UNIVERSIDADE ESTADUAL PAULISTA
"JÚLIO MESQUITA FILHO"
FACULDADE DE ENGENHARIA DE ILHA SOLTEIRA

DEPARTAMENTO DE ENGENHARIA ELÉTRICA – DEE
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

MATIUDE - Monitoramento e Acesso a Transdutores Inteligentes (Padrão IEEE 1451) Utilizando Dispositivos Embarcados (celulares e PDA´s)

Tércio Alberto dos Santos Filho

Orientador: Eng. Eletr. Alexandre César Rodrigues da Silva
Co-orientador: Prof. Dr. Aparecido Augusto de Carvalho

Dissertação apresentada à Faculdade de Engenharia - UNESP – Campus de Ilha Solteira, para obtenção do título de Mestre em Engenharia Elétrica.
Área de Conhecimento: Automação.

Ilha Solteira, Junho de 2007.

Dedicatória

Aos meus pais Tercio Alberto dos Santos e Eridan Pereira da Silva Santos e ao meu irmão Carlos Eduardo da Silva Santos

Agradecimentos

Agradeço a Deus por ter me dado força e proteção nos momentos difíceis, sabedoria para escolher os caminhos corretos e também pelos momentos de alegria que tive durante esta caminhada.

Quero agradecer aos meus pais Tércio Alberto dos Santos e Eridan Pereira da Silva Santos, pelo carinho, o amor, o apoio nas horas difíceis e pela força que eles sempre me deram confiando no meu trabalho. Descrever meus agradecimentos pelos meus pais é a parte mais difícil do meu trabalho. Agradeço a cada instante da minha vida por Deus ter me dado pais tão maravilhosos.

Quero agradecer ao meu irmão Carlos Eduardo da Silva Santos vulgo, (Caca), que sempre esteve ao meu lado, como irmão, amigo, companheiro e entre outras qualidades. Este meu irmão, sem medir esforços esteve ao meu lado nos momentos de felicidade ou de tristeza e que sem hesitar acreditou a cada momento em meu trabalho. Agradeço também a sua esposa Mirelle, pelas diversas formas que contribuiu para o melhoramento do meu trabalho e sua filha Lavinia.

Ao meu orientador Prof. Eng El. Alexandre César Rodrigues da Silva, por ter acreditado em meu trabalho, por me ensinar e desafiar durante os dois últimos anos contribuindo para a minha formação. Agradeço também pela paciência e dedicação despendidas durante a condução do trabalho. No campo pessoal, agradeço a amizade e os diálogos descontraídos que mantivemos no tempo em que trabalhamos.

Ao meu co-orientador Prof. Dr. Aparecido Augusto de Carvalho que na ausência do meu orientador esteve contribuindo com minha pesquisa.

A minha namorada Denise Rodrigues Dias que esteve presente em alguns dos principais desafios de minha vida e que sempre me deu força e acreditou no meu trabalho. Pelas longas conversas, pelo carinho, compreensão e amor.

Ao meu amigo Thiago Alexandre Prado por ter me ajudado nas pesquisas, dando apoio ao meu trabalho, sugestões construtivas e pela amizade. Agradeço a companhia nas viagens a congressos, viagens agradáveis e de grande aproveitamento quanto para o nível profissional e pessoal, e que sempre em nossos projetos brincamos de forma responsável.

Ao meu amigo Marcos Antonio Estremot por confiar em meu trabalho, apoiar e ajudar. Agradeço pelas nossas conversar durante nossas viagens e pelas madrugadas em que passamos trabalhando e se divertindo.

Agradeço aos meus amigos de Rio Verde pelo apoio e a força oferecida durante todos os momentos, Heverton, Vinícius, Rodrigo, Wanderlan, Rafael Ratke e sua esposa Bruna.

Agradeço aos meus primos que confiou em meu trabalho, André, Marcelo, Leandro, Iara, Lara.

Agradeço aos meus tios pelas conversas e pelo apoio, Olivar, Olívio, Patrícia, Carmem Luzia, Ernesta, Elma e Adilson.

Agradeço a família de Denise pelo apoio e o carinho que me deram, José Rodrigues Dias, Alice, Daniela, Valdir, Denildo, Joyce.

Agradeço a Eugênia Guedes de Carvalho Ferreira que desde quando comecei o trabalho ela esteve presente na minha vida, dando conselhos e me orientando.

Agradeço aos meus amigos do LPSSD (Laboratório de Processamento de Sinais e Sistemas Digitais) que tive a oportunidade de estar trabalhando junto Silvano, Edson, Ricardo Pace Ferraz, João Sakamoto, André Lacotis Kokumai, Daniela Martins Pellegrini.

Agradeço ao Valdo, amigo que está presente nas viagens a trabalhado para Jales-SP, que durante desta, tivemos longas conversas e trocas de experiências. Agradeço também por se disponibilizar de seu conhecimento e tempo, trabalhando atentamente na correção do meu trabalho.

Agradeço ao Programa de Pós-Graduação em Engenharia Elétrica de Ilha Solteira – UNESP e a FUNDUNESP (proc. 00861/03 - DFP), ao Conselho Nacional de Desenvolvimento Científico e Tecnológico – Programa Nacional de Microeletrônica – CNPQ - PNM (Proc. 132878/2005-4) e a PI componentes pelo suporte oferecido.

Resumo

Neste trabalho foi desenvolvido um aplicativo para os dispositivos móveis (celulares e PDA´s) chamado MATIUDE (Monitoramento e Acesso aos Transdutores Inteligentes Utilizando os Dispositivos Embarcados) para o monitoramento e acesso aos transdutores inteligentes em conformidade com o padrão IEEE 1451. O sistema realiza a comunicação com o NCAP utilizando a rede ethernet. A comunicação entre o NCAP com os módulos inteligentes é realizada por meio da interface física padronizada TII podendo acoplar num total de 255 transdutores por módulo STIM. Os testes foram realizados em um sistema de distribuição de água no laboratório Hidrologia e Hidrometria, na maquete de uma granja e no monitoramento de temperatura ambiente, porém desenvolvidos no Laboratório de Processamento de Sinais e Sistemas Digitais. Os resultados obtidos comprovam o bom funcionamento do sistema.

Palavras Chaves: Padrão IEEE 1451, STIM, NCAP, dispositivos móveis, celular, PDA.

Abstract

On this work, an applicatory for mobile devices (cellular and PDA´s) called MATIUDE (Monitoring and Access to the Smart Transducers Using the Embedded Devices) was developed for monitoring and access to the smart transducers in compliance with the standard IEEE 1451. The communication between the NCAP and the smart modules is performed though the physical interface standardized TII limiting the system in a total of 255 transducers per module STIM. The tests were carried in a water distribution system in the Hydrology and Hydrometric laboratory on a mockup's farm and ambient temperature monitoring, both developed in the Signal Processing and Digital System Laboratory. The results prove the good working.

Palavras Chaves: Standart IEEE 1451, STIM, NCAP, cellular, PDA´s.

Lista de Ilustrações

Lista de Tabelas

Lista de Abreviaturas e Siglas

API´s	Interface de Programação de Aplicativos
CDC	Configuração dos Dispositivos Conectado
CDDL	*Common Development and Distribution License*
CDMA	*Code Division Multiple Acess*
CLDC	Configuração dos Dispositivos Conectados Limitados
EIA/TIA	Associação das Indústrias Elétricas e Telefonia dos E.U.A
EPP	*Enhanced Parallel Port*
FPGA	*Field-Programmable Gate Array*
GPRS	*General Packet Radio Service*
GSM	*Global System for Mobile*
http	*HyperText Transfer Protocol*
IDE	*Integrated Development Environment*
IEEE	*Institute of Eletrical and Eletronic Engineers*
IP	*internet Protocol*
J2EE	*Java 2 Enterprise Edition*
J2ME	*Java 2 Micro Edition*
J2SE	*Java 2 Standart Edition*
JDBC	*Java Database Connectivity*
JDK	*Java Development Kit*
JNI	*Java Native Interface*
JSP	*JavaServer Page*
JVM	*Java Virtual Machine*
KVM	*Kilo Virtual Machine*
MATIUDE	Monitoramento e Acesso a Transdutores Inteligentes Utilizando Dispositivos Embarcados
MIDP	*Mobile Information Device Profile*
MMC Card	*MultiMediaCard*
MMS	*Multimidia Message Service*
MMX	*MultiMídia eXtension*
NCAP	*Network Capable Application Processor*

PCA	*Personal Client Architecture*
PDA	*Personal Digital Assitent*
PHP	*Hypertext Preprocessor*
PXA	*Processor XScale Architecture*
RI	Implementação de Referência
ROM	*Random Only Memory*
RTOS	*Real Time Operating System*
SD Card	*Secure Digital Card*
SDK	*Software development Kit*
SEM	*Enhanced Message Service*
SMS	*Short Message System*
SoC	*System-on-a-Chip*
SRAM	*Static Random Access Memory*
SSL	*Secure Socket Layer*
STIM	*Smart Transducer Interafce Module*
TCP	*Transmission Control Protocol*
TDMA	*Time Division Multiple Access*
TEDS	*Transducer Eletronic Data Sheet*
TII	*Transducer Indepedent Interface*
UART	*Universal Asynchronous Receiver/Transmitter*
UDP	*User Datagram Protocol*
USB	*Universal Serial Bus*
VM	*Virtual Machine*
VPN	*Virtual Private Network*
WAP	*Wireless Aplication Protocol*
XML	*eXtensible Markup Language*

Sumário

Capítulo 1

Introdução Geral

1.1. Introdução

A tecnologia dos processadores criou duas grandes vertentes, uma relacionada aos microcomputadores, máquinas que operam em alta freqüência chegando à casa dos GHz (*Gigahertz*) e alto consumo de energia, enquanto que a outra atende aos quesitos dos dispositivos móveis. Os dispositivos móveis possuem comunicação sem fio, processadores mais lentos e com menor poder de processamento, porém altamente miniaturizados e com um consumo baixo de energia, podendo funcionar por longos períodos de tempo sem a necessidade de recarga.

