos*. Estes são projetados para otimizar o espaço e para facilitar a montagem ou em razão dos custos. Os diversos tipos são formados por uma pastilha de silício e por terminais de metais conforme a figura 8.

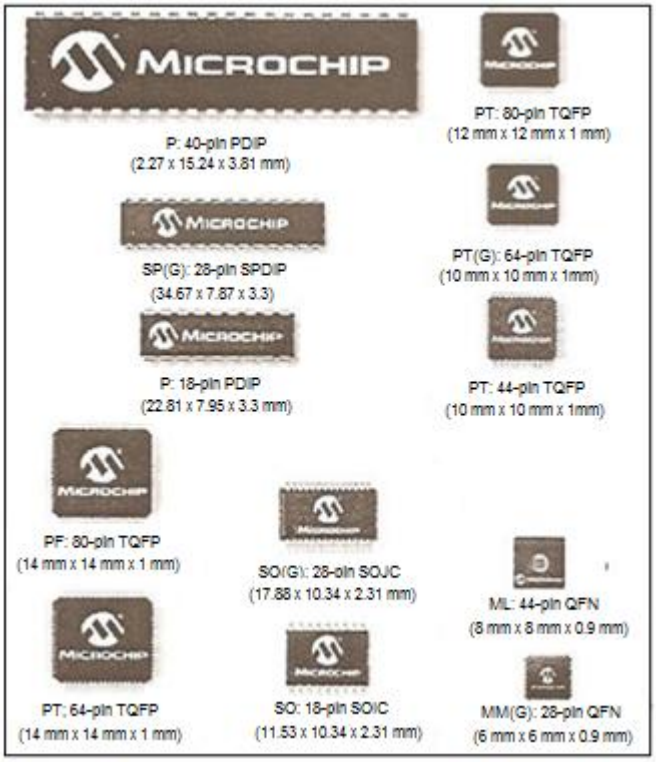

Figura 8 - Tipos de encapsulamento.

Fonte: Oliveira (2006).

4.3.Circuito impresso

O circuito impresso foi criado para substituir as antigas pontes de terminais onde se fixa os componentes eletrônicos, unindo os CIs de forma permanente e eficiente.

O circuito impresso pode ser confeccionado em uma placa de fenolite, fibra de vidro, fibra de poliéster, filme de poliéster, filmes específicos à base de diversos polímeros, etc., que possuem a superfície coberta em uma ou nas duas faces, por uma fina película de cobre, prata, ou ligas à base de ouro, níquel entre outras, nas quais são desenhadas as linhas condutoras que representam o circuito onde serão fixados os componentes eletrônicos através do processo de soldagem, (Figura 9).

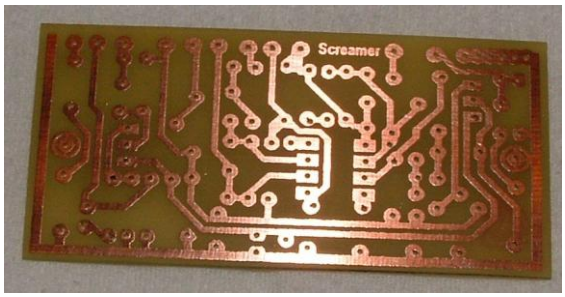

Figura 9 - Circuito Impresso na Placa Fenolítica.

Fonte: OLIVEIRA (2006).

Existem diversos *softwares* para a confecção dos circuitos, como exemplo o eagle 5.7 conforme a figura 10.

A confecção de circuitos primeiramente se faz um esquemático do circuito desejado, logo após é gerado o *layout* figura 11. O desenho do *layout* que será utilizado para a transposição em uma placa de circuito impresso e posteriormente fazer a realização do processo de soldagem dos componentes**.**

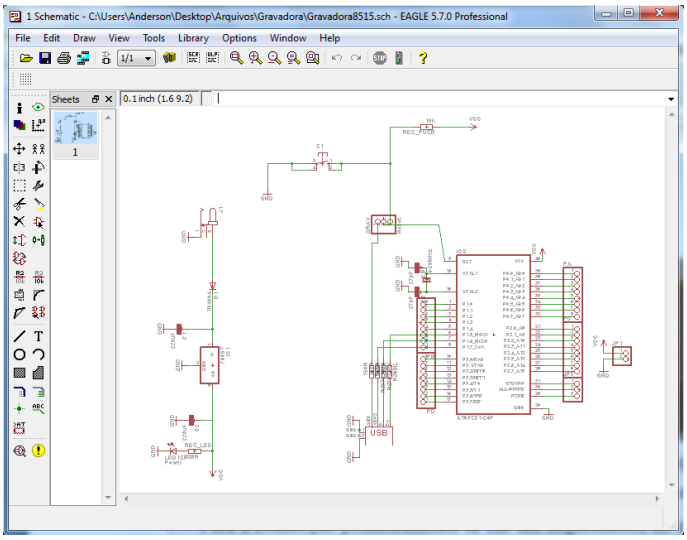

Figura 10 - Ambiente do esquemático do Eagle 5.7.

Fonte: EAGLE (2008) .

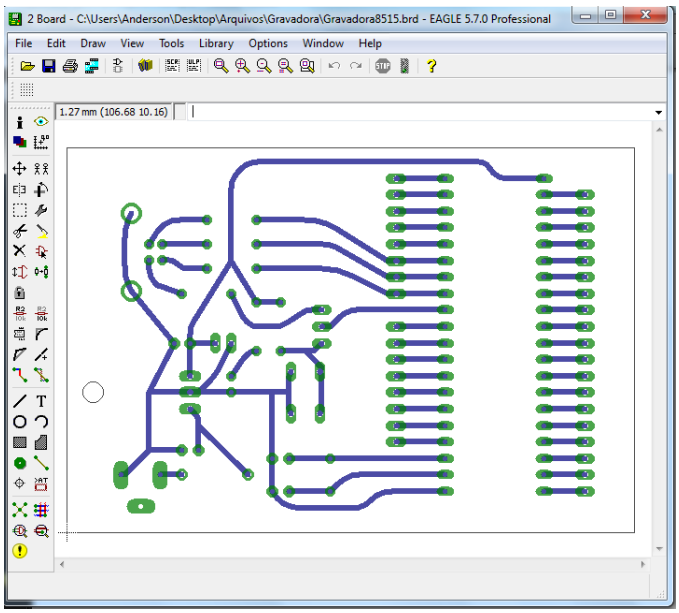

Figura 11 - Ambiente do *board* no Eagle (layout).

Fonte: EAGLE (2008) .

4.4. Arquitetura geral de um sistema embarcado

Os sistemas embarcados são circuitos integrados compostos por uma unidade de processamento de informações vindas de um software que são processados nesta unidade. Este software é designado *firmware*, semelhante ao de um computador pessoal, podendo ser escrito em diversas linguagens de programação, a partir de um algoritmo lógico (OLIVEIRA, 2006).

Segundo Oliveira (2006), o componente embarcado necessita de uma memória para armazenar o *software* projetado pelo desenvolvedor e, também, existem memórias que são utilizadas para armazenar dados temporários vindos de processamentos.

A memória disponível em um chip embarcado normalmente é limitada, com isso, obriga o desenvolvedor a trabalhar com dispositivos e software que consomem pouca memória. Normalmente a programação utilizada é o Assembly ou o C que aumenta o tempo e o custo de um software para o sistema embarcado (TAURION, 2005).

Existem diversos tipos de memórias como as seguintes.

Memória RAM (Random Access Memory – Memória de Acesso Aleatório): é volátil (ou seja, não mantém seus dados sem energia elétrica). Sempre que alimentada, inicia com dados inconsistentes. Possui o acesso aleatório a determinado endereço (OLIVEIRA, 2006).

Memória ROM (Read Only Memory – Memória exclusivamente de leitura): é uma memória somente de leitura e possui a programação definida pelo fabricante (OLIVEIRA, 2006).

Memória PROM (Programmable Read Only Memory) – Memória programável exclusivamente para leitura): é um tipo de memória ROM gravada um única vez pelo fabricante (OLIVEIRA, 2006).

Memória EPROM (Electrically Programmable Read Only Memory – Memória programável apagável eletricamente exclusivamente para leitura): é um tipo de memória PROM, cujo conteúdo pode ser apagado quando submetido a processos especiais, como luz ultravioleta. Pode reprogramada algumas vezes (OLIVEIRA, 2006).

Memória EEPROM ou E^2PROM (Eletricamente Erasable Programmable) Read-Only Memory – Memória programável apagável eletricamente exclusivamente para leitura): é do tipo ROM, que pode ser programada e também apagada por processos elétricos, por quantidades de vezes indefinida (OLIVEIRA, 2006).

Memória FLASH é uma memória semelhante a EEPROM mais o processo de escrita e leitura pode ser feita em velocidades altas, conseqüentemente seu custo é alto (OLIVEIRA, 2006).

Cada tipo de memória está relacionado para uma situação e função. A memória FLASH é utilizada para gravar o *firmware*, ou seja, memória de programa, armazena o *software* embarcado que o programador projetou.

A memória de dados de dados compreende a parte onde serão alocados (armazenados) os registros, as variáveis e todos os espaços reservados para o processamento temporário. Por ser uma memória temporária se por algum motivo faltar alimentação a mesma perderá toda informação.

A memória de armazenamento é utilizada em grandes aplicações de sistemas embarcados, por ser uma memória não volátil, ela é capaz de armazenar informações importantes ou algum parâmetro de processamento que serão utilizados posteriormente, essa memória é a EEPROM.

4.5.Comunicação

A comunicação está relacionada com o método de interação de um dispositivo eletrônico com outro, seja ele computador ou um hardware externo.

4.5.1. Comunicação serial

A comunicação serial é um tipo de transmissão de dados mais comum entre dispositivos embarcados e um computador pessoal. Essa transferência de dados é feita através do envio de bits serial ou enfileirados por uma única via. Essa transferência é feita pelo (Tx) para envio e (Rx) para recebimento (OLIVEIRA, 2006).

O controle da transmissão dos dados transmitidos é feito por campos para determinar a quantidade de bytes. Um tipo mais comum é a utilização de um bit de início mais sete ou oito bits de dados e por fim um ou dois bits de parada, ou seja, são necessários dez bits para enviar um caractere. Para fazer essa transmissão, necessita-se de alguns padrões entre o transmissor e o receptor. São eles:

Bit por segundo (Baund Rate): significa a velocidade com a qual os bits são transmitidos serialmente;

Bits de dados (Character Lenght): é o número de bits de um caractere, composto de oito bits;

Paridade (Parity): é um bit de configuração em 0 ou 1, para assegurar que o número total de bits 1 no campo de dados é par ou ímpar, como desejado.

Bits de parada (Start and Stop Bits): bit que determina início e fim de bloco, pode ser zero ou um.

Controle de fluxo (Flow Control): pode ser feito por hardware e software. A negociação por hardware usa linhas de controle para transmitir o sinal e receber as condições. O DSR (Data Set Ready), DTR (Data Terminal Ready), CTS (Clear to Send) e RTS (Request to Send) são padrões de negociação por hardware da comunicação serial. Há também o software para controle de fluxo dos sinais: XOR e XOFF, que habilita ou desabilita a transmissão (OLIVEIRA, 2006).

4.6.Microcontrolador

O microprocessador, também conhecido como Unidade Central de Processamento, é responsável por realizar todos os cálculos em um circuito, porém ele sozinho não faz nada. É preciso vários dispositivos periféricos, como memória de programa, memória de dados, barramentos entre outros (OLIVEIRA,2006).

O microcontrolador (MCU) é um componente que possui em um mesmo invólucro, um microprocessador, memória, conversores de sinal de digital para analógico, barramentos e outros periféricos. É praticamente um computador completo em um único encapsulamento (OLIVEIRA,2006).

De tamanho, custo e desempenho reduzido, os microcontroladores são ideais em aplicações que necessitam de menores dimensões, tempo e custo, como por exemplo, na automação industrial, residencial, brinquedos, e eletrodomésticos.

4.6.1. Microcontrolador atmel AVR atmega8515

O microcontrolador ATMEGA 8515 (ATMEL) um dos primeiros fabricado originalmente pela Atmel. Esta família começou a ser comercializada em 1997. Hoje, existe uma variedade de microcontroladores desta família, com várias possibilidades para implementação, desde simples projetos até aplicações bem complexas.