Hoje o poder computacional dos equipamentos celulares e PDA´s (*Personal Digital Assistant*) são semelhantes as dos computadores da década de 90. PDA´s, Smartphones, Celulares, MP3[1] players e demais aparelhos embarcados convergem para uma interação de informação e acessibilidade, trazendo um novo horizonte para o ambiente de *Web-solutions*.

Desenvolver soluções para equipamentos embarcados exige a padronização de ferramentas, métodos e linguagens, que se faz necessária por uma razão: dar à aplicação meios de atuar em diversas soluções (aparelhos) sem necessidade de retrabalhar à aplicação. Esta característica gera menor mão-de-obra no desenvolvimento de aplicações, maior desempenho, aproveitamento de tempo (1).

O número de usuários que utilizam os dispositivos móveis aumenta gradativamente, devido os aparelhos disponibilizarem uma ampla variedades de serviços, como: *web-mail*, *multimídia*, agenda telefônica, jogos e aplicações voltadas para uma determinada tarefa. Por outro lado, as redes de sensores e atuadores a cada dia tornam-se cada vez mais presentes no nosso meio, por exemplo: ao entrar em uma porta, detectores de incêndios, na área da saúde e em diversas outras áreas. O fato de vivermos

[1] MP3 – abreviação de “MPEG Audio Layer-3”.

em um mundo de informações e comunicações exige não apenas que as informações relacionadas com os transdutores sejam compartilhadas de um ponto para o outro, mais também distribuídas entre diversos equipamentos. Em determinados ambientes os valores de leitura dos sensores devem ser analisados constantemente para que sejam tomadas as medidas e precauções necessárias. Dessa forma, o acesso às máquinas para a manipulação do sistema torna-se desgastante e às vezes impreciso, em se tratando de tempo de acesso.

Para a manipulação e monitoramento dos transdutores inteligentes de forma confiavél e eficaz, foi desenvolvida nesta dissertação um aplicativo para os dispositivos móveis, chamado MATIUDE (Monitoramento e Acesso a Transdutores Inteligentes Utilizando os Dispositivos Embarcados). O MATIUDE é capaz de interagir com processador NCAP utilizando a rede ethernet e, desta forma, realizar o monitoramento e o controle dos transdutores inteligentes de acordo com o padrão IEEE 1451 (*Institute of Eletrical and Eletronic Engineers*). O usuário independente de sua localização pode observar como a rede de transdutores inteligentes está se comportando ou até mesmo verificar se houve alguma alteração durante um intervalo de tempo através dos relatórios.

As áreas de aplicações dos dispositivos móveis são diversas, podendo-se citar a área acadêmica, educação, saúde, casas inteligentes entre outras. A exploração dos dispositivos móveis para o acesso a transdutores inteligentes, utilizando ferramentas e recursos de domínio aberto e padronizados são as idéias que norteiam os objetivos deste trabalho.

1.2. Estado da Arte

Ainda que os serviços de telecomunicações móveis tenham emergido comercialmente recentemente, verificou-se a explosão da demanda a partir de meados da década de 90. A trajetória do desenvolvimento da comunicação móvel pode ser entendida a partir da classificação de três gerações (2):

- Primeira geração (1G) – Serviços públicos de comunicação móvel, no início dos anos 80, usavam a tecnologia analógica FDMA (*Frequency Division Multiple Access*) de transmissão de voz por meio de sinais de rádio entre os celulares e estações de rádio. Fundamentalmente essa geração se caracterizou pela baixa taxa de penetração devido aos elevados preços e a limitada qualidade dos serviços e capacidade dos aparelhos, que eram muito grandes e pesados para serem considerados portáteis (2);

- Segunda geração (2G) – Na década de 90 surgiu a segunda geração de serviços públicos de comunicação móvel. Os aparelhos foram reduzidos significativamente em tamanho, permitindo de fato a portabilidade, enquanto a tecnologia analógica foi substituída pela digital que, consequentemente, permitiu também a transmissão de dados. A transmissão de dados utilizada foram TDMA (*Time Division Multiple Access*) e a CDMA (*Code Division Multiple Acess*). Em relação a transmissão dos dados, inicialmente essa taxa não ia além dos 9,6 Kb/s. O GSM (*Global System for Mobile*) diferencia-se muito de seus anteriores sendo que o sinal e os canais de voz digitais, o que significa que o GSM é visto como um sistema de celular de *segunda geração* (2,5G) (2).
- Terceira geração (3G) – A terceira geração dos telefones celulares caracteriza-se pela maior mobilidade e velocidade de transmissão dos dados e a conexão com várias fontes de dados e de aplicações multimídia através do suporte de acesso à internet. A tecnologia de transmissão de rádio utilizada é a W-CDMA (*Wide band Code Division Multiple Access*), uma técnica de rádio de banda larga, podendo chegar a uma taxa de transmissão de até 2 Mb/s, o que permite o acesso à internet móvel e a transmissão de vídeos (2).

Na tecnologia dos dispositivos móveis houve um grande avanço, na transmissão de dados, no aumento do processamento, imagens e na multiplataforma das linguagens de programação utilizadas para desenvolvimento de sistemas e aplicativos. Hoje, pode-se observar a grande quantidade de linhas telefónicas móveis em uso e a grande quantidade de dispositivos celulares e PDA´s disponíveis no mercado com características de *hardware* diferenciadas. Na Tabela 1.1 são apresentadas as tecnologia utilizadas e a quantidade em milhões de assinantes segundo a GSM World (3).

Tabela 1.1 - Estatística de Assinantes de telefonia celular.

Fonte: Wikipédia, a enciclopédia livre.

Data	Mundo	GSM	GSM 3G	CDMA	CDMA 1X	CDMA 1X EV-DO	US TDMA	PDC	iDEN	Anal
Dezembro 2000	721.3	455.1		82.2			65.2	50.8		68.0
Dezembro 2001	935.3	627.1		113.0			93.6	56.8		43.6
Dezembro 2002	1129.8	787.5		142.7			109.2	60.1		30.0
Dezembro 2003	1382.9	1012	2.8	98.9	80.1	4.6	100.1	58.1	13.4	12.9
Dezembro 2004	1714.1	1296	16.3	87.4	131.9	12.3	90.0	54.2	16.8	9.2
Dezembro 2005	2177.1	1709.2	50.0	62.4	213.1	21.2	48.5	46.3	21.1	5.4
Julho 2006	2405.8	1941.6	74.7	37.0	225.0	34.5	26.1	38.5	23.8	4.5

A grande quantidade de dispositivos móveis e a facilidade de desenvolvimento de *software* fazem surgir novos projetos, além da integração com outros dispositivos, como GPS (*Global Position System*) (4).

No ambiente cooperativo, o usuário encontra-se em constante locomoção, de acordo, os seus compromissos podem ser visualizados através dos dispositivos móveis utilizando um *software* de agenda de compromissos cooperativos poupando tempo e recursos. Esse trabalho pode ser visualizado de acordo com a referência (5).

Em outro trabalho, de acordo com a referência (6), pode-se citar a transferência de imagens de vídeo para os aparelhos celulares, ou seja, as imagens obtidas através de uma webcam conectada ao microcomputador são repassadas através da rede para dispositivo celular. Os testes foram realizados utilizando o simulador *Wireless Toolkit* (6).

Podemos citar também o monitoramento da água do rio Caloosahatchee River Basin no sudoeste da Florida utilizando uma rede de tran[2]sdutores inteligentes. Para o desenvolvimento do servidor, foi utilizado o LabVIEW e o Java junto ao JDBC (*Java Database Connectivity*) que foi desenvolvido de acordo com o padrão IEEE 1451. A aquisição desses dados é realizada através de um PDA utilizando a rede *wireless* e também apresentada através da internet (7).

[2] Transdutor inteligente - transdutor é o nome geral dado tanto para sensores quanto atuadores, que são os dispositivos primários de detecção e atuação em um determinado processo.

Capítulo 2

Padrão IEEE 1451

2.1. Introdução

Neste capítulo apresenta-se uma abordagem sobre o padrão IEEE 1451, padrão utilizado para interligar transdutores inteligentes em rede. Como proposta para esta dissertação foi estudada os padrões IEEE 1451.1 e IEEE 1451.2.

2.2. Padrão IEEE 1451

Os microprocessadores começaram a ser empregados na segunda metade da década de 80, visto que virinha a predominar na década de 90. Neste caso, facilitou para que diversas redes de transdutores inteligentes viessem a ser desenvolvida, porém, dificultou a conectividade entre as redes. Este fato ocorre devido os diferentes tipos de rede e a grande quantidade de redes disponíveis, muitas delas proprietária (8).

Pensando neste intuito vieram às primeiras idéias para a padronização, onde ao invés de fabricar transdutores que suportem diversas tecnologias e protocolos de comunicação, um processo mais apropriado foi padronizar a interface entre transdutores e nós de rede, nós e protocolos, tornando assim independente a escolha do transdutor. De acordo, o padrão de interfaceamento IEEE 1451 para transdutores inteligentes.

O padrão IEEE 1451 foi iniciativa da NIST (*National Institute of Standards and Technology*) e da IEEE (*Institute of Electrical and Electronic Engineers*) que então desenvolveram diversos conceitos para a construção de transdutores inteligentes e consequentemente como interligar em rede. Para o desenvolvimento de transdutores inteligentes o NIST/IEEE propõe o emprego de ferramentas padronizadas, sistemas abertos, ferramentas de domínio público, sistema distribuído e plataforma orientada a objeto. O padrão IEEE 1451 é composto por diferentes comitês que foram denominados como: IEEE 1451.0, IEEE 1451.1, IEEE 1451.2, IEEE 1451.3, IEEE 1451.4, IEEE 1451.5, IEEE 1451.6 e IEEE 1451.7. Nesta dissertação faz-se ênfase o modelo IEEE 1451.1 e IEEE 1451.2 que foram os padrões utilizados neste trabalho. O objetivo do

IEEE 1451 é desenvolver uma interface padronizada para conectar transdutores em uma rede de comunicação, assim, as interfaces padronizadas devem utilizar as tecnologias de rede de controle existente e suportar os diferentes tipos de sensores e atuadores, tornando este dispositivo inteligente. A seguir, apresentam-se os dois padrões IEEE 1451.1 e IEEE 1451.2 utilizados nesta dissertação.

2.2.1. Padrão IEEE 1451.1

O padrão estabelece uma divisão do sistema em dois módulos, um deles contendo o transdutor e o outro contendo os elementos necessários para a comunicação com a rede. Dando origem ao padrão IEEE 1451.1 e ao padrão 1451.2.

O padrão IEEE 1451 introduz o conceito de Processador de Aplicação com Capacidade de Operar em Rede (NCAP). O NCAP é um módulo que contém tanto elementos de *software* quanto elementos de *hardware,* onde é responsável pela troca de informação ocorrida entre o transdutor e o ambiente externo que pode ser outro transdutor ou uma rede na qual está conectado. Para desempenhar estas atividades, o NCAP deve possuir capacidade de controlar o STIM e comunicar-se com uma rede de controle como, por exemplo, a Ethernet. O comitê IEEE 1451.1 apresenta uma configuração detalhada principalmente na especificação do *software* de aplicação [10]. Para maiores detalhes sobre o padrão IEEE 1451.1 pode ser encontrado em (8)(9) (10).

2.2.2. Padrão IEEE 1451.2

O padrão IEEE 1451.2 foi o primeiro módulo aprovado pela família IEEE 1451. Os principais objetivos do padrão IEEE 1451.2 são as seguintes:

- Fornecer capacidade *plug and play* para conectar transdutores inteligentes em ambiente de rede;
- Possibilitar e simplificar a criação de redes de transdutores inteligentes;
- Facilitar o suporte para vários fornecedores.

De forma resumida, o padrão IEEE 1451.2 especifica a capacidade de *plug and play* de um dispositivo transdutor, ou seja, a capacidade de conectar e operar evitando assim dificuldades na reconfiguração do sistema toda vez que for conectado um novo dispositivo.

O módulo STIM e NCAP foi desenvolvido e implementado no Laboratório de Processamento de Sinais e Sistemas Digitais da Faculdade de Engenharia de Ilha Solteira e pode ser encontrado com maiores detalhes em (8) (9)(10).

Capítulo 3

Arquitetura de um celular

3.1. Introdução

Os aparelhos celulares possuem um dos processadores mais complexos disponíveis no mercado. Os processadores dos aparelhos celulares são altamente dedicados para efetuar cálculos e manipulações de sinais complexos, porém, de forma minituarizada. Eles apresentam alguns serviços e características semelhantes aos dos microcomputadores, como: acesso a web, caixas de som e microfones minituarizados, microprocessador, *display*, teclado para os comandos de entrada, ROM (*Random Only Memory*), memória *flash*, armazenamento de músicas, vídeos, e tantas outras diversas características.

Apesar dos microcomputadores e aparelhos celulares apresentarem algumas caracteristicas e serviços semelhantes, os dispositivos móveis apresentam inumeras restrições, como a capacidade de armazenamento de dados, largura de banda, consumo de energia, menor capacidade de processamento, entre outras.

Atualmente existe uma grande gama de dispositivos celulares disponíveis no mercado, com marcas, processadores e sistemas operacionais próprios. O poder de processamento e os sistemas operacionais destes dispositivos permitem desenvolver *software*s que podem ser executado sem diversos aparelhos.

Apresenta-se no Tópico 4.2 algumas características referentes aos processadores da família Intel.

3.2. Processadores da Família Intel

3.2.1. Processador PCA

A Intel PCA (*Personal Client Architecture*) foi concebida tendo em mente dispositivos portáteis capaz de se conectar a internet sem o uso de fios e cabos. Seus componentes se caracterizam pelo pequeno tamanho e baixíssimo consumo de energia.

Microprocessadores que utilizam esta arquitetura, contém, no mesmo encapsulamento, além do processador propriamente dito, subsistemas de comunicações e processamento de dados, como interface infravermelho, USB (*Universal Serial Bus*), *Bluetooth*, UART (*Universal Asynchronous Receiver/Transmitter*) com um modem completo e portas seriais convencionais, no mais tradicional estilo SoC ("*System-on-a-Chip*") (10).

O "coração" da PCA é o microprocessador *XScale*, o padrão da Intel para processadores de baixo consumo de energia, que pode operar em diferentes freqüências (na faixa de 133 MHz a 624 MHz) dependendo das necessidades do dispositivo. Além disso, a Intel PCA incorpora a tecnologia de gerenciamento de energia denominada "*SpeedStep*" que, de modo similar ao usado nos modernos microcomputadores portáteis ("*notebooks*"), na medida que varia a demanda de potência, muda o "modo de operação" do chip, alterando sua freqüência de operação e tensão de alimentação dinamicamente, saltando entre os cinco modos de operação disponíveis para minimizar o consumo de energia e aproveitar ao máximo a carga da bateria (10).

Dentre as famílias de processadores PCA da Intel, as seguintes se destacam:

- PXA800F;
- PXA27X.

3.2.2. Intel® PXA800F

O PXA800F é fabricado segundo a tecnologia de processo flash com a lógica de 0,13 µm. É um processador totalmente integrado, que encontra-se no núcleo do sistema para telefones celulares GSM/GPRS (*Global System for Communication/General Packet Radio Service*) atuais. Esse processador de alto desempenho e eficiente em energia, integra as tecnologias *Intel XScale*® com a memória *On-Chip Flash* e com a *Micro Signal Architecture*, oferecendo excelente desempenho em aplicativos de voz e de uso intenso de computação para telefones celulares. As características do Processador PXA800F (12) são:

- Solução completa GSM/GPRS Class 12;
- Núcleo Intel XScale de alto desempenho/baixo consumo de energia, com folga de sobra, líder de sua classe para aplicativos "*rich-data*";
- A memória *Intel On-Chip Flash* integrada e a SRAM (*Static Random Access Memory*) integrada oferecem armazenamento para a pilha de comunicação GSM/GPRS, RTOS (*Real Time Operating System*) e código do aplicativo para uma solução móvel em um único chip;

- Suporta um conjunto rico de recursos, incluindo displays coloridos, reconhecimento de voz, *Voice Memo Pad*, Bluetooth, decodificação MP3 e MPEG-4, WAP (*Wireless Application Protocol*), SMS (*Short Message System*), SEM (*Enhanced Message Service*), MMS (*Multimidia Message Service*), localização de posição, cliente USB, SD Card (*Secure Digital Card*), MMC Card (*MultiMediaCard*) e Sony Memory Stick e câmeras digitais.

O processador celular Intel PXA800F (PXA é o acrônimo de *Processor XScale Architecture*) é o primeiro produto a integrar completamente uma solução de banda básica GSM/GPRS em um único chip com um processador de aplicativo de alto desempenho e uma memória flash. Essa solução caracteriza-se por um processador baseado na tecnologia *Intel XScale* capaz de executar em até 312 MHz, além da *Intel Micro Signal Architecture*, um núcleo de processador de sinais *dual-MAC* que pode executar em até 104 MHz. A integração da memória *Intel On-Chip Flash* e da SRAM no processador resulta em um aumento significativo no desempenho do processamento e na redução do consumo de energia. Esse produto empacotado em um único chip é parte de uma solução de sistema que inclui uma plataforma de desenvolvimento para dispositivos móveis de voz/dados com quatro bandas e recursos completos (12).

3.2.3. Processadores Intel PXA27X

Estes processadores são destinados aos PDA´s e telefones celulares, que aceitam dados transportados por diferentes padrões de comunicação sem fio e alta taxa de transmissão (“banda larga”), WiFi, WiMax e Bluetooth, com capacidade de processamento suficiente para que os dispositivos com eles equipados possam ser usados para videoconferência com vídeo de qualidade compatível a de DVDs e jogos tridimensionais de alto desempenho. Trata-se da família Intel PXA27x, composta por quatro microprocessadores, PXA270, 271, 272 e 273, cujas diferenças restringem-se à quantidade de memória incorporada a cada chip e à largura do barramento externo (10).