4.6.1.1.Principais características

No aspecto geral, a arquitetura AVR, “Alf(EGIL Bogen) e Vegard(Wollan)’s processador Risc”, apresenta características como arquitetura Haward, ou seja, memória de dados e memória de programa são separadas. Possui ainda freqüência de operação

em torno de 16MHz e, como esta freqüência não é dividida, como no 8051 e em alguns PICs, que tem sua freqüência dividida por 12 e 4, respectivamente, ele se torna mais rápido do que os demais microcontroladores. Além disso, a maioria das instruções é executada em um ciclo de clock e obtém em torno de 1MIP (milhões de instruções por segundo) por MHZ. Todas estas características comprovam a preocupação existente com desempenho pela Atmel ao utilizar-se este microcontrolador. Na figura 12 será abordado, com maiores detalhes, o microcontrolador da Atmel Atmega8515.

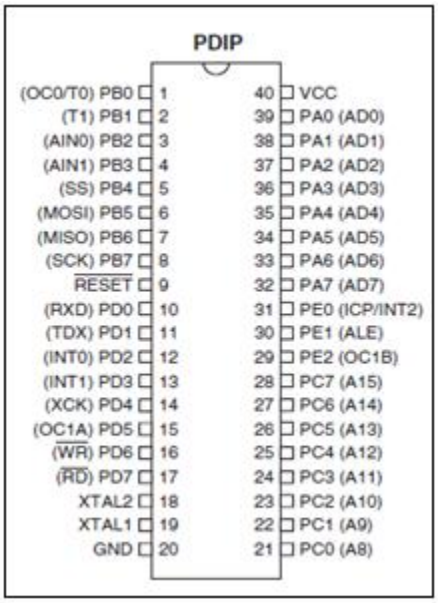

Figura 12 - Pinagem do ATmega 8515.

Fonte: ATMEL (2000).

Este microcontrolador possui várias características para um desenvolvimento de um bom trabalho. A partir das características foram apontados pra nosso o trabalho. Umas das principais características mostrados no quadro 1 (ATMEL, 2000):

Descrição
Microcontrolador de 8 bits baseado na Arquitetura ARV RISC.
130 instruções realizadas de registrador para registrador ou constante para registrador.
16 MIPS de throughput a 16 MHz.
Utiliza arquitetura de Harvard (memória e barramentos separados para dados e programa).
32 registradores de trabalho de propósitos gerais conectados diretamente com a ALU.
8Kb de memória In-System re Programmable Flash, utilizada para armazenamento de programas
512 bytes de EEPROM
512 bytes de SRAM
32 linhas de I/O de propósitos gerais

USART serial programável
Watchdog Timer programável com oscilador interno
Modo Power-down: salva o conteúdo dos registradores paralisa o oscilador, desabilitado todas as outras funções do chip até a próxima interrupção ou reset
Modo Standby: o oscilador está em execução enquanto os demais dispositivos estão em sleeping. Essa característica permite uma rápida inicialização com baixo consumo
Unidade de Interrupção possui registradores de controle no espaço de I/O, com um bit de habilitação de interrupção no registrador de status
Tensão de alimentação 4,5 – 5,5 V

Quadro 1: Características do microcontrolador ATmega 8515.

Fonte: ATMEL (2000).

O ATmega 8515 é suportado por um conjunto de sistemas completo de programas e ferramentas desenvolvimento, incluindo: compiladores C, montadores Macro, depurador de programas/simuladores, circuitos emuladores, e os kits de avaliação.

4.6.1.2. Universal synchronous and asynchronous receiver and transmitter (USART)

Os traços pontilhados na figura 13 separam o diagrama de blocos da USART do AVR em três partes principais (de cima para baixo): gerador de clock, transmissão e recebimento. Os registradores de controle são compartilhados entre todas as unidades. A lógica do gerador de clock consiste em utilizar a entrada de clock externo para construir uma lógica para sincronizar operações síncronas em modo escravo e gerar a taxa de transmissão. O pino de XCK (Transfer Clock) é utilizado para realizar transferência em modo síncrono. A transmissão utiliza: buffer de escrita, registrador de shift serial, gerador de paridade e um control logic para o envio de diferentes formatos de tamanho de frames. O buffer de escrita permite uma transferência contínua de dados sem nenhum atraso entre os frames. A recepção é o módulo mais complexo da USART devido ao clock e aos dados da unidade de recovery. Essa unidade, é utilizada na recepção de dados em modo assíncrono. Ademais, a recovery units, inclui a checagem de paridade, um control logic, um shift register e dois níveis de buffers de recebimento (UDR). A recepção suporta o mesmo formato de frame que a transmissão e pode detectar erros nos frames (ATMEL, 2000).

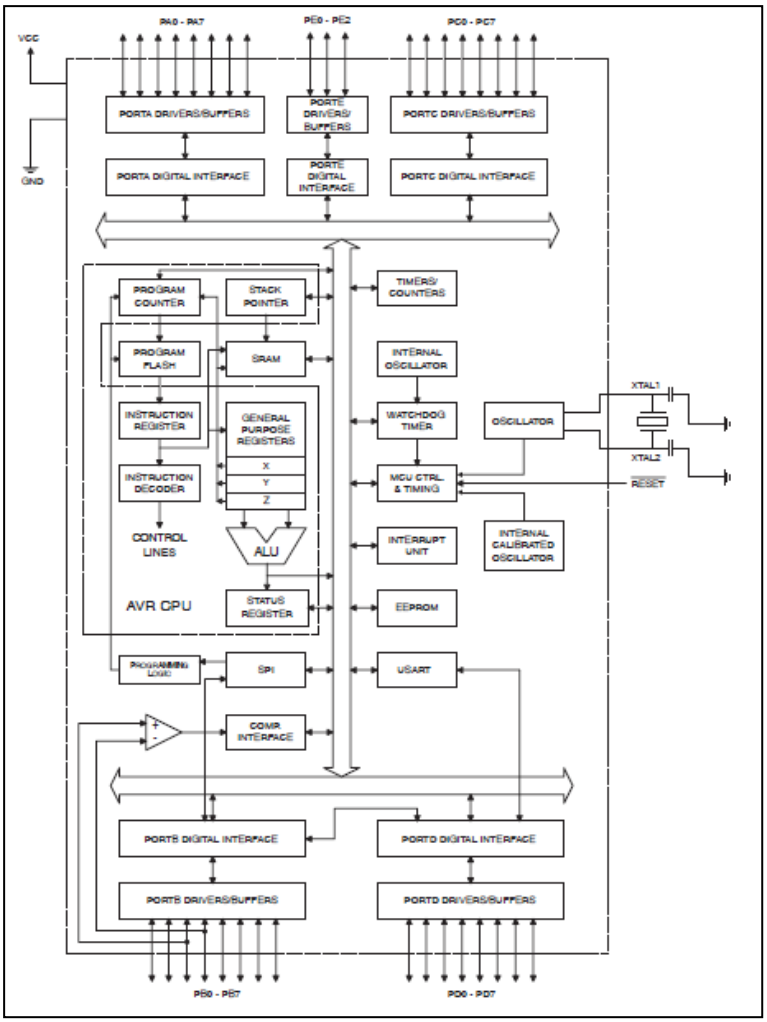

Figura 13 - Diagrama de blocos do Microcontrolador.

Fonte: ATMEL (2000).

O gerador de “clock” gera um sinal para a transmissão e o recebimento. A USART (Universal Synchronous and Asynchronous Receiver and Transmitter) suporta 4 modos para operação de clock: Normal assíncrono, Double Speed assíncrono, Mestre síncrono e Escravo síncrono. O bit UMSEL (USART Mode Select) do controle da USART e o bit C do registrador de status, UCSRC (USART Control and Status Register C), que selecionam entre o modo de transmissão síncrono e assíncrono do microcontrolador. O modo de operação Double Speed (somente em modo assíncrono) é controlado pelo U2X localizado no registrador de UCSRA . Quando a USART está operando em modo síncrono (UMSEL = 1), o Data Direction Register do pino XCK (DDR_XCK) controla se a fonte do clock será interna (modo mestre) ou externa (modo escravo). O pino XCK somente é /ativado quando se está utilizando o modo síncrono. A figura 14 mostra o diagrama de blocos do gerador de clock (ATMEL, 2000).

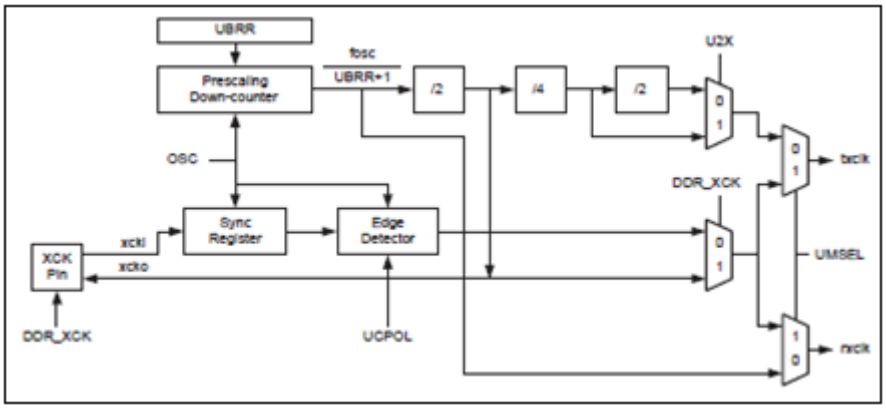

Figura 14 - Diagrama de bloco, gerador de clock.

Fonte: ATMEL (2000).

O registrador de taxa de transmissão (UBRR) e o down-counter são conectados para funcionar como um prescaler programável ou um gerador de taxa de transmissão. A transmissão divide a saída do clock do gerador de taxa de transmissão em 2, 8 ou 16 dependendo do modo utilizado. A tabela 1 contém as equações para calcular a taxa de transmissão em bits por segundo.

Operating Mode	Equation for Calculating Baud Rate[1]	Equation for Calculating UBRR Value
Asynchronous Normal mode (U2X = 0)	$BAUD = \frac{f_{OSC}}{16(UBRR+1)}$	$UBRR = \frac{f_{OSC}}{16BAUD} - 1$
Asynchronous Double Speed mode (U2X = 1)	$BAUD = \frac{f_{OSC}}{8(UBRR+1)}$	$UBRR = \frac{f_{OSC}}{8BAUD} - 1$
Synchronous Master mode	$BAUD = \frac{f_{OSC}}{2(UBRR+1)}$	$UBRR = \frac{f_{OSC}}{2BAUD} - 1$

Quadro 2: Equações para calcular a taxa de transmissão em bits por segundo.

Fonte: ATMEL (2000).

A transmissão de dados é iniciada escrevendo o dado a ser transmitido no sinal UART I/O Data Register (UDR). O dado é transferido do UDR para o Transmit shift register quando um novo caracter foi escrito no UDR depois que o stop bit do caracter anterior foi enviado. Neste caso, o "shift register" é carregado imediatamente. Ou, quando um novo caracter foi escrito no UDR antes que o stop bit do caracter anterior tenha sido enviado. Então, o "shift register" é carregado quando já tiver enviado o stop bit do caracter sendo transmitido (ATMEL, 2000).

Quando o receptor da UART detecta um start bit válido, ele começa o "polling" dos bits de dados recebidos. Além disso, este dispositivo também atualiza o UART Status Register (USR) quando um byte é recebido ou quando ocorre um erro de condição, como o registrador RX ser sobrescrito (dado foi recebido antes que o dado anterior tivesse sido lido do RX) ou um erro de framing (dado recebido em tamanho

diferente do esperado, geralmente devido à perda de um stop bit). O formato dos frames variam de 5 à 9 bits possuindo um start e um stop bit. Pode se utilizar bits de paridade e mais um stop bit. O UART Status Register possuí o seguintes bits:

• Bit 7 (RXC): UART Receive Complete - Setado quando um caracter é transmitido.

• Bit 6 (TXC): UART Transmit Complete - Setado quando o caracter no Transmit Shift register foi lido e nenhum novo dado foi escrito no UDR.

• Bit 5 (UDRE): UART Data Register Empty - Setado quando o caracter escrito no UDR é transferido para o Transmit Shift register.

• Bit 4 (FE): Framing Error - Setado ao detectar um erro de framing.

• Bit 3 (OR): Overrun - Setado ao detactar um erro de overrun.

• Bits 2 a 0 (Reserved Bits) - Bits reservados.