A família PXA27x agregou novas funções que melhoraram significativamente o desempenho dos microprocessadores, algumas delas baseados nos processadores de micros de mesa. O exemplo mais ilustrativo é a incorporação da tecnologia Wireless MMX (“*MultiMídia eXtension”*), um subconjunto de instruções destinado a melhorar o desempenho de aplicativos multimídia. As principais características do Processador PXA27x são as seguintes (13):

- *Intel XScale® Technology* - Núcleo escalável de até 624 MHz;
- A plataforma Wireless de Intel®: A segurança em serviços tais como a inicialização, o armazenamento seguro da informação confidencial e o suporte para protocolos de segurança tais como VPN (*Virtual Private Networks*) e SSL (*Secure Socket Layer*);
- As instruções Wireless familiares da tecnologia de Intel® MMX projetaram alto desempenho multimídia, jogos 3-D e vídeo avançado;
- Intel® Quick Capture tecnologia suporta 4+ Megapixel, câmera que captura imagens digital, vídeo, pré-visualização em tempo real;
- A tecnologia Wireless de Intel SpeedStep® com cinco modalidades *low-power* pode mudar a freqüência e a tensão dinamicamente. O *software Wireless Intel SpeedStep Power Manager* permite gerência interna, inteligente da energia;
- Intel® Mobile Scalable Link permite até 416 Mb/s entre o link de comunicação e o processamento do aplicativo;
- Suporta um barramento de memória de 100 MHz a uma variação de memória 1.8 V, 2.5 V, 3.0 V e 3.3 V (13).

3.3. Tecnologia de Comunicação dos Dispositivos Embarcados

A crescente necessidade de maior mobilidade e as melhorias das tecnologias das redes sem-fio permite maior número de conexões rápidas e estáveis para os dispositivos móveis. Neste contexto pode-se citar diferentes tipos de tecnologia utilizada para a transferência de dados (14)(15)(16)(17)(18):

- *TDMA*: (*Time Division Multiple Access*, ou Acesso Múltiplo por Divisão do Tempo), ou seja, voz e dados passando por um único canal faz com que a conexão se torne mais lenta e cara para o usuário. O TDMA estabelece uma conexão de no máximo 9,6 Kb/s e para estabelecer esta conexão leva-se cerca de 8 segundos. Apesar das limitações do TDMA, essa tecnologia ainda atende a maior parte das necessidades dos usuários, oferece identificação de chamada, atendimento simultâneo de ligação, envio e recebimento de mensagens de texto, indicação de caixa postal e possibilidade de conferência a três.

- *CDMA*: (*Code Division Multiple Access*, ou Acesso Múltiplo por Divisão de Códigos), ou seja, espelhamento espectral. A tecnologia CDMA mantém uma conexão 100% ativa e a taxa de transferência pode chegar ao máximo a 256 Kb/s. A principal diferença da tecnologia CDMA é a velocidade de transmissão de dados e a cobrança por pacote, portanto, diversos serviços podem ser oferecidos, tais como: envio de documentos, imagens e sons com velocidade superior à da tecnologia TDMA.

A tecnologia CDMA foi escolhida pela ITU (União Internacional de Telecomunicações) como tecnologia base para uma das migrações previstas para a terceira geração de telefonia celular que permitirá transmissões, por exemplo, de vídeo *on-demand* com alta qualidade. Algumas de suas evoluções já disponíveis, como CDMA2000 e W-CDMA, já permitem que o telefone celular sirva de acesso rápido à internet.

- *GSM:* (*Global System for Mobile Communication*, ou Sistema Global de Comunicação), utiliza o protocolo de comunicação GPRS (*General Packet Radio Service*), tendo as mesmas características do CDMA. A tecnologia GSM é utilizada como padrão para telefonia celular digital na Europa desde 1992 e está presente na América do Sul desde 1998. Apesar disto, por se tratar de uma tecnologia ainda recente no Brasil, sua abrangência nacional ainda não tem o mesmo alcance das demais operadoras já instaladas, o que deve ser resolvido em pouco tempo (14).

Apresentam-se no próximo capítulo as linguagens de programação para os dispositivos móveis.

Capítulo 4

Linguagem de Programação para os Dispositivos Móveis

4.1. *Introdução*

Este capítulo tem como objetivo apresentar as principais linguagens de programação para os dispositivos embarcados (celulares e PDA´s) e algumas de suas características mais relevantes. O seu estudo permite realizar a escolha mais adequada para se desenvolver o sistema de acordo com as normas do padrão IEEE 1451 e que funcione em uma quantidade maior de dispositivos móveis. As linguagens analisadas neste capítulo foram, WAP, *.Net Compact Framework*, SuperWaba e J2ME.

Neste projeto utilizou-se a linguagem J2ME devido a grande porcentagem de dispositivos móveis oferecerem suporte a aplicações, além de estar presente em uma grande gama de diferentes aparelhos.

Um ponto importante que devemos salientar é o suporte oferecido pela *Sun Microsystem* disponibilizando *software*s para desenvolvimento de aplicações para os dispositivos móveis de forma gratuita e uma grande quantidade de listas de discussões disponíveis na internet. Por isso, de acordo com esse contexto e suas demais características a linguagem J2ME foi escolhida para este projeto.

4.2. WAP (Wireless Aplication Protocol)

O WAP (*Wireless Aplication Protocol*) é um protocolo de comunicação e um ambiente de aplicações para a distribuição de recursos de informação, serviços de telefonia avançados e acesso à internet a partir de dispositivos móveis (19).

Quando apresentada pela primeira vez, a expectativa para a tecnologia WAP era grande, pois todos esperavam navegar na internet com seus telefones, mas a realidade foi bastante diferente. Apesar da tecnologia WAP fornecer um ambiente comum para os

dispositivos móveis e seus protocolos baseados no protocolo de internet, não significa que o WAP foi projetado para suportar o conteúdo inteiro da internet. As páginas HTML (*Hyper Text Markup Language*) embora elaboradas com arquivos de multimídia, *frames*, cores e efeitos dinâmicos, quando são transportadas para páginas WAP perdem toda a sua característica, pois são apresentadas em tela de 5 linhas com 20 caracteres, além das limitações do próprio dispositivo móvel. Apesar da existência de *software*s para conversão de páginas para aplicações WAP, eles não são eficazes, pois não usam a configuração do dispositivo do usuário. Toda página é tratada por meio das mesmas regras de conversão internas. Os aplicativos WAP são, quase todos, desenvolvidos para usuários WAP (19). Na Figura 4.1 apresenta-se uma interface em linguagem WAP.

Figura 4.1 - Interface da linguagem WAP.

4.3. .NET Compact Framework

O *.NET Compact Framework* é a plataforma de desenvolvimento para *Smart Device.* Trata-se da iniciativa da *Microsoft .NET* desenvolvida por usuários experientes. O *.NET Compact Framework* trouxe ao mundo o código gerenciado e o XML[3] (*eXtensible Markup Language*) *Web Services* para *Smart Devices*. Tem como característica a execução com segurança, *download* de aplicações em dispositivos como PDA´s, telefones celulares e outros dispositivos (20).

Em função do *.NET Compact Framework* ser um sub-conjunto do *.NET Framework*, os programadores podem facilmente usar os conhecimentos de programação e os códigos existentes nos dispositivos, desktop e servidores. A Microsoft liberou no *Visual Studio .NET* 2003 o *Smart Device Application*, que contém o *.NET Compact Framework*. Isso significa que programadores de *Visual Studio* que tem experiência com o *.NET Framework* podem desenvolver aplicações para qualquer dispositivo que rode *.NET Compact Framework.* Devido a isso, muitos desenvolvedores *Visual Studio .NET* são programadores potenciais para *Smart Device* (20).

[3] XML – linguagem de programação criada pela W3C (World Wide Web Consortium), uma linguagem de marcação que combinasse a flexibilidade da SGML (Linguagem Padronizada de Marcação Generica) com a simplicidade da HTML.

Além disso, pelo fato de compartilhar as ferramentas e modelos de programação do *.NET Framework*, o *.NET Compact Framework* tem reduzido drasticamente o custo, e acrescido eficiência no desenvolvimento de aplicações para *Smart Devices* (20). Nos Tópicos 4.3.1, 4.3.2, 4.3.3, 4.3.4 e 4.3.5, apresentam-se as características das linguagens.

4.3.1. Código Compartilhado e Aumento da Eficiência

Como o *.NET Compact Framework* emprega o mesmo modelo de programação através dos aparelhos, o processo de desenvolvimento de aplicação funciona em diferentes aparelhos. Muitos dos códigos para uma aplicação, podem ser compartilhados através de diferentes aparelhos e computadores. Isto aumenta consideravelmente a eficiência no desenvolvimento de aplicações (20).

4.3.2. Características Enterprise-Class para Capacitar mais Aparelhos

O sucesso dos aparelhos *Microsoft Pocket PC* está crescendo, em parte porque eles tem recursos computacionais necessários para lidar com sofisticadas aplicações de negócios. O *.NET Compact Framework* tem vantagens para a plataforma *Pocket PC* por prover avançados sistemas que simplificam o processo de desenvolvimento de aplicações para dispositivos *Pocket PC* (20).

4.3.3. Código Robusto, Execução Segura

O *.NET Compact Framework* oferece um ambiente robusto e seguro para executar código no lado do cliente. O modelo de código gerenciado (*managed code*) suportado pelo *.NET Compact Framework* aumenta a confiabilidade do código, assim reduz os defeitos no *software*. A execução do código gerenciado assegura que o comportamento da aplicação não será capaz de travar o aparelho. Ao mesmo tempo, o modelo de segurança através do *.NET Compact Framework* assegura que códigos maliciosos não permitirão obter acesso para os recursos de segurança do sistema. O modelo de segurança também permite atualizar aplicações para serem distribuídas pela rede sem fio dentro de um caminho seguro. (20).

4.3.4. Amplo Suporte para Aplicações Off-line

Com o código seguro no cliente, o *.NET Compact Framework* capacita aplicações que podem ser executadas *Off-line*. Isto permite uma experiência tranqüila em relação a qualquer problema com conexão persistente que possa ocorrer quando se acessa uma rede. O desenvolvedor pode escolher a combinação certa de programação do lado cliente ou servidor para interagir com a experiência do usuário (20).

4.3.5. Custos Reduzidos de Desenvolvimento na Criação de Oportunidades

Esta novidade de desenvolvimento para o *.NET Compact Framework* diz a ampla criação de aplicações e serviços que ajudam empresas a vencerem grandes mercados para os dispositivos móveis e criar novas oportunidades para desenvolvedores. Muitos negócios que não foram realizados por uma questão de custo ou de treinamento especial de desenvolvedores para aplicações móveis serão capaz de desenvolver novas aplicações móveis muito mais eficientes, com o objetivo de diminuir os custos, e aumentar as oportunidades (20).

O *.NET Compact Framework* é parte chave da Microsoft para oferecer suporte aos clientes com grandes experiências. Isso porque cada desenvolvedor *Visual Studio .NET* poderá ser um desenvolver *Smart Device*, haverá exponencialmente mais desenvolvedores para o *.NET Compact Framework* que qualquer outro dispositivo ou plataforma de programação móvel. O mais importante, do que *.NET Compact Framework* reduzirá os custos drasticamente, e aumentará a eficiência, o desenvolvimento de aplicações para *Smart Devices*. Isto irá com o tempo, ajudar as companhias reduzir os custos dos negócios tornando-as capazes para desenvolver novas aplicações móveis que aumentarão a eficiência dos empregados e os capacitarão para novas oportunidades de negócios. Na Figura 4.2 apresenta-se a interface gerada pela linguagem de programação *.NET Compact Framework* (20).

Figura 4.2 - Interface da linguagem .Net Compact Framework.

4.4. SuperWaba

O SuperWaba, foi criado a partir do projeto intitulado Waba. Este projeto pertencia a um americano Rick Wild, que o começou em 1999, porém pouco tempo depois foi descontinuado. O Brasileiro Guilherme Campos Hazan, em busca de novas tecnologias para desenvolvimento, o conheceu, e a partir daí, com a devida licença de seu autor, começou a desenvolver novas funcionalidades, e a melhorar alguns aspectos que o projeto Waba tinha. A partir deste momento, o projeto mudou de nome, para o conhecido SuperWaba. Além do nome, o tipo de licença do antigo Waba foi também mudada. O projeto estava sobre licença BSD[4] (*Berkeley Software Distribution*), e hoje está sobre a licença LGPL[5] (Licença Pública Geral Menor), o que permite que usuários da plataforma fechem seus códigos e vendam seus programas comercialmente (21).

O SuperWaba pode ser usado em qualquer IDE (*Integrated Development Environment*) utilizada para o desenvolvimento Java. É constituída de uma VM (*Virtual Machine*) e diversas API's (Interface de Programação de Aplicativos) extras, e implementa os bytecodes do Java, por ser compilada no próprio compilador Java. Como Java é uma marca registrada da empresa *Sun MicroSystems*, ela não pode ser designada

[4] BSD - os créditos dos autores originais devem ser mantidos, mas não estabelece outras limitações para o uso do código.
[5] LGPL - é uma licença de software livre aprovada pela FSF (Fundação para o Software Livre) escrita com o intuito de ser um meio-termo entre a GPL (Licença Pública Geral) e licenças mais permissivas como a licença BSD e a licença MIT (*Massachusetts Institute of Technology*).

para o SuperWaba, o qual não tem nenhuma ligação com a *Sun MicroSystems*, ou seja, SuperWaba não é Java (21).

O SuperWaba pode ser utilizado em diversas aplicações em se tratando de PDA's, proporciona uma API ampla, trazendo componentes gráficos e os acessos de *hardware* que o J2ME (*Java 2 Micro Edition*), isso por ser extremamente direcionada para PDA's (21).

SuperWaba não é utilizado em aplicações direcionada a celulares. O J2ME proporciona uma independência muito grande para celulares, sendo que seus próprios fabricantes desenvolvem as VM's para cada dispositivo. As VM's do SuperWaba são desenvolvidas e distribuídas juntamente com seu SDK (*Software Development Kit*), e por isso há um pouco de restrição quanto a quantidade de aparelhos que podem funcionar (21). Na Figura 4.3 apresenta-se a interface gerada pela linguagem de programação SuperWaba.

Figura 4.3 - Interface da linguagem SuperWaba no PDA.

4.5. J2ME (Java 2 Micro Edition)

Antigamente os dispositivos celulares e PDA´s não apresentavam a opção de "*download"* de aplicativos ou instalação de *software* pois já viam configurados no processo de fabricação. Hoje em dia os equipamentos não têm mais essa natureza estática, sendo que um aparelho contendo a JVM (*Java Virtual Machine*) é possível fazer *download*s e instalar aplicativos Java.

A plataforma J2ME foi projetada para aparelhos com memória, vídeo e poder de processamento limitado. A J2ME disponibiliza o poder e os benefícios da tecnologia

Java ao consumidor e aos dispositivos embarcados. Incluem relações de usuário flexíveis, um modelo de segurança robusta, uma larga escala de protocolos de rede internos e a sustentação extensiva para as aplicações network.

As aplicações baseadas em especificações de J2ME são escritas uma vez para uma escala larga dos dispositivos, contudo exploram potencialidades nativas de cada dispositivo. Pensando neste foco, a Sun introduziu os conceitos de configuração, que define uma plataforma Java para uma ampla variedade de dispositivos. O perfil adiciona capacidades especificas a uma determinada configuração e está mais orientada ao desenvolvimento de aplicações específicas para um dispositivo (22)(23). Na Figura 4.4 apresenta-se o aplicativo desenvolvido em J2ME sendo executado.

Figura 4.4 - Software desenvolvido utilizando a linguagem J2ME

4.5.1. Configurações da J2ME

A linguagem J2ME define os dispositivos através de dois tipos de configuração, o CDC (Configuração de Dispositivos Conectados) e o CLDC (Configuração dos Dispositivos Conectados Limitados), que possuem as seguintes características (22):

CDC:

- 512 kbytes de memória necessária para executar o Java;
- 256 kbytes alocação de memória para tempo de execução;
- Rede e largura de banda persistente e alta.

CLDC:

- 128 kbytes de memória necessária para executar o Java;
- 32 kbytes para alocação de memória;
- Interface restrita;
- Baixa potência, alimentado por bateria;
- Rede e largura de banda baixa e acesso intermitente (22).

Na Figura 4.5 apresenta-se a relação entre o J2SE, plataforma voltada para os microcomputadores, e as configurações dos dispositivos móveis CDC e CLDC direcionada para os dispositivos móveis.

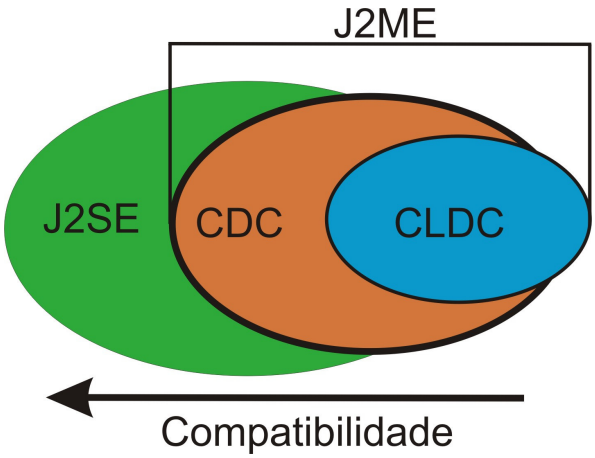

Figura 4.5 - Relação entre as configurações.

4.5.2. Edições do Java e suas Máquinas Virtuais

A Java atualmente trabalha com três edições distintas, sendo que cada edição voltada para um tipo de tecnologia (22):

- J2EE (*Java 2 Entreprise Edition*) – o J2EE trabalha com tecnologia baseada em servidores, visando o ramo empresarial. O J2EE disponibiliza suporte interno para servlets, JSP (*Java Server Pages*) e XML;
- J2SE (*Java 2 Standart Edition*) – Para as aplicações mais comuns, sendo executado em máquinas simples de computadores pessoais e estações de trabalho;
- J2ME – Voltado para dispositivos embarcados de pequeno porte, com memória, vídeo e poder de processamento limitado.

Na Figura 4.6 apresenta-se às máquinas virtuais e suas respectivas edições Java.

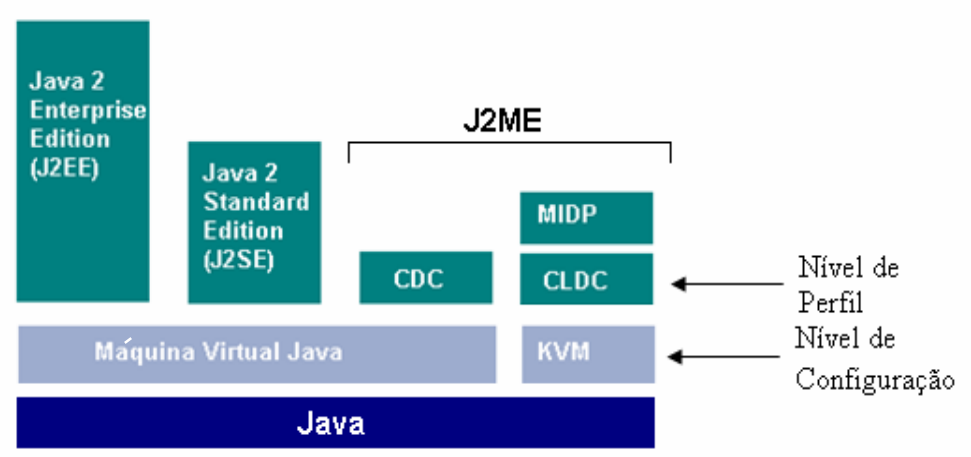

Figura 4.6 - Edições do Java e suas respectivas máquinas virtuais.