4.7.Modem GSM

O modem GSM da TATO Equipamentos Eletrônicos (figura 15) é um equipamento para transmissão de voz e dados, através da tecnologia celular GSM/GPRS, desenvolvido para proporcionar ao cliente, facilidade de instalação e integração. Este equipamento pode ser utilizado em aplicações diversas como telemetria, segurança, monitoração automotiva e residencial. Faz o uso da tecnologia GPRS, que é capaz de realizar comunicação de dados em velocidades de até 85.6Kbps para receber dados e de 42.8Kbps para transmitir além de conexões de voz, CSD, SMS e todas as funções convencionais de um aparelho celular, como agenda telefônica, transferência de chamadas (NOGUEIRA, 2010).

Figura 15 - Modem GSM.

Fonte: NOGUEIRA (2010).

A TATO Equipamentos Eletrônicos desenvolve este equipamento utilizando o módulo SIM340D (Figura 16). Este módulo é GSM/GPRS quadband com pilha TCP/IP embutida e é do tipo *board-to-board*, utilizado em aplicações onde a transmissão via tecnologia GSM/GPRS tenha disponível, capaz de fazer transmissão de voz, SMS, dados ou FAX, e possui baixo consumo de energia. Possui um tamanho, de 33mm x 33mm x 3mm, e é utilizado em aplicações industriais, sejam como M2M, telemetria ou qualquer outra forma de comunicação móvel. (TRILA21, 2009).

Figura 16 - Módulo SIM340D.

Fonte: TRILA21 (2009)

O módulo possui outras características conforme o quadro 2.

Características	Implementação
Alimentação	Voltagem: 3,4V – 4,5V.
Consumo de energia	Possui baixo consumo de energia - 2,5 mA.
Portas	Porta serial e uma porta de *debug* que ajuda no desenvolvimento de aplicativos, porem essas portas não podem trabalhar, ao mesmo tempo.
Freqüência	Quad-band GSM / GPRS trabalha em freqüências GSM / GPRS 900/1800 MHz e 850/1900MHz.
Conexão GPRS	GPRS multi-slot classe 8 (opcional). GPRS multi-slot classe 10 (padrão). GPRS classe estação móvel B.
Dados GPRS	GPRS transferência de dados GPRS downlink: max. 85.6 kbps.

	GPRS transferência de dados de uplink: max. 42,8 kbps. Esquema de codificação: CS-1, CS-2, CS-3 e CS-4. SIM340D suporta os protocolos PAP (*Password Authentication Protocol*), Senha de Protocolo de Autenticação, geralmente usado para conexões PPP. SIM340D integra o protocolo TCP / IP. Packet Switched Broadcast Channel Support Control (PBCCH).
SMS	MT, PB, CB, texto e modo PDU Armazenamento de SMS: o cartão SIM
Interface do SIM	Suporte o cartão SIM: 1.8V, 3V
Antena Externa	Antena de 50Ω.

Quadro 3 - Características do módulo SIM340D.

Fonte: SHANGHAI (2008)

Para o funcionamento do modem é necessário colocar um cartão SIM Card. Este é colocado em um soquete presente na placa do modem, que se encontra dentro da caixa metálica. Para isso, deve-se abrir o modem e inserir o cartão SIM no *socket* que está presente na parte interna do modem. No *socket* existe uma tampa metálica que é desconectada pressionando e empurrando para o lado, conforme descrito a mesma. Depois de inserido fecha-se a tampa e conecta-se a antena disponível no modem e faz-se a alimentação com a fonte disponível, na qual possui a entrada AC 90 – 240V e saída DC 9V com corrente de 1 ampare (SHANGHAI, 2004).

Para o controle e configuração do módulo é necessária a ligação do cabo serial padrão RS232 ao conector DB9, e conectar ao computador ou um equipamento que controlará o modem em uma porta serial padrão RS232.

Para controlar o modem pelo computador é através de um terminal ou equipamento, deve-se possuir a seguinte configuração na porta serial: 1 *start bit*, 1 *stop bit*, 8 *data bits*, *no parity* e a velocidade de transmissão da porta pode ser configurada entre a 300 *bauds* a 56.7600 *baunds*. A partir disso, a configuração do modem para controlar chamadas de voz, CSD e SMS, o terminal se comunica com o modem através de comandos AT (SHANGHAI, 2004).

O modem reconhece os comandos AT e retorna a resposta de acordo com o comando transferido ao modem. Caso o comando de retorno seja ERROR, é possível que o mesmo tenha sido transferido de forma incorreta que a função não seja reconhecida pelo modem ou, ainda, o comando prevê uma condição específica para esta resposta.

Existem diversos comando AT de configuração e controle do modem. Algum deles então descritos no quadro 4.

Comando AT	Resposta / Ação	Comentário
AT	OK ERROR	
AT+IPR=X	OK	Seta a velocidade de comunicação da porta serial. X: 0(Ajusta a configuração em *autobanding*) 300 1200 2400 4800 9600 19200 28800 38400 57600 115200
AT+CPIN?	+CPIN: SIM PIN +CPIN: READY +CPIN: ERROR	Status do PIN code: - caso a resposta seja SIM PIN é necessário entrar com o PIN. - caso a reposta seja PIN READY o modem já está operacional. - caso a reposta seja ERROR é preciso inserir o PIN code.
AT+CPIN=”xxxx”	OK ERROR	Insere o PIN code: - caso a resposta seja OK o PIN code foi inserido com sucesso. - caso a resposta seja ERROR o PIN code foi inserido de forma incorreta, já estava inserido ou ainda o SIM CARD está inserido de forma incorreta.
AT+CSQ	+CSQ: 000, 099 +CSQ: 001, 099 +CSQ: 010, 099 +CSQ: 020, 099 +CSQ: 030, 099 +CSQ: 031, 099	Verifica a qualidade do sinal: o valor XXX indica a intensidade do sinal GSM: - 000 indica ausência completa de sinal; - 001 equivale a uma barra de sinal (como indicado em um telefone celular); - 010 equivale a duas barras; - 020 equivale a três barras ; - 030 equivale a quatro barras; - 031 equivale a cinco barras de sinal, sendo este o nível máximo de sinalização medido;

		Se o sinal for dado como 000, verifique se a antena está conectada, se o SIM CARD está colocado corretamente e se o PIN NUMBER foi inserido.
AT+CLCK=”SC”,0,”xxxx”	OK	Desbloqueio do PIN: dessa forma não é necessário inserir o PIN sempre que o modem for religado. O xxxx deve ser o PIN code.
AT+CBAND?	+CBAND: 3 +CBAND: 4	Verifica a freqüência em que o modem está configurado: - 3 corresponde ao sistema PCS, 1900Mhz. - 4 corresponde a 900/1800Mhz. Para o Brasil é necessário configurar em 900/1800, opção 4. Se a freqüência não estiver configurada corretamente, o nível de sinal é dado como 000.
AT+CBAND=X	OK	Seta a freqüência de operação do modem: - X deve ser 3 para operação em 1900Mhz. - X deve ser 4 para operação em 900/1800Mhz. Após esse comando o Modem deve ser desligado e religado para entrar em operação normal.
AT+CMGF = 1	OK	Configurar SMS para envio no modo texto.
AT+CMGS= “7588366712”	>	Fornecer o número do telefone do celular que deseja enviar a SMS, e aguarda para digitar a mensagem desejada até 160 caracteres e finalizar a mensagem com CRTL Z (pressionando a tecla “CRTL” mais a tecla “Z”) ou o caractere 26 em decimal ou 1A em hexadecimal
AT+CMGL=X	OK	Seta a mensagem: X- “ALL” todos o SMS recebidas são listada. X– “1”, caso queira listar uma em específico, por exemplo, “1”.
AT+CMGD=X	OK	Seta a mensagem: x- “ALL” todos o SMS recebidas são apagados. x – “1”, caso queira apagar uma em específico, por exemplo, “1”.
AT+CIPHEAD=X	OK	X : 0 – não adiciona na ao cabeçalho o IP. 1 – adiciona um campo de informações antes dos dados de um

		servidor remoto automaticamente ao cabeçalho, o formato é + IPD (comprimento de dados), + IPD é sinal.
AT&W	OK	Grava o parâmetro atual no modem definido pelo usuário.
AT+CGATT=X	OK	ligar / desligar o serviço gprs. X: 0 – desliga, 1 – liga.
AT+CGDCONT=1,“IP”,“APN”	OK	configura conexão gprs - APN deve ser substituído pelo provedor de serviços (em geral a própria operadora). APN: tim.br
AT+CSTT="APN","X","Y”	OK	set APM, nome do usuário, senha. APN: tim.br X - nome do usuário: tim Y - senha: TIM
AT+CIICR	OK	estabelece uma conexão sem fio com GPRS ou CSD.
AT+CIFSR	xxx.xxx.xxx.xxx	Tem acesso ao endereço do IP local.
AT+CDNSORIP=X	OK	Conectar através do nome de IP ou pelo nome do domínio do servidor. X: 0 – endereço de IP. 1 – nome do domínio do servidor.
AT+CIPSTART = "X","IP","PORTA"	OK *CONNECT OK **CONNECT FAIL	Inicia a conexão TCP ou UDP X : TCP / UDP. IP: 189.3.47.201 PORTA: 7979 *conexão estabelecida com o servidor. ** falha na conexão com o servidor.
AT+CIPSEND=X	> *SEND OK	Envia dados através da conxão TCP ou UDP. X – número de caracteres a serem enviados, depois de escritos seram enviados automaticamente. X – nulo aguarda o comando CTRL+Z > agarda para escrever os dados, e por fim escreve CTRL+Z para enviar os dados. * dados enviados com sucesso.
AT+CIPCLOSE	OK	Fecha a conexão TCP ou UDP.
AT+CIPSHUT	SHUT OK	Desconecta a conexão sem fio.

Quadro 4 - Comandos AT.

Fonte: SHANGHAI (2008)

Estes comandos AT descrito acima são destinados para algumas situações, outras se necessitam de outros comandos disponíveis para download na documentação da SIMCOM.

O propósito do estudo de sistemas embarcados, em específico o microcontrolador atmega 8515 e o modem GSM, compuseram conhecimento para o desenvolvimento do cliente controlador, que é responsável pela integração da residência com o servidor. O controlador, em seu hardware possui um modem GSM, um microcontrolador, sensores e atuadores que compõem o dispositivo responsável pelo monitoramento e controle residencial. O software desenvolvido em C para o microcontrolador é responsável pela integração com modem GSM e monitoramento dos sensores e controle dos atuadores.

Capítulo 5. PROJEÇÃO DE DESENVOLVIMENTO DO TRABALHO

A projeção de desenvolvimento do trabalho teve, como norteadores, os requisitos identificados pelas especificações dos objetivos. Além disto, buscou-se, na medida do possível, manter simplicidade tanto na abordagem da solução quanto na implementação dos códigos. A figura 17 mostra um diagrama representativo do cenário encontrado de modo a visualizar a necessidade e, ao mesmo tempo, servir como referência para a modelagem do que se esperava como solução.

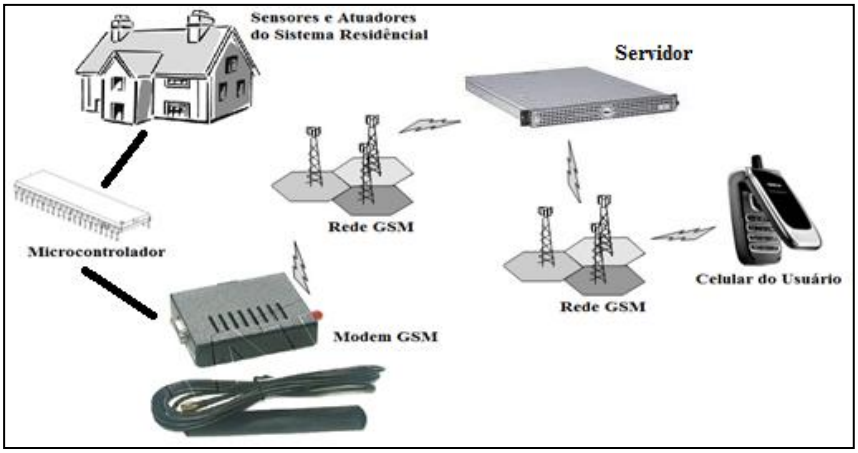

Figura 17 – Diagrama representativo do sistema.

A solução deveria considerar, entre outras coisas, de que forma ocorreria a comunicação entre a aplicação instalada no dispositivo móvel, com a aplicação do servidor, e por sua vez, com o *firmware* gravado no microcontrolador.