A KVM (*Kilo Virtual Machine*) é uma implementação da VM otimizada para ser utilizado em dispositivos limitados e foi desenvolvida para ser facilmente portável. Deve-se salientar que a máquina virtual clássica pode trabalhar com os dispositivos embarcados, sendo que haverá aumento de espaço ocupado. As classes J2ME foram alteradas para que possam ser usadas nos dispositivos e muitas outras foram retiradas, pois não faziam sentido neste tipo de dispositivo (22). Na Figura 4.7 apresenta-se a arquitetura da linguagem J2ME.

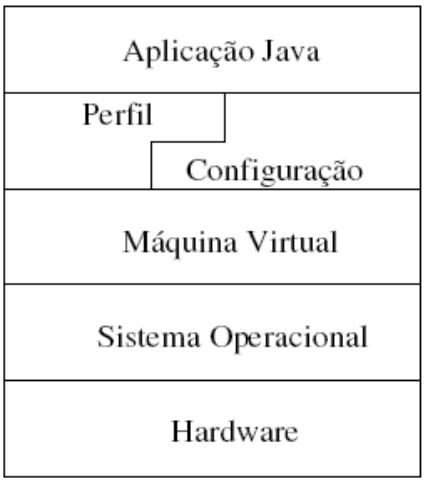

Figura 4.7 - Arquitetura J2ME.

O KVM é uma máquina virtual compacta e portátil, desenhada para dispositivos com poucos recursos. A KVM foi projetada para ser pequeno, para uso de memória estática entre 40 e 80 Kb, portável, modular e tão completo e eficiente quanto possível. A KVM está implementada em ANSI C e logo pode ser facilmente trabalhada em outras plataformas em que exista um compilador C (22).

4.5.3. MIDP

A MIDP (*Mobile Information Device Profile*) define as API´s para os componentes entrada e tratamento de eventos de interface com o usuário, armazenamento persistente, interligação em rede e cronômetros, levando em consideração as limitações de tela e memória dos dispositivos móveis. Uma aplicação MIDP pode receber o nome de MIDlet que são similares aos Applets. Pode-se definir um applet como um *software* que pode ser instalado e executado em páginas Web. A MIDP foi elaborada para trabalhar com os dispositivos CLDC e requer algumas especificações mínimas de *hardware*. As especificações podem ser definidas a seguir (23):

Visor:

- Tamanho da tela: 96x54;
- Formato da tela (proporção de aspecto): 1:1.

Entrada:

- uma mão, ou;
- duas mãos, ou;
- tela de toque.

Memória:

- 128 Kbytes para os componentes MIDP;
- 8 Kbytes para dados das aplicações;
- 32 Kbytes para o tempo de execução JAVA.

Rede:

- Duplex, sem fio, possivelmente intermitente e com largura de banda limitada.

Por possuir uma grande variedade de *software*, a MIDP estabeleceu alguns requisitos mínimos de sistema (23):

- Um kernel para controlar o *hardware*, que possua uma entidade escalonável para rodar a Máquina Virtual Java;
- Um mecanismo para ler e escrever na memória para suportar as APIs;
- Acesso de leitura e escrita à rede sem fio;
- Capacidade de escrever num display *bit-mapped*;
- Um mecanismo para capturar entrada de um *input device* .

4.5.4. Arquitetura de uma MID

Para definirmos a arquitetura de uma MID, primeiramente iniciamos pelo *hardware*, que é o nível mais baixo que pode ser visualizado na Figura 4.8, um nível acima encontra-se o sistema operacional nativo. Após, vem o CLDC onde se encontra a máquina virtual K (KVM). A KVM permite que as APIs Java de alto nível sejam construídas. Em cima do CLDC, rodam as APIs MIDP que podem ser visualizadas na Figura 4.8. As classes OEM (*Original Equipment Manufacturer*, Fabricante de Equipamento Original) são definidas pela MIDP e podem ser utilizadas para funcionamento específico para determinados aparelhos, o que significa que eles podem ou não ser portáveis para outras MIDs. Uma aplicação OEM são aquelas que não fazem parte da especificação (22).

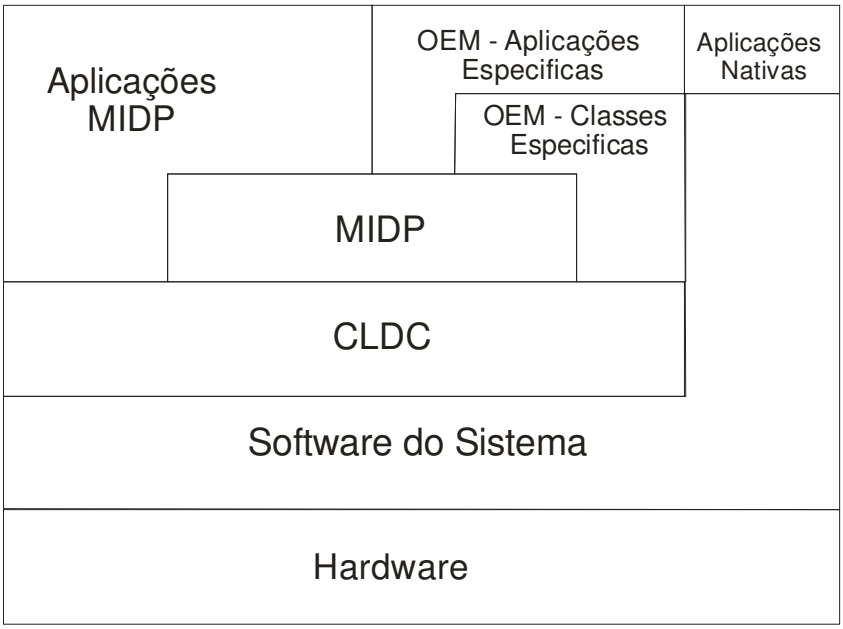

Figura 4.8 - Arquitetura do Dispositivo de Informação móvel.

Um aplicativo MIDlet é aquele que usa APIs definidas pelo MIDP e CLDC, e são portáveis entre vários aparelhos. Os Aplicativos nativos são os que estão implementados diretamente no sistema e não são escritos em Java (22).

4.5.5. Aplicações

O MIDP determina um modelo de aplicação que permite que os recursos limitados dos MIDs sejam compartilhados por várias aplicações, as MIDlets. Este compartilhamento é viável mesmo com os limitados recursos e *framework* de segurança

do MID, pois eles são obrigados a compartilhar classes e estão sujeitos a um conjunto de políticas e controles (23).

Os elementos de uma MIDlet *suite*, dos quais se espera que implementem as funções necessárias pelos usuários para instalar, selecionar, rodar e remover midlets, são:

- *Ambiente de Execução*: é compartilhado por todas as MIDlets que estão na mesma MIDlet suite, e qualquer MIDlet pode interagir com outra que esteja no mesmo pacote (23).
- *Empacotamento do MIDlet suite:* uma ou mais MIDlets podem ser empacotadas num único arquivo JAR (*Java Archive*), que contém as classes compartilhadas e os arquivos de recursos utilizados pelas MIDlets, além de seu conteúdo. Existem vários atributos pré-definidos que permitem identificação de uma MIDlet, como nome, versão, tamanho de dados, descrição, etc (23).
- *Descritor de Aplicação:* é utilizado para gerenciar a MIDlet e é usada pela própria MIDlet para atributos de configuração específica. O descritor permite que seja verificado que a MIDlet é adequada ao aparelho antes de carregar todo o arquivo JAR da MIDlet *suite*. Ele também permite que parâmetros sejam passados para as MIDlets sem modificar os arquivos JAR (23).
- *Ciclo de Vida da Aplicação:* uma MIDlet não deve possuir um método *public void static main().* O *software* de gerenciamento de aplicação deve suprir a classe inicial necessária pelo CLDC para iniciar a MIDlet. Quando uma MIDlet é instalada, ela é mantida no aparelho e fica pronta para uso. Quando é executada, uma instância é criada através de seu construtor público sem argumentos, e da MIDlet, ou seja, entra no estado **Ativo** através do método *startApp()*. Quando ela é terminada, é destruída, e os recursos utilizados podem ser recuperados, entra no estado **Destruído** *destroyApp()*, incluindo os objetos criados e suas classes. Quando ocorre uma chamada, devido sua prioridade ser maior, a MIDlet entra no estado de **Pausado** *PauseApp()*, quando é finalizada a chamada a MIDlet volta para o modo **Ativo**. Na Figura 4.9 apresenta-se o ciclo de vida da MIDlet (23).

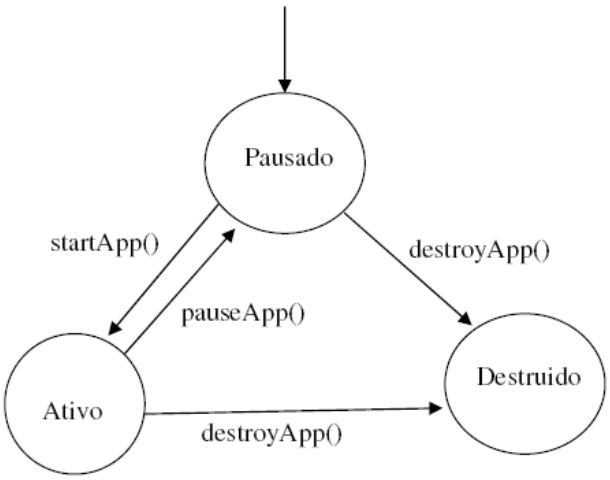

Figura 4.9 - Ciclo de Vida da MIDlet.