O caminho a ser seguido pelas mensagens trocadas seria: sair do telefone celular e entrar na rede GSM (GPRS) da operadora de telefonia, já encapsulada em TCP/IP. Em seguida, entrar no segmento de rede IP da mesma operadora para ser roteada para o seu destino (o modem GSM), seguido do microcontrolador, que está conectado aos sensores e atuadores. Toda essa comunicação é feita com o protocolo TCP (comunicação com conexão e com confirmação de mensagens), ao contrário de outras conexões como a UDP, que o processo de envio de mensagem não possui confirmação recebimento da mensagem quando enviada.

5.1. Protocolo e troca de mensagens

Para viabilizar o monitoramento e controle da forma desejado foi necessário estabelecer um protocolo de comunicação entre as aplicações. O protocolo, aqui referido, trata exatamente da maneira de como cada aplicação toma conhecimento sobre o que acontece do lado oposto.

Como em todo o restante do projeto, o objetivo, neste ponto também, foi buscar uma solução simples. Assim, ficou estabelecido que cada mensagem é composta por uma cadeia de números “zero” e “um”.

Como neste projeto optou-se por monitorar 5 sensores e 3 atuadores, cada mensagem trocada é composta por duas partes de três números cada. A parte inicial representa o estado ou ação desejada para os atuadores, enquanto a segunda parte, serve para indicar o estado dos sensores utilizados. Assim, um exemplo de mensagem trocada poderia ser:

SINAIS > 01001101

Como se pode ver no exemplo acima, cada mensagem trocada contém também um prefixo que é “SINAIS>”. A utilização deste prefixo foi inserida com a finalidade de facilitar a identificação do tipo de mensagem está sendo trocada. Quando o cliente conecta ao servidor, necessita sincroniza, isto é feito através de informações entre as duas aplicações, (cliente – servidor), utiliza o prefixo de identificação: “GETSINAIS>”.

Este meio para identificação de mensagens facilita ainda uma provável expansão dos tipos de mensagens possíveis a serem enviadas e recebidas, mantendo a facilidade do tratamento delas.

O processo de desenvolvimento desta projeção foi dividido em três etapas, a primeira é a implementação do servidor; a segunda, o processo de montagem do sistema controlador e configuração microcontrolador, conexão do modem via rede GSM com o servidor e, a terceira, implementação do dispositivo celular, que está também conectado ao servidor via rede GSM para atribuir a interface ao usuário. A figura 18, representa no formato de diagrama de seqüência, como é feita a conexão, entre cliente móvel, servidor e cliente controlador, e o processo de troca de mensagem.

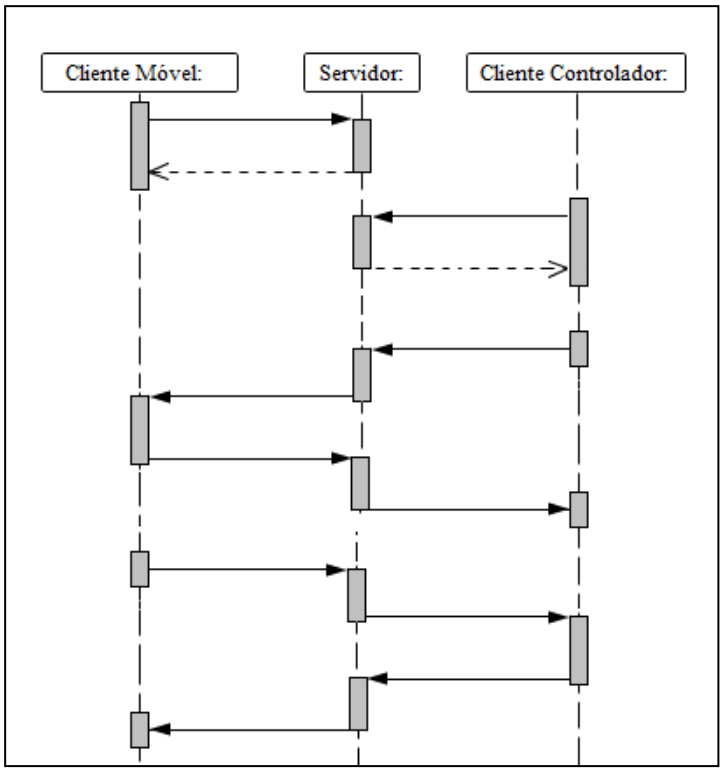

Figura 18 – Diagrama de sequência do sistema.

5.2.Servidor.

O servidor tem o propósito exclusivo de aguardar por, no mínimo, duas solicitações de conexão na porta escolhida para o projeto, uma o cliente controlador, e o segundo o cliente o celular.

O servidor desenvolvido em linguagem Java é composto de um *socket* com um IP público, em que é possível acessar por meio da rede da operadora de celular, o mesmo, é inicializado com uma porta pré-definida de número 7979 conforme escolhido, figura 19. Seguido da inicialização, o mesmo, fica aguardando o cliente se conectar, para chamar uma *thread* que executa paralelamente para cada cliente, figura 20. O servidor encarrega de receber e transferir as mensagens de um cliente aos outros, atribuindo um processo de comunicação entre clientes.

```
int porta=7979;
        try {
            socketServidor = new ServerSocket(porta);
        } catch (IOException ex) {
            Logger.getLogger(Main.class.getName()).log(Level.SEVERE,
null, ex);
        }
```

Figura 19 – Socket é inicializado com uma porta pré-definida 7979.

```
while (true) {
            try {
                socketCliente = socketServidor.accept();
                for (int i = 0; i <= 3; i++) {
                    if (clientes[i] == null) {
                        (clientes[i]= new ThreadCliente(socketCliente,
clientes)).start();
                        break;
                    }
                }
            } catch (IOException ex) {
              Logger.getLogger(Main.class.getName()).log(Level.SEVERE,
null, ex);
            }
```

Figura 20 - Threads para executar clientes em paralelo.

5.3.Sistema controlador residencial

O desenvolvimento desta etapa descreve como será feito a montagem do sistema controlador residencial, começando pela criação do sistema controlador, fazendo o estudo e montagem do hardware até a confecção da maquete para demonstração. Depois teve o processo de codificação na linguagem C, suportada pelo microcontrolador para fazer a configuração do modem, e execução do monitoramento e controle.

O microcontrolador e o modem estão conectados entre si, via porta serial, sendo necessária a montagem de um cabo padrão RS-232. Os dois dispositivos devem operar com a mesma taxa de transmissão (*baud rate*) para haver uma sincronia correta. A pré-configuração do modem e a conexão com o servidor é realizada através de comandos AT, que é o comando responsável por configurar o modem e montar a interface de acesso à rede GSM no formato do protocolo TCP. Todos os comandos AT necessários para esta etapa são encontrados na lista fornecida pelo fabricante do modem (TRILA21, 2009).

Por outro lado, o microcontrolador está interagindo com a residência através dos sensores e atuadores. Os sensores ficam conectados através dos pinos da porta PORTC e os atuadores através da porta PORTA. A PORTC é composta de 8 bits (pinos), 5 destes bits estão em uso e os outros 3 estão disponíveis para outros sensores que o

usuário necessitar futuramente. A PORTA, também de 8 bits, 3 estão conectados os atuadores e as outras estão disponíveis.

5.3.1. Hardware

Os 5 sensores são, o fototransistor, o sensor de presença, botão *push-button* e o *ldr* (*light dependent resistor* – resistor dependente de luz). Os 3 atuadores são, um climatizador, uma lâmpada e uma tomada. Todos os sensores são disponibilizados de forma estratégica para registrar informações sobre a porta se está aberta ou fechada, se o ambiente está com pouca luminosidade, há presença de pessoas no interior da residência, etc. Tais informações são coletadas pelo microcotrolador, utilizando a lógica de circuitos digitais.

A disposição dos sensores foi instalada em pontos estratégicos, o fototransistor instalados nas portas, o *push-button* na janela, um sensor de presença no interior da casa, e um *ldr* para identificação de luminosidade na residência. E os controladores, o climatizador colocado para deixar a temperatura ambiente, uma tomada de uso geral localizada em um lugar de preferência do usuário e uma lâmpada para iluminar a residência quando preciso, conforme a figura 21.

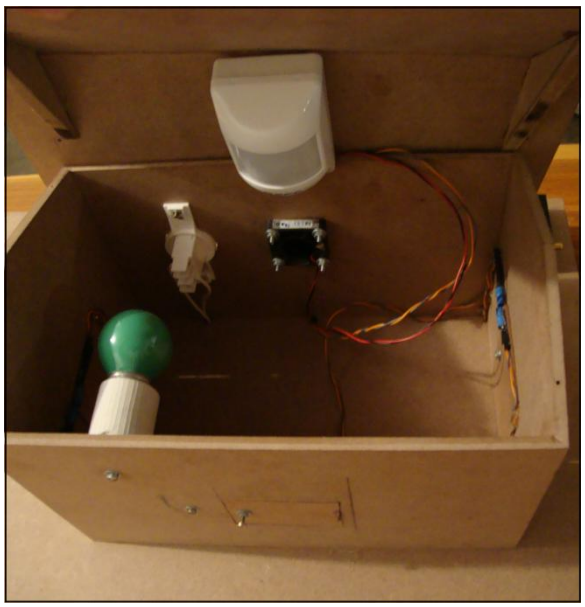

Figura 21 - Sensores e dispositivos de controle.

No primeiro momento, foram feitos protótipos de hardware (Figura 22) para comunicar com o modem GSM. Este protótipo é uma gravadora de microcontrolador da Atmel o Atmega 8515, que disponibiliza de sinais RX e TX - RS232/TTL e de diversas porta de saída ou entradas de sinais de acordo a programação gravada no *chip*.

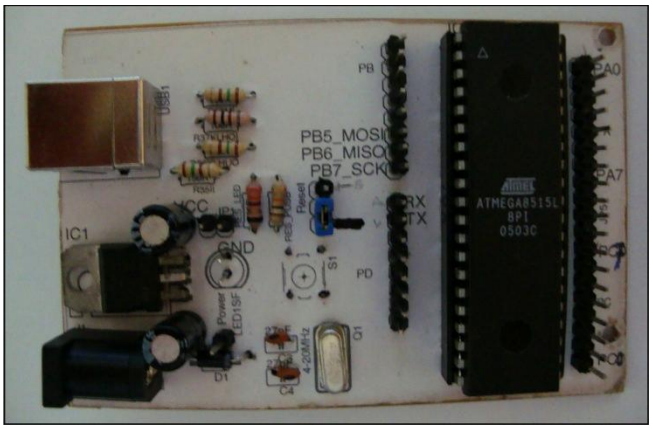

Figura 22 - Protótipo para a comunicação com o módulo.

Para a aquisição desse protótipo foi projetado o esquemático figura 35 em apêndice A, que é constituído de do microcontrolador Atmel 8515, uma fonte de alimentação para todo o circuito e resistores e capacitores para a gravação e no apêndice A, figura 36 e 37, o layout da placa.

Para fazer a comunicação com o conector DB9 disponível no modem GSM, projetou-se uma placa utilizando um Max232 (Figura 23) que faz a conversão dos sinais RS232/TTL do microcontrolador para os sinais RS232 do módulo.

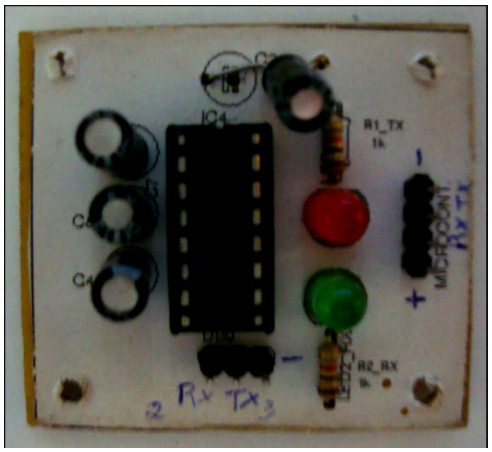

Figura 23 - Conversor de sinais RS232/TTL para sinais do modulo.

Na Figura 23 acima os quatro pinos são ligados ao microcontrolador o (+) representa 5V e o (-) o GND para a alimentação da placa, o dois pinos do meio RX e o TX são respectivamente pinos de recepção e transmissão, o RX é ligado ao TX da placa do microcontrolador (Figura 23) os três pino são ligados ao conector DB9 o TX ao pino 2 e o RX ao pino 3 e o GND ao pino 5.

Os capacitores eletrolíticos são utilizados para configurar o funcionamento correto do Max232, alguns deles trabalham com sua polaridade invertida conforme a

figura 38 no apêndice A, o capacitor C3 e o capacitor C4 os dois são invertidos a polaridade, e no apêndice A, nas figuras 39 e 40, o layout da placa.

5.3.1.1. Sensores e Atuadores

O fototransistor é um transistor que, ao invés de ser acionado pela corrente presente em seu terminal base, é acionado pela luz que incide sobre ele. Portanto, ele não possui esse terminal, possuindo geralmente um formato muito semelhante ao de um LED. O circuito do sensor fototransistor (figura 41 no apêndice A) é constituído de um emissor e um receptor. O emissor fica emitindo luz a todo momento enquanto o receptor fica aguardando o feixe dessa luz. Se o receptor identifica a luz, a porta está fechada, ele conduz, conforme o circuito passará um sinal de 0 v para o circuito de destino. Caso contrário se a porta for aberta, o receptor não recebe luz. Contudo, irá enviar um sinal positivo +5V ao circuito destino.

O LDR, sensor resistivo e um componente onde há uma variação da luminosidade que incide sobre ele resultando numa variação na sua resistência.

Conforme a figura 41 no apêndice A, o LDR será interligado ao Amplificador Operacional comparador. Um AmpOp, ligado dessa forma, compara a tensão das duas entradas, positiva e negativa. Quando a tensão da entrada positiva for superior a da entrada negativa, encontra-se na saida do AmpOp, a tensão de alimentação do circuito. Se a tensão da entrada negativa for superior à da entrada positiva, acontecerá o contrário, ou seja, na saida do AmpOp encontraremos uma tensao de 0 V. O potenciômetro é utilizado para diminuir ou aumentar a sensibilidade do sensor. O layout da placa está localizado no apêndice A, figura 42 e 43.

O circuito dos relés responsável pela parte de controle do sistema (figura 41 no apêndice), possui 3 ou mais reles que atuam como funcionamento de atuadores. Estes são ligados quando o microcontrolador receber a informação do modem para habilitar o funcionamento.

O circuito dos sensores e circuito dos relés foram montado em uma única placa a imagem vista na (Figura 24) em que a placa acomoda todos os componentes interligados aos sensores e ao CI, LM 339, responsável pelo monitoramento e os componentes interligados ao relé responsáveis pela atuação dos relés que habilita o funcionamento do climatizador, lâmpada ou tomada. Esta placa é interligada a placa principal, pela PORTC e PORTA respectivamente correspondem as portas dos sensores e as portas dos atuadores.

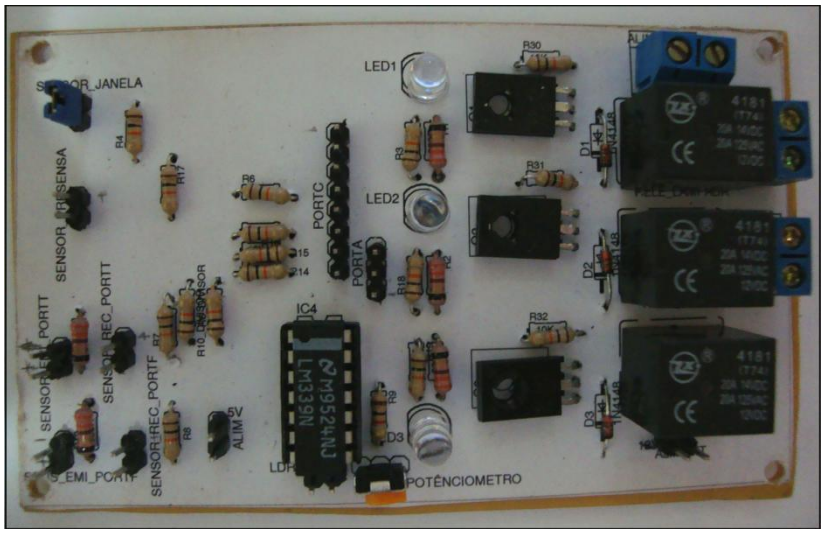

Figura 24 - Placa dos sensores e dos relés.

5.3.2. Maquete

A maquete (Figura 25) utilizada para demonstrar como estão distribuídos os sensores e os atuadores por toda a residência. Nas portas foram utilizados os fototransistores, na janela foi utilizado um *push-botton*, o sensor de presença posicionado no teto por ter uma maior visão do interior da residência e o sensor LDR que poderá ser posicionado em qualquer lugar que não capta a luz emitida pela lâmpada. O climatizador representado pela ventoinha, a tomada e a lâmpada foram fixados na parede da casa.

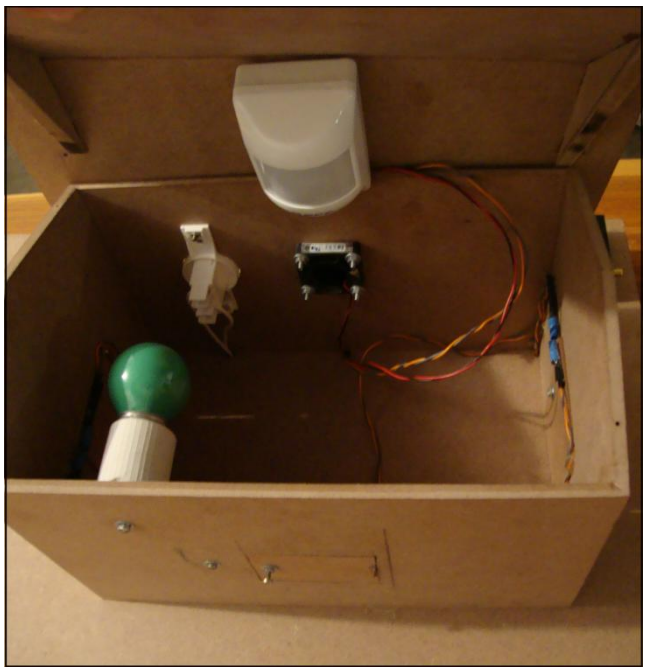
Figura 25 - Maquete, visão do interior da residência.

O sistema principal é composto pelo hardware desenvolvido e o modem GSM que foram posicionados na parte externa, para ter uma facilidade para o manuseio, conforme a figura 26.

Figura 26 - Maquete, posicionamento do hardware e do modem GSM.

5.3.3. Firmware

A codificação do firmware, primeiramente foram feitas todas as pre-configurações do microcontrolador, inicializando os 8 pinos do PORTA como saída, *DDRC = 0* e 8 pinos da POTC como entrada, *DDRA = 255*. Depois foram feitas as configurações da porta serial *RX* e *TX*, utilizando a biblioteca USART disponível para o microcontrolador, setando a taxa de transferência de *9600* e ciclo de clock *4Mhz* equivalente as configurações do modem GSM que está conectado ao microcontrolador. Este processo anterior possibilita o envio e recebimento de mensagens entre microcontrolador e modem, com o objetivo de transferir os comandos AT.

Para o envio dos comandos AT utiliza-se a função uart_puts(“string”); passando a *string* que é composta pelo comando AT, exemplo: uart_puts(“AT+CGATT=1”) e também a função uart_putc('\r'); para caracteres simples. Para o recebimento de dados do módulo utiliza-se a função uart_getc(); na qual a função retorna qualquer dado, desde *status* de retorno dos comandos AT passados ao módulo ou mensagens do próprio cliente.

O algoritmo de monitoramento residencial será feita a verificação dos diversos estados que os sensores iram disponibilizar, a partir deles será montado uma lógica para

distinguir os estados presentes, e será transferido ao celular através comunicação GRPS utilizando a conexão TCP-IP. A verificação dos estados é através da função *analiza_sensores( )* conforme o trecho de código descrito na figura 27.

```
void analiza_sensores(){
      if((valor_temp & 0b00010000)!=0){//JANELA ABERTA
            ja='1';
      }else{ja='0';}
      if((valor_temp & 0b00001000)!=0){//SENSOR DE PRESENSA
            sp='1';
      }else{sp='0';}
      if((valor_temp & 0b00000100)!=0){//SENSOR DE LUZ DETECTOR DE LUZ
            ld='1';
      }else{ld='0';} ...}
```

Figura 27 - Codificação responsável por verificar o estado dos sensor.

O algoritmo de controle tem como base os estados reconhecidos através de um padrão de códigos que o celular do cliente irá emitir ao modem GSM para ativar ou desativar os dispositivos. A codificação referente ao controle residencial primeiramente temos uma verificação se chegou uma mensagem e que possivelmente ira fazer outro teste para saber qual controle que o cliente solicitou conforme a figura 28.

```
      if (buffer1[i]=='C'&&buffer1[i+1]=='E'&&buffer1[i+2]=='L'&&
          buffer1[i+3]=='_'&&buffer1[i+4]=='S'&&buffer1[i+5]=='I'&&
          buffer1[i+6]=='N'&&buffer1[i+7]=='A'&&buffer1[i+8]=='I'&&
          buffer1[i+9]=='S'){
            if(buffer1[i+11]=='1'){la='1'; PORTA |= (1<<2);
            }else{la='0';PORTA &= ~(1<<2);}
            if(buffer1[i+12]=='1'){to='1'; PORTA |= (1<<1);
                  }else{to='0'; PORTA &= ~(1<<1);
                  }
                  if(buffer1[i+13]=='1'){cl='1';PORTA |= (1<<0);
                  }else{cl='0'; PORTA &= ~(1<<0);
                  }
                                    FLAG = 9; valor_temp=PINC;
                  valor_temp_ant=valor_temp;
                  uart_putc('\n'); uart_puts("AT+CIPSEND");
                  uart_putc('\r');
                  index =0;inicializa_buffer(); break;
            }...
```

Figura 28 - Codificação responsável para executar o controle solicitado.

5.3.4. Ferramentas de desenvolvimento

A ferramentas de desenvolvimento do firmware contido microcontrolador AVR Atmega8515 foi a IDE AVR Studio, disponível na figura 29 a imagem do ambiente de desenvolvimento nele é possível programar nativamente em assembly, mas para programar em C é necessário instalar o WinAVR, pacote de programas em código aberto para desenvolvimento com microcontroladores da família AVR da Atmel no ambiente Windows. Ao ser instalado o WinAVR, ele que fica acoplado ao AVR Studio.

Uma das grandes vantagens de utilizar a linguagem C é porque ela uma linguagem de alto nível e bastante acessível ao programador para realizar as rotinas. Mas em contrapartida consome um pouco mais de memória programável do microcontrolador.

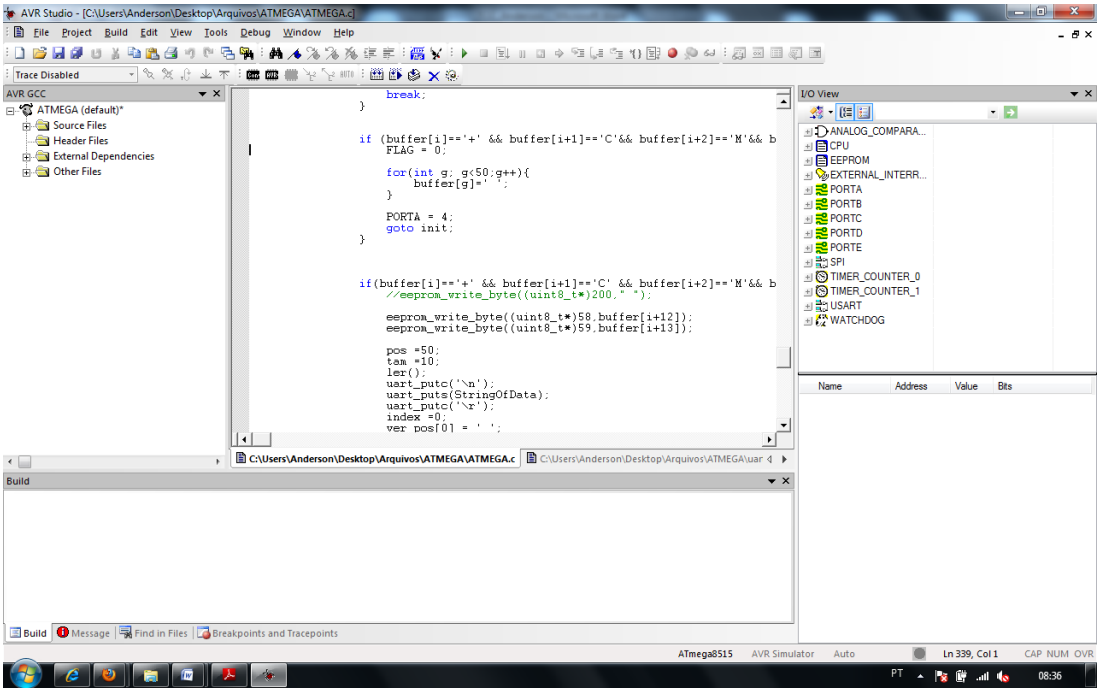

Figura 29 - Imagem do ambiente de programação AVR Studio 4®

5.4. Aplicação do celular

A aplicação implementa a interface com o usuário na aplicação móvel. Além disso, ela também realiza a conexão com a outra aplicação remota que se encontra no servidor para que seja possível a troca de mensagens.