4.5.6. Pacotes opcionais

A plataforma de J2ME pode ser estendida adicionando vários pacotes opcionais a uma pilha da tecnologia que inclua CLDC ou CDC e um perfil associado. Criado para dirigir-se a exigências muito específicas das aplicações, os pacotes opcionais oferecem APIs padrão para usar tecnologias existentes e emergentes tais como o conectividade da base de dados, mensagens wireless, os multimedia, o Bluetooth, e os serviços da foto receptora. Os pacotes opcionais são modulares, os colaboradores podem evitar de carregar as despesas gerais da funcionalidade desnecessária incluindo somente os pacotes que uma aplicação realmente necessita (23).

Capítulo 5

Linguagem Java, NetBeans, NetBeans MobilityPack 5.5, Slackware 10, Tomcat

5.1. Introdução

No presente capítulo, são apresentadas as principais ferramentas utilizadas para o desenvolvimento do *software* MATIUDE e para o desenvolvimento da parte lógica do NCAP. As ferramentas utilizadas neste projeto, foram obtidas de forma gratuita e oferecem o suporte necessário para o desenvolvimento do sistema além de uma vasta documentação. As ferramentas descritas possuem as caracteristicas ressaltadas pela norma do padrão IEEE 1451.

5.2. Linguagem de Programação Java

Para o desenvolvimento da programação da parte lógica do NCAP é necessária uma linguagem que atinja os requisitos exigidos pelo sistema, neste contexto, optou-se pela linguagem Java. Java é uma linguagem de programação orientada a objetos desenvolvida pela *Sun MicroSystems*. Modelada depois de C++, a linguagem Java foi projetada para ser pequena, simples e portável a todas as plataformas e sistemas operacionais, tanto o código fonte como os binários (24).

Os arquivos do Java são compilados e convertidos de arquivos texto para um formato que contém blocos independentes de *bytes codes*.

Em tempo de execução estes *bytes codes* são carregados e verificados através do *Byte Code Verifier* (uma espécie de segurança) e passam a seguir para o interpretador para serem executados. Caso este código seja acionado diversas vezes, existe um passo chamado *JIT Code Generator*, que elimina a utilização por demasia do tráfego da rede (24).

Java tem a aparência de C ou de C++, embora a filosofia da linguagem seja diferente. Java também possui características herdadas de muitas outras linguagens de programação: *Objective-C*, *Smalltalk*, *Eiffel*, *Modula-3*, etc. Muitas das características desta linguagem não são totalmente novas. Java é uma feliz união de tecnologias testadas por vários centros de pesquisa e desenvolvimento de *software* (24).

Um programa em Java é compilado para o chamado "*byte-code*", que é próximo às instruções de máquina, mas não de uma máquina real. O "*byte-code*" é um código de uma máquina virtual idealizada pelos criadores da linguagem. Por isso Java pode ser mais rápida do que se fosse simplesmente interpretada (24).

Java foi criada para ser portável. O "*byte-code*" gerado pelo compilador para a sua aplicação específica pode ser transportado entre plataformas distintas que suportam Java (Solaris 2.3®, Windows-NT®, Windows-95®, Mac/Os etc). Não é necessário recompilar um programa para que rode numa máquina e sistema diferente, ao contrário do que acontece, por exemplo, com programas escritos em C e outras linguagens. Esta portabilidade é importante para a criação de aplicações para a heterogênea internet (24).

A portabilidade é uma das características que se inclui nos objetivos almejados por uma linguagem orientada a objetos. Em Java ela foi obtida de maneira inovadora com relação ao grupo atual de linguagens orientadas a objetos (24).

Java suporta herança, mas não herança múltipla. A ausência de herança múltipla pode ser compensada pelo uso de herança e interfaces, onde uma classe herda o comportamento de sua superclasse além de oferecer uma implementação para uma ou mais interfaces (24).

Java permite a criação de classes abstratas. Outra característica importante em linguagens orientadas a objetos é a segurança. A presença de coleta automática de lixo, evita erros comuns que os programadores cometem quando são obrigados a gerenciar diretamente a memória (C, C++, Pascal). A eliminação do uso de ponteiros, em favor do uso de vetores, objetos e outras estruturas substitutivas trazem benefícios em termos de segurança. O programador é proibido de obter acesso a memória que não pertence ao seu programa, além de não ter chances de cometer erros e uso indevido de aritmética de ponteiros. Estas medidas são particularmente úteis quando pensarmos em aplicações comerciais desenvolvidas para a internet (24).

A presença de mecanismos de tratamento de exceções torna as aplicações mais robustas, não permitindo que elas abortem, mesmo quando rodando sob condições anormais. O tratamento de exceções será útil na segunda parte deste trabalho para

modelar situações tais como falhas de transmissão e formatos incompatíveis de arquivos (24).

A linguagem permite a criação de maneira fácil, de vários "*threads*" de execução. Os *threads* são fluxos de execução que rodam dentro de um processo (aplicação). Normalmente as *threads* compartilham regiões de memória, mas não necessariamente. Process[6]os, os avós dos *threads* permitem que o seu sistema operacional execute mais de uma aplicação ao mesmo tempo enquanto que *threads* permitem que sua aplicação execute mais de um método ao mesmo tempo. E é particularmente poderoso nos ambientes em que aplicações Java são suportadas, ambientes estes que geralmente podem mapear os *thread*s da linguagem em processamento paralelo real (24).

Na Figura 5.1 apresenta-se o fluxo de execução da thread (24).

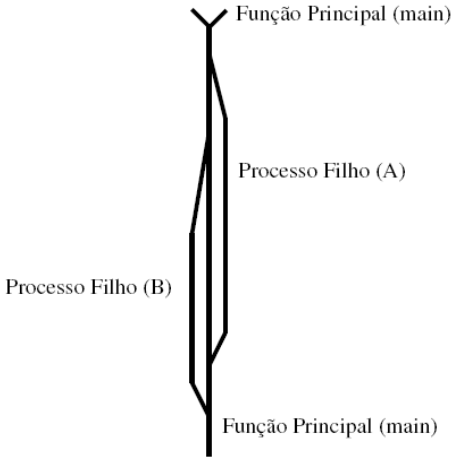

Figura 5.1 - Fluxo de execução da thread.

Como Java foi criada para ser usada em computadores pequenos, ela exige pouco espaço, pouca memória. Java é muito mais eficiente que grande parte das linguagens de "*scripting*" existentes, embora seja cerca de 20 vezes mais lenta que C, o que não é um marco definitivo. Com a evolução da linguagem, serão criados geradores de "*byte-codes*" cada vez mais otimizados que trarão as marcas de desempenho da linguagem mais próximas das de C++ e C. Além disso, um dia Java permitirá a possibilidade de gerar código executável de uma particular arquitetura, tudo a partir do "*byte-code*" (24).

[6] Processo – é a execução de um programa uma entidade fundamental que necessita de recursos para poder realizar sua tarefa de executar o programa.

5.2.1. Suporte para Programação de Sistemas Distribuídos

Java fornece facilidades para programação com *Socket*s, *remote method call*, TCP/IP, etc. Podemos destacar um dos processos mais utilizados para a comunicação entre *software*s que é *Socket*. Existem dois modos de sua utilização (24):

- Orientado a conexão – trabalha sobre o protocolo TCP (*Transmission Control Protocol* – Protocolo de Controle de Transmissão):
  - Vantagens da conexão orientada (TCP):
  - Os pacotes enviados através do protocolo TCP podem sofrer perdas. Quando há perda, esses pacotes são retransmitidos, mantendo assim a integridade do documento enviado;
  - O protocolo TCP garante a ordenação dos pacotes;
  - Permite a utilização de fluxo de dados.
- Desvantagens da conexão orientada (TCP):
  - É mais lento do que o modo orientado a datagrama, devido ao processo de verificação de pacotes;
  - O servidor se comporta de forma diferente do cliente.
- Orientado a datagrama – trabalha sobre o protocolo UDP (*User Datagram Protocol* – Protocolo Datagrama de Usuário):
  - Vantagem orientada a datagrama:
  - Mais rápida do que a orientada a datagrama.
- Desvantagens orientadas a datagrama:
  - Devido a não verificação o seu serviço não é confiável podendo haver perdas de pacotes;
  - Não garante a ordenação de pacotes.

Os dois protocolos (TCP e UDP) trabalham sobre o protocolo IP (*internet Protocol*). O protocolo IP define onde os pacotes deverão percorrer (24).

5.3. O NetBeans

Neste tópico, apresenta-se a IDE (*Integrated Development Environment*) de desenvolvimento de *software* em Java e o pacote de desenvolvimento de *software* para os dispositivos móveis.

NetBeans é um projeto *open-source* de sucesso, com uma grande base de usuários, uma crescente comunidade e parceiros mundiais. *Sun MicroSystems* fundou o projeto NetBeans em junho de 2000 e continua sendo seu principal patrocinador. Hoje, existem dois produtos: a IDE NetBeans e a Plataforma NetBeans (25).

A NetBeans IDE é um ambiente de desenvolvimento, uma ferramenta para programadores que permite escrever, compilar, depurar e instalar programas. A IDE é completamente escrita em Java, mas pode suportar qualquer linguagem de programação. Existem também um grande número de módulos para extender a IDE NetBeans. A NetBeans IDE é um produto livre, sem restrições de como ele pode ser usado (25).