Existe na aplicação uma estratégia de um *loop* infinito empregado para aguardar mensagens figura 30, assim que uma mensagem chega com uma terminação de um caractere representando fim de linha, por seguinte é feita a verificação da mensagem e

apresentada a ação que o controlador enviou ao servidor, e o servidor transmitiu ao celular.

```
While (true) {
        StringBuffer sb = new StringBuffer();int c = 0;
        while (((c=entrada.read())!='\n')&&(c!=-1)){
            sb.append((char)c);//pega a entrada e guarda em sb
        }
        if (sb.toString().equals("CONN>Controlador")) {
            controladorStringItem.setText("Conectado");
            despachante.send("GETSINAIS\n");
        }
        if (sb.toString().equals("DSCN>Controlador")) {
            controladorStringItem.setText("Desconectado");
        }
        if (sb.toString().startsWith("SINAIS>")) {
            lugarSeparador=sb.toString().indexOf(">")+1;
            conjunto=sb.toString().substring(lugarSeparador);
            if (conjunto.charAt(0) == '1') {
                  janelaStringItem.setText("ligada.");
            } else {janelaStringItem.setText("desligada.");}
            if (conjunto.charAt(1) == '1') {
                  janelaStringItem.setText("ligada.");
            } else {janelaStringItem.setText("desligada.");}
            if (conjunto.charAt(2) == '1'){
                  janelaStringItem.setText("ligado.");
            } else {janelaStringItem.setText("desligado.");}
            if (conjunto.charAt(3) == '1'){
                  janelaStringItem.setText("aberta.");
            } else {janelaStringItem.setText("fechada.");}
            if (conjunto.charAt(4) == '1'){
                senPresencaStringItem.setText("ativado.");
            } else{senPresencaStringItem.setText("desativado.");}
            if (conjunto.charAt(5) == '1'){
                  luzStringItem.setText("não existente.");
            } else {luzStringItem.setText("existente."); }
            if (conjunto.charAt(6) == '1'){
                  portaFrenteStringItem.setText("aberta.");
            } else {portaFrenteStringItem.setText("fechada.");}
            if (conjunto.charAt(7) == '1'){
                  portaTrasStringItemItem.setText("aberta.");
            } else {portaTrasStringItemItem.setText("fechada.");
        }
}
        if (c == -1) {break;}
}
```

Figura 30 - Estratégia de um loop para receber mesagens.

Para que o usuário possa enviar uma mensagem, existe uma rotina para a composição de mensagens figura 31, e envio que é feita através da classe *Despachante*. Na interface da aplicação existe um menu, e cada iten, envia uma *string* com o status desejado pelo usuário precedido de um prefixo "CEL_SINAIS". Quando o celular conecta ao servidor ele pede uma atualização do status do controlador, ou a quando o usuário o solicita uma atualização é condicionado o envio da *string*, "CEL_GET-_SINAIS".

```
Public void commandAction(Command command, Displayable displayable) {
        if (displayable == clienteForm) {
            if (command == atualizaCommand) {
                despachante.send("CEL_GETSINAIS\n");
            } else if (command == climatizadorCommand) {
                despachante.send("CEL_SINAIS>001\n");
            } else if (command == desligaTodosCommand) {
                despachante.send("CEL_SINAIS>000\n");
            } else if (command == exitCommand) {
                try {
               // ==> Envia mensagem de saída pro servidor
                    despachante.send("/fui\n");
               // ==> Fechar todas as conexões antes de sair
                    if (despachante != null) {
                        despachante.stop();
                    }
                    if (saida != null) {
                        saida.close();
                    }
                    if (entrada != null) {
                        entrada.close();
                    }
                    if (conexao != null) {
                        conexao.close();
                    }
                } catch (IOException ex) { ex.printStackTrace();
                }
                midlet.notifyDestroyed();
                midlet.destroyApp(true);
            } else if (command == lampadaCommand) {
                despachante.send("CEL_SINAIS>100\n");
            } else if (command == ligaTodosCommand) {
                despachante.send("CEL_SINAIS>111\n");
            } else if (command == tomadaCommand) {
                despachante.send("CEL_SINAIS>010\n");
            }
        }
    }
```

Figura 31 - Rotina para a composição de mensagens.

A classe *Despachante* também mantém seu fluxo em um loop, aguardando pelo recebimento da mensagem que se deseja enviar para aplicação remota. No caso em questão, a condição de quebra do fluxo normal de operação é ter uma mensagem com valor *null* sendo posta para entrega.

Existe em vários pontos da classe *ClienteRemotoMobile*, uma rotina que invoca o método *send()* contido na classe *Despachante*, figura 32. Sua função é simplesmente enviar pela rede a mensagem ao destinatário com o qual se mantém a conexão que, neste caso, é a aplicação do cliente controlador (microcontrolador - modem).

```
public class Despachante extends Thread {
    private OutputStream saida;
    private String messagem;
    public Despachante(OutputStream saida) {
        this.saida = saida;
        start();
    }
    public synchronized void send(String mensagem) {
        messagem = mensagem;
        notify();
    }
    public synchronized void run() {
        while (true) {
            // Espera até uma conexao
            if (messagem == null) {
                try {
                    wait();
                } catch (InterruptedException e) {
                }
            }
            if (messagem == null) {break;}
            try {
                saida.write(messagem.getBytes());
            } catch (IOException ioe) {ioe.printStackTrace();
            }
            messagem = null;
        }
    }

    public synchronized void stop() {
        messagem = null;
        notify();
    }
}
```

Figura 32 - A classe *Despachante*.

O método *send()* foi implementado desta maneira porque, na classe de interface com o usuário, não é instanciado nenhum objeto do tipo *Despachante*. Isto apenas ocorre dentro da classe *ClienteRemotoMobile*. Esta implementação foi feita desta maneira para seguir alguns princípios de organização de código, encapsulamento correto e até mesmo buscando a atomicidade de algumas operações.

5.5.Ferramentas de desenvolvimento

A aplicação do celular assim como do servidor, foi desenvolvida na plataforma Java, foi à base de tecnologia para a implementação destas duas etapas. Deste modo, todas as ferramentas e acessórios de desenvolvimento foram influenciados por esta escolha. Mais ainda, as ferramentas são todas distribuídas pelo mesmo fornecedor, Sun Microsystems.

A seleção e escolha das ferramentas utilizadas tiveram, além dos óbvios critérios de qualidade técnica, uma característica considerada fundamental: distribuição e uso livre para o desenvolvimento de aplicações como as propostas nas etapas do servidor e no cliente móvel.

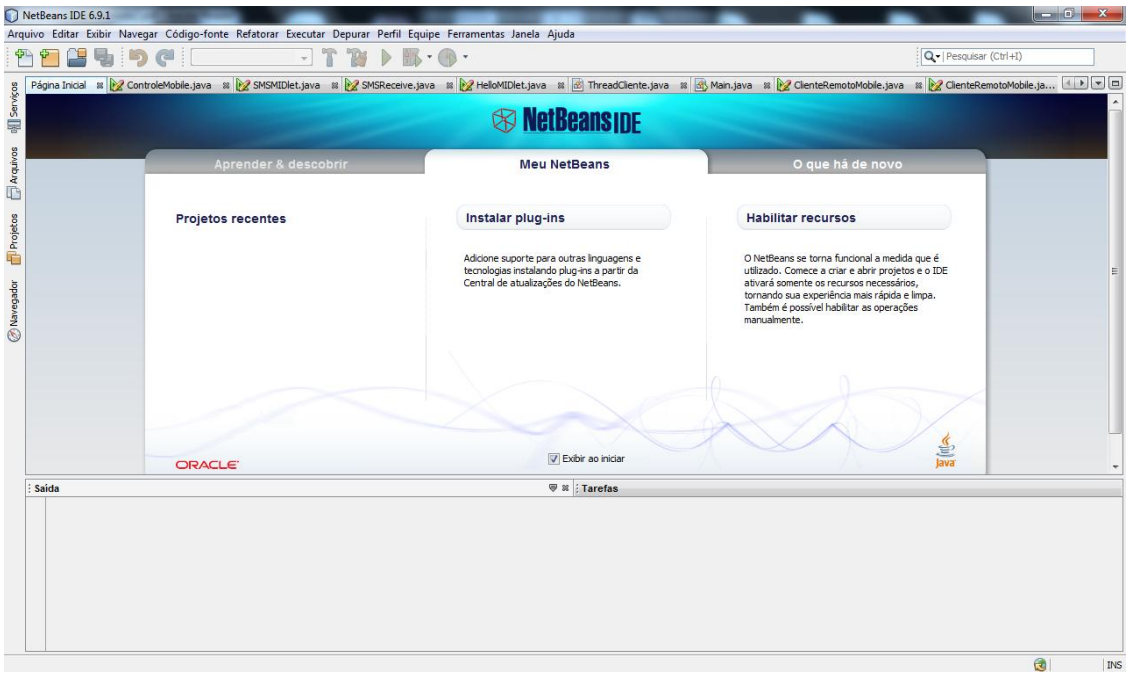

Figura 33 - NetBeans IDE.

5.5.1. Netbeans IDE.

O NetBeans é um IDE mantido pela Sun Microsystems que é distribuído gratuitamente e tem código aberto. Hoje, é uma das principais ferramentas para desenvolvimento em Java disponíveis. Também, é adotada por um grande número de desenvolvedores, o que colabora com seu desenvolvimento e evolução, figura 33.

Desde seu início até os dias atuais, o NetBeans vem ganhando inúmeras características importantes para desenvolvedores de vários segmentos. De fato, em seus

dez anos de vida, é possível desenvolver em algumas outras linguagens de programação como C++, PHP e Ruby.

No caso do desenvolvimento em Java, o NetBeans oferece a possibilidade da implementação de aplicações em todas as áreas: desktop, móvel,Web e corporativas. Isto é uma característica muito atraente para o desenvolvedor e colaborou de maneira muito benéfica aos propósitos do projeto descrito neste relatório.

O NetBeans segue uma tendência vista em muitas outras soluções que é a de não se limitar e ser um excelente ambiente de desenvolvimento mas, se transformar em uma plataforma sobre a qual se pode criar projetos de maneira a usufruir de toda a tecnologia envolvida e disponível na ferramenta.

Existem inúmeras capacidades e características disponíveis no NeBeans IDE. Discorrer sobre todas elas, além de estar fora do escopo deste relatório, tornaria este relatório muito extenso para um documento desta natureza.

5.5.2. Mobility pack

O Mobility Pack é instalado no NetBeans IDE como um *plugin*. Ele proporciona ao ambiente de desenvolvimento tudo o que é necessário ao desenvolvimento de aplicações para dispositivos móveis.

Ao ser instalado ele oferece suporte para CLDC para as mais atuais versões como do mesmo modo ao MIDP 1.0, 2.0 e 2.1. Oferece também suporte a CDC com Personal Profile.

Além deste suporte, o desenvolvedor tem ao seu dispor ferramentas para desenvolvimento visual de maneira muito simples e intuitiva, especificando inclusive o fluxo da aplicação de forma visual. A parte de testes de debug também foi favorecida com a presença de emuladores bastante eficientes que tem um bom desempenho nas simulações.

Capítulo 6. TESTES E ANÁLISES

Foram realizados diversos testes presente na realidade sobre o monitoramento e controle residencial a partir disso retirou uma amostragem para fazer as devidas análises sobre a eficiência e qualidade do sistema, perante as grandes situações existentes sobre a ativação do sensor até o recebimento da mensagem no celular do cliente, assim como a eficiência de controle, a partir do envio da mensagem por parte do cliente com o código de ativação ou desativação dos dispositivos existentes na residência.

6.1. Teste e análise I – monitoramento

Tendo a residência em estado normal, fazendo a abertura de uma porta, observa-se que em questão se segundos o cliente recebeu a mensagem do controlador com a descrição do estado (Porta da frente: aberta.) conforme a figura 37.

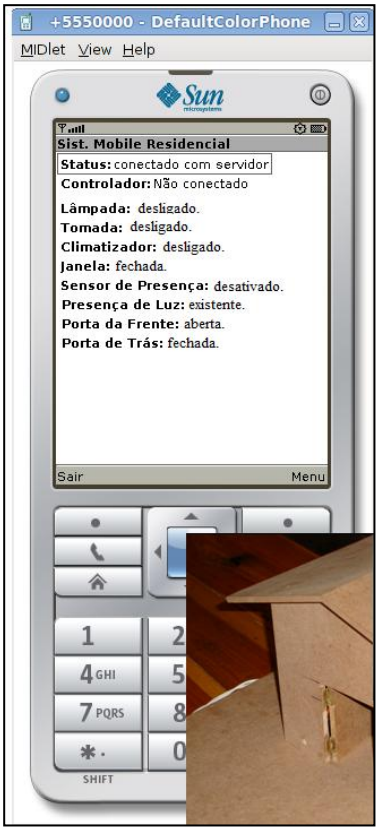

Figura 34 - Teste 1 - Porta da frente aberta.

O monitoramento residencial está bem funcional e eficiente, pois o mesmo trás a descrição do que está acontecendo na residência, através da mensagem enviada via conexão TCP , a partir disso a aplicação do celular interpreta e mostra na sua interface ao usuário.

6.2.Teste e análise II – controle

Tendo a residência em estado normal, quando fizer o envio de um comando clicando em “menu” e “lâmpada”, a conexão existente com o servidor, o servidor transfere ao cliente controlador, observamos que em questão de segundos o sistema ligou a lâmpada.

O controle residencial está bem funcional e eficiente, pois o mesmo é ativado a partir de um comando clicando em “menu” e “lâmpada”, ao servidor o sistema reconhece o comando e executa a atuação do dispositivo solicitado.

O processo de envio de mensagens do cliente emissor ao receptor foi efetivado instantaneamente chegando há 2 segundos. Estando incluso nesta medida de tempo o momento em que um evento, provocado pelo usuário ou pela mudança de estado de algum dos sensores, dispara uma ação e o envio da mensagem, concluindo que o sistema teve uma performance excelente.

Capítulo 7. CONCLUSÃO

Este trabalho monográfico teve como objetivo geral, a construção de um sistema de gerenciamento e controle residencial remoto, dotada de dois sistemas embarcados conectados a uma rede GPRS, intermediado por um servidor a fim de implementar um algoritmo capaz de identificar as situações distintas utilizando sensores e fazer o controle a partir do usuário.

As três etapas da solução, servidor, sistema controlador e sistema móvel (celular), tiveram seu comportamento funcional dentro do que se esperava. Não houve nenhum entrave para a instalação e execução de nenhuma delas. O tempo de execução esteve sempre compatível com qualquer outra aplicação mesmo até comparando com aplicações compiladas nativamente para os ambientes operacionais.

Quanto à parte de comunicação, considerando o trajeto desde a parte da rede GSM da operadora entrando pela rede IP e Internet, teve uma performance satisfatória. O tipo de conexão escolhida no dispositivo móvel foi o GPRS.

Um dos pontos dificultadores encontrado foram à implementação das configurações do modem e configuração do modem para conectar a rede GPRS. A implementação do algoritmo em linguagem de programação C, utilizado no microcontrolador, teve uma limitação do uso de linguagem de alto nível em sistemas embarcados, pois o tamanho da memória de programa disponível é considerado pequeno para gravar um firmware desenvolvido em C.

Outra dificuldade foi à configuração dos telefones celulares. Nem todos os celulares oferecem facilidade para ativação das conexões e alguns que foram testados, de uma determinada marca, impediram a aplicação móvel de utilizar a conexão disponível.

Também foram observados que alguns detalhes da teoria não podem ser ignorados, pois na prática são esses detalhes que fazem a diferença e podem induzir à graves erros, como o exemplo do estudo padrão de comunicação do modem GSM, que tem que haver uma sincronia nas informações transmitidas e recebidas.

Concepções simples podem ter um bom campo de aplicações. Esta é uma idéia que foi reforçada pelo trabalho neste projeto. A quantidade de aplicações envolvidas além de razoavelmente grande mostra boas possibilidades de variação. É fácil encontrarmos disponíveis várias soluções de controle de dispositivos e a inclusão de

dispositivos móveis neles, como foi visto aqui, é indiscutivelmente possível e relativamente fácil.

Em suma, todos os resultados foram bastante animadores e demonstram que este tipo de solução, dentro da abordagem adotada, pode ser perfeitamente utilizado em outros produtos. Para criar um servidor local no sistema controlador, sendo mais lógico em algumas situações, fica a proposta para trabalhos futuros.

7.1. Sugestão para trabalhos futuros.

As sugestões para trabalhos futuros, controle de concorrências dos clientes, autenticações, verificando se o cliente possui autorização para conectar ao servidor, e também a criação de uma interface, para o servidor, no intuito de monitorá-lo via algum terminal.

Outra sugestão que pode ser feita, a depender do ambiente do usuário, é a alteração do servidor, criando junto à aplicação do controlador (microcontrolador - modem). Ou quando se trata de um sistema muito simples e o cliente não prefere que o servidor fique remotamente, uma boa solução é utilizá-lo no modem GSM.

Por fim, podem criar clientes para sistemas de celulares que possui tecnologia android, este celular possui mais tecnologia, que podem ser exploradas, construindo softwares com mais aparências e interface mais intuitiva.

REFERÊNCIAS

ARTHUR,Rangel; FIGUEIREDO, Rafael ; NASCIMENTO, Luiz Henrique Bonani. **Projeto de um Controlador de Alarme de Carro via SMS.** Disponível em: http://revistavirtual.unisal.br:81/seer/ojs-2.2.3/index.php/123/article/view/47/57. Acesso em 30 mai. 2011

ATMEL, disponível em: www.atmel.com. Acesso em 30 jun. 2011.

AURESIDE (Associação Brasileira de Automação Residencial). **Temas técnicos**: Conceitos Básicos, Benefícios da automação. Disponível em: http://www.aureside.org.br/temastec/default.asp?file=concbasicos.asp Acesso em: 15 jul. 2011.

AUTRAN, Gustavo. **Assaltos a residências preocupam cariocas**. Disponível em: http://oglobo.globo.com/rio/mat/2009/10/15/assaltos-residencias-preocupam-cariocas-768076135.asp . Acesso em: 30 abr. 2011.

BOLZANI, Caio A. M. **Residências Inteligentes**: um curso de Domótica. 1.ed. São Paulo: Editora Livraria da Física, 2004.332 p.

KIOSKEA - Computing community, disponível em: http://www.tato.ind.br/detalhesproduto.asp?id=87 Acesso em 30 jun. 2011.

MUCHOW, John W.Core J2ME – Tecnologia & MIDP. 1ª Ed, São Paulo: Editora Pearson Makron Books, 2006.

MANGANELLI, Elenice Colle; ROMANI, Juliano. **Protocolos de sincronização de dados em ambientes wireless: um estudo de caso**. 2004. 139 f. Tcc (Bacharelado) - Curso de Ciências Da Computação, Universidade Federal De Santa Catarina, Florianópolis, 2004.

MEDINA, Marcelo.net. **Dicionário Do Aurélio Online** – 2008, disponível em: www.atmel.com. Acesso em 03 mar. 2011.

NOGUEIRA, Octavio Tato Equipamentos Eletrônicos Ltda. Disponível em: http://www.tato.ind.br/detalhesproduto.asp?id=87 Acesso em 28 mai. 2011.

OLIVEIRA, Adriano Marcio. **Automação Residencial**. Disponível em: http://www.aureside.org.br/temastec/TCC_AutoRes10.pdf . Acesso em: 18 maio 2011.

OLIVEIRA, André Schneider de. **Sistemas Embarcados: hardware e o firmware na pratica**/ Andre Schneider de Oliveira, Fernando Souza de Andrade. –1. ed. São Paulo: Érica, 2006.

PRESTES, Maria Luci de Mesquita. **A pesquisa e a construção do conhecimento científico / Maria Luci de Mesquita Prestes**. São Paulo: Respel, 2002.

PIROPO, B. http://www.bpiropo.com.br/fpc20051107.htm. Acesso em 05 out. 2011.

SHANGHAI, **SIMCom Wireless Solutions Ltd**. 2008, documentação, disponível em:

http://res.trilha21.com/001000340/doc/ATminglingji_AT_COMMAND_INTERFACE.pdf . Acessado em 15/03/2010.

SINCON, **Segurança**, documentação, disponível em: http://www.tato.ind.br/detalhesproduto.asp?id=87. Acesso em 30/04/2011

SÍNDICONET, **Segurança**, documentação, disponível em: http://www.sindiconet.com.br/4196/informese/noticias/seguran%C3%A7a/conhe%C3%A7a-a-rua-em-sp-onde-quase-todas-as-casas-foram-assaltadas .Acesso em 30/04/2011

SANTOS Ricardo Di Lucia, REDES GSM, GPRS, EDGE E UMTS, Documentação, disponível em: http://www.gta.ufrj.br/ensino/eel879/trabalhos_vf_2008_2/ricardo/1.html. Acesso em 25/08/2011

TANEMBAUM, ANDREW. S. **Redes de Computadores**. 4ª Ed, Rio de Janeiro: Editora Campus, 2003.

TRILHA21, documentação, disponível em: http://www.trilha21.com.br/TrilhaStore/faces/ShowProduct.jsp?id=1. Acesso em 22/03/2011.

APÊNDICE A.

Esquemático e layout das placas de circuito impresso para a confecção do projeto cliente controlador.

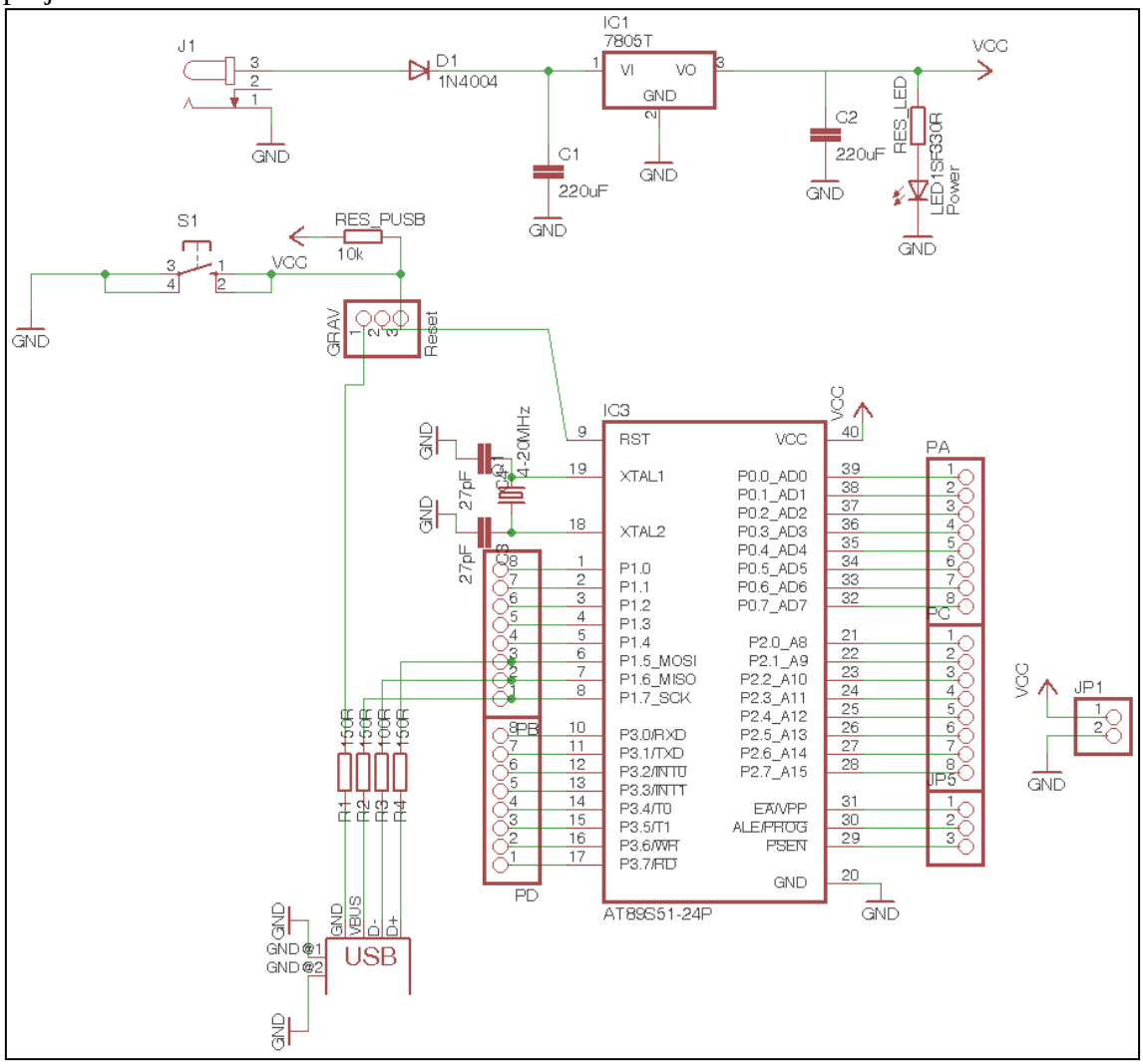

Figura 35 - Esquemático do circuito principal.

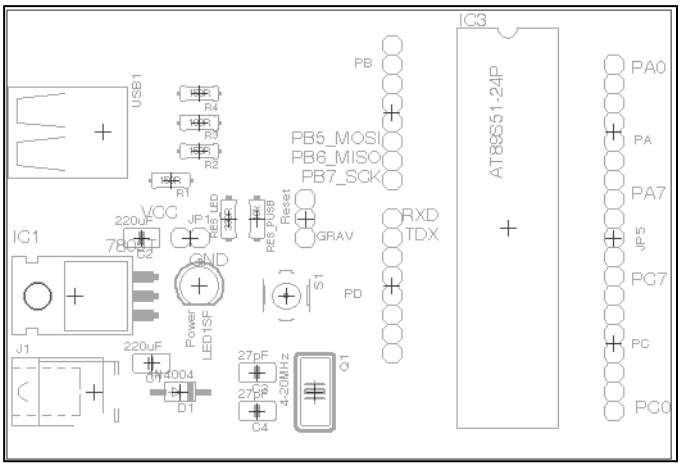

Figura 36 – Layout da parte superior da placa principal.

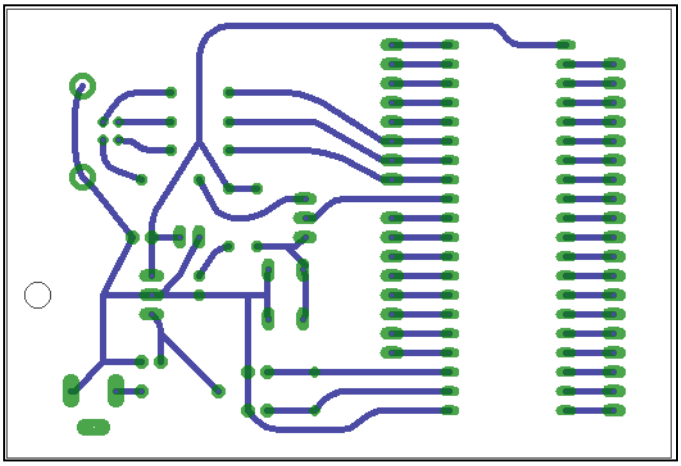

Figura 37 - Layout do circuito impresso da placa principal.

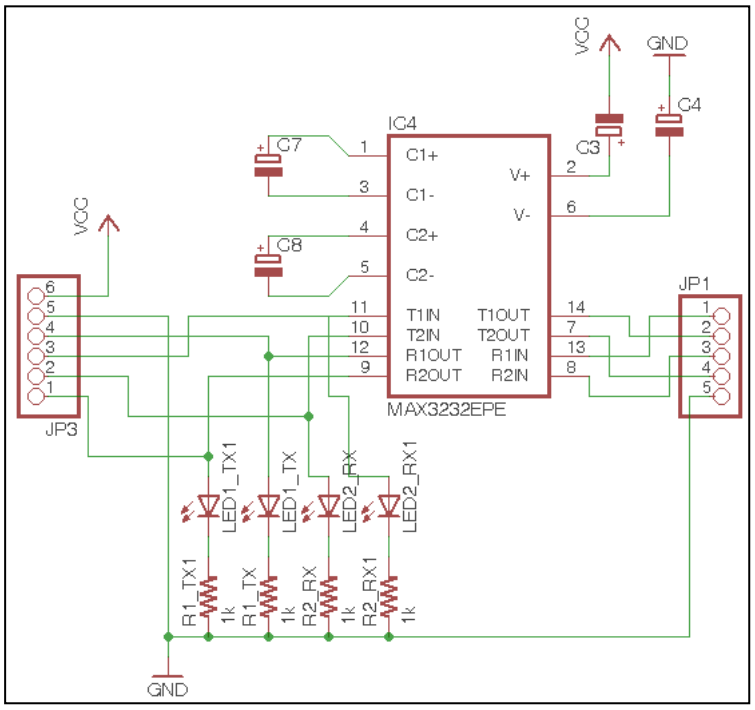

Figura 38 - Esquemático do circuito da placa de conversão dos sinais RS232.

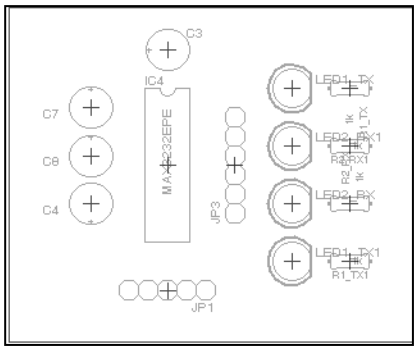

Figura 39 - Layout da parte superior da placa de conversão dos sinais RS232.

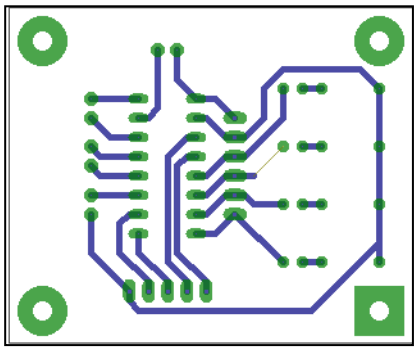

Figura 40 - Layout da placa de conversão dos sinais RS232.

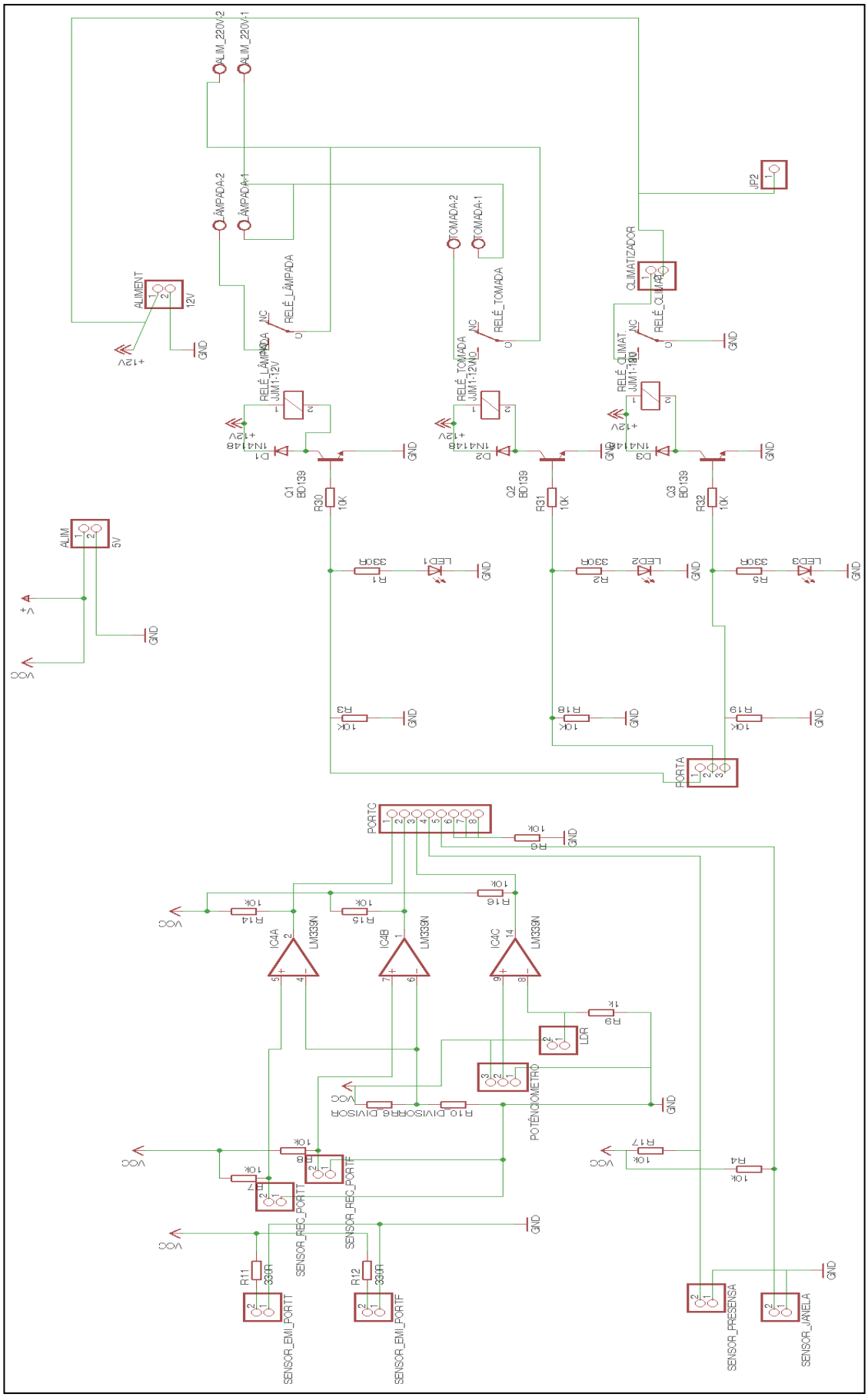

Figura 41 - Esquemático do circuito dos sensores e atuadores.

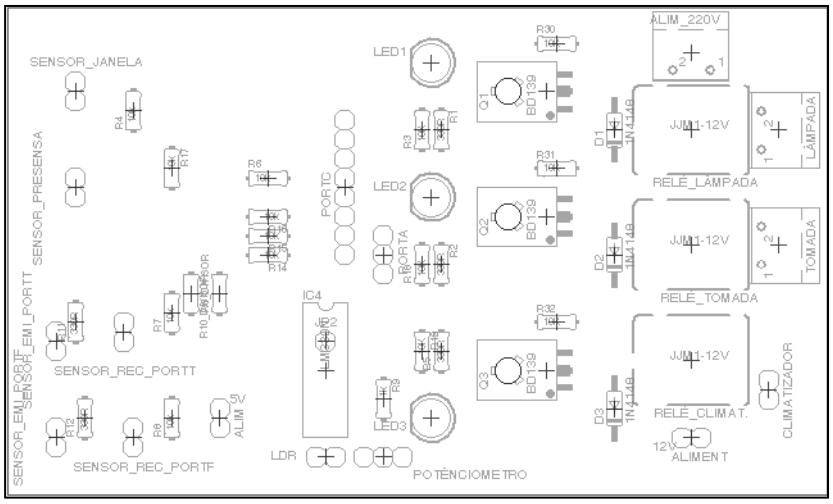

Figura 42 - Layout da parte superior da placa de sensores e atuadores.

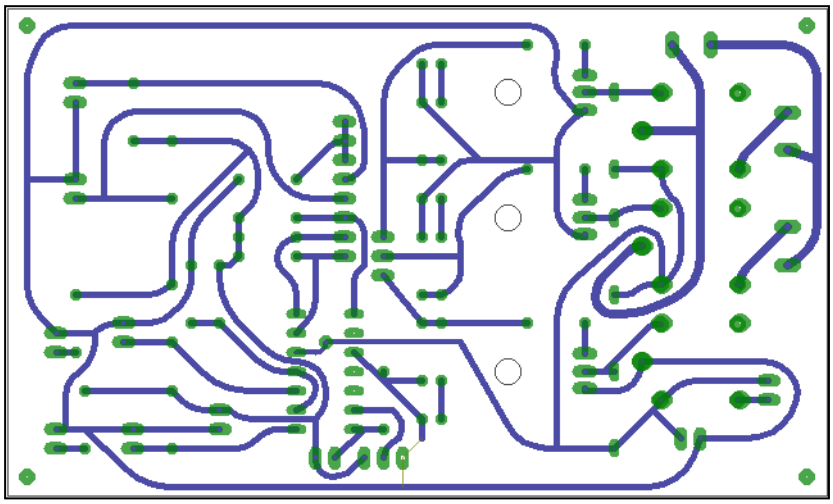

Figura 43 - Layout do circuito imresso da placa de sensores e atuadores.