Também está disponível a NetBeans Platform; uma base modular e extensível que pode ser usada como infra-estrutura para se criar grandes aplicações de *desktop*. Parceiros fornecem *plug-ins* que podem ser facilmente integrados na plataforma, e que podem ser utilizados para desenvolver ferramentas e soluções próprias. Ambos os produtos são *open source* e livres para uso comercial e não comercial. O código fonte está disponível para ser usado através da licença CDDL (*Common Development and Distribution License*) (25). Na Figura 5.2 apresenta-se a interface gráfica do NetBeans IDE.

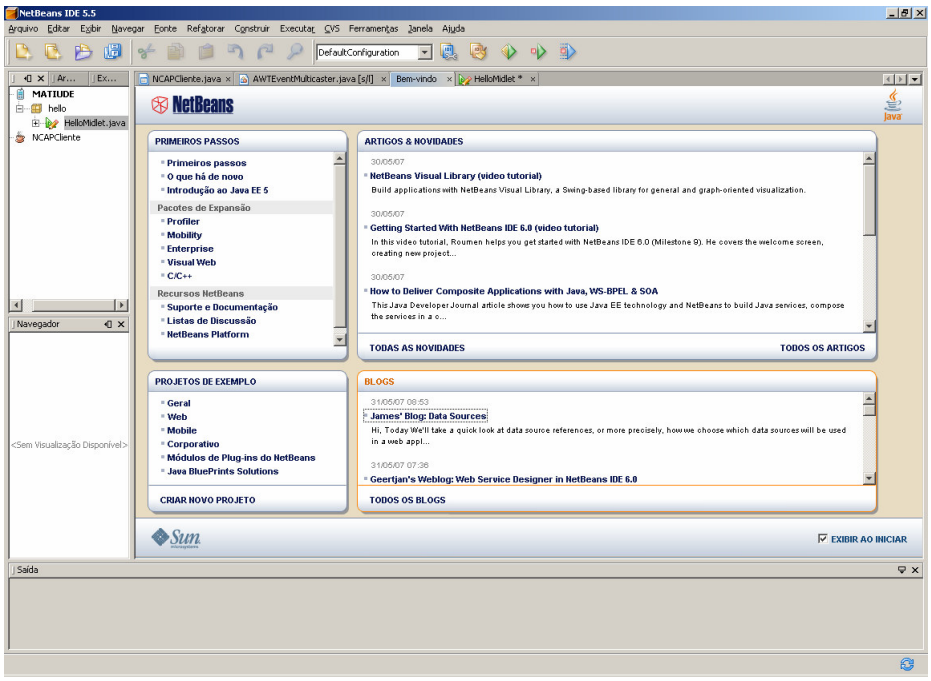

Figura 5.2 - Interface gráfica do NetBeans IDE.

5.4. Netbeans Mobility Pack 5.5

O *Netbeans Mobility Pack 5.5* é um pacote suplementar que oferece ferramentas para criar aplicativos do Java 2 Micro Edition compatíveis com as tecnologias e Configuração de Dispositivo Limitados Conectados (CLDC) e Perfil de Dispositivo de Informação Móvel (MIDP), fornecendo um ambiente de programação similar para aparelhos móveis, facilitando a visualização espacial e de distribuição de componentes, essas ferramentas formam nossa IDE de desenvolvimento (26).

O *Netbeans Mobility Pack 5.5* é executado no J2SE JDK 5.0 (Java 2 JDK, edição padrão), composto pelo Java Runtime Environment e por ferramentas de desenvolvimento para compilar, depurar e executar aplicativos escritos na linguagem Java. O *Netbeans Mobility Pack 5.5* é executado em sistemas operacionais que oferecem suporte à máquina virtual Java como o Microsoft Windows 2000 Professional SP4 (*Service Pack 4*), Microsoft Windows XP Professional SP2 (*Service Pack 2*) e Red Hat Fedora Core 3 (26). Na Figura 5.3 apresenta-se a interface de desenvolvimento do NetBeans para os dispositivos embarcados.

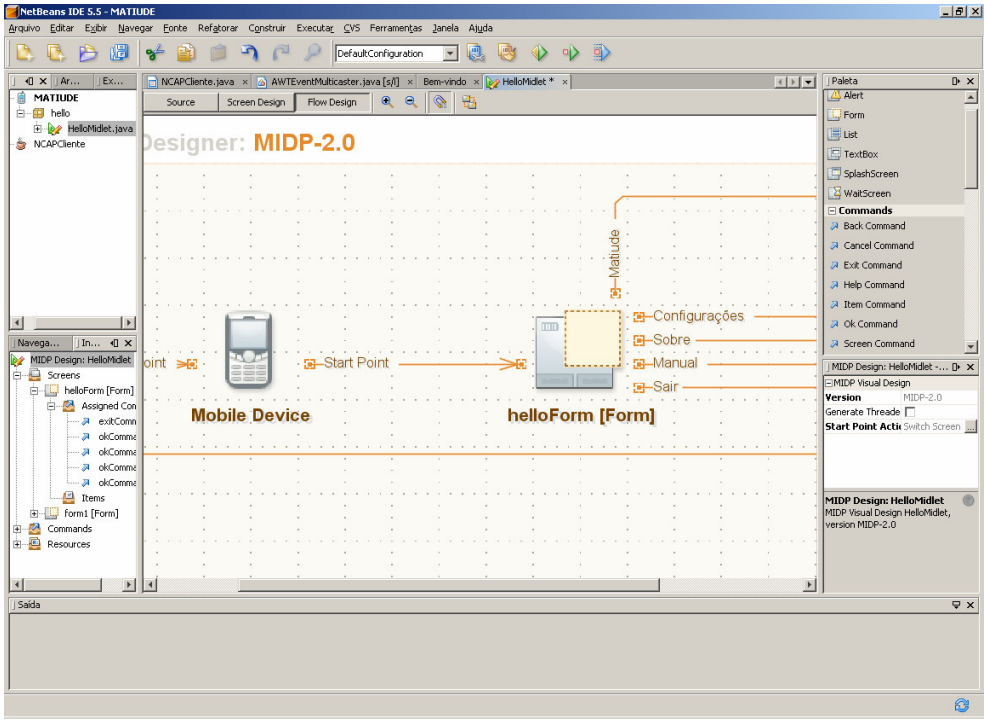

Figura 5.3 - Interface de desenvolvimento do Netbeans Mobility Pack 5.0.

5.5. Sistema Operacional Slackware 10

Para o desenvolvimento de um servidor seguro e robusto é necessário um sistema operacional que seja capaz de realizar tais configurações para o nosso projeto especificamente e que siga as normas do padrão IEEE 1451. Neste projeto o sistema operacional utilizado foi o Slackware 10. O Slackware 10 apresenta várias vantagens e suas características são bastante atraentes ao administrador, como na facilidade de administração, segurança e estabilidade. No caso dos servidores pode-se configurar os *software*s em modo texto, o acesso remoto faz com que o usuário se torne independente de sua localização, a simplicidade dos arquivos de inicialização e configuração tornam a reconfiguração da máquina rápida e sem complicações ao usuário. Outra vantagem é base de dados dos pacotes, que é toda em modo texto, o que facilita muito a busca de arquivos, pacotes instalados e também a confecção de utilitários que usem essa base para propagação de instalações, facilita *upgrades*, ou mesmo para localizar em qual pacote se encontra determinado arquivo. Isso apenas com comandos comuns utilizados na *shell*, facilitando a tarefa de recuperar um sistema danificado ou saber quais pacotes reparar (27).

A distribuição Slackware 10 é gratuita, *open-source* e pode ser obtido através de site ou do FTP (*File Transfer Protocol*) disponível na internet.

5.6. O Servidor Tomcat

O Tomcat é um servidor de aplicações Java para web. É um *software* livre e de código aberto criado dentro do conceituado projeto Apache Jakarta e oficialmente endossado pela Sun como a RI (Implementação de Referência) para as tecnologias JSP (*Java Servlet e JavaServer Pages*). Atualmente, o Tomcat tem seu próprio projeto dentro da Apache *Software* Foundation. O Tomcat é robusto e eficiente o suficiente para ser utilizado mesmo em um ambiente de produção (28).

Tecnicamente, o Tomcat é um *Container Web*, parte da plataforma corporativa Java Enterprise Edition (J2EE ou Java EE) que abrange as tecnologias Servlet e JSP, incluindo tecnologias de apoio relacionadas como *Realms* e segurança, *JNDI Resources* e *JDBC DataSources*. O Tomcat tem a capacidade de atuar também como servidor web/HTTP, ou pode funcionar integrado a um servidor web dedicado como o Apache httpd ou o Microsoft IIS (28).

5.7. Os Servlets

Os Servlets são classes Java que são instanciadas e executadas em associação com servidores web, atendendo as requisições realizadas por meio do protocolo http. Ao serem acionadas, os objetos Servlets podem enviar a resposta na forma de uma página HTML. Servlets podem assumir vários tipos de servidores, não só web. Os Servlets são tipicamente usados no desenvolvimento de sites dinâmicos, não possuem interface gráfica e são executadas dentro do ambiente Java denominado de *Container*. O *Container* gerencia as instâncias dos Servlets e provê os serviços de rede necessários para as requisições e resposta (29).

Capítulo 6

Desenvolvimento do Sistema MATIUDE

6.1. Introdução

Nesta dissertação foi desenvolvida um sistema chamado MATIUDE visualizando os dispositivos móveis. O MATIUDE é capaz de interagir com a parte lógica do NCAP desenvolvido em um microcomputador utilizando a linguagem Java, assim, realizar o monitoramento e o acesso aos transdutores inteligentes utilizando a porta paralela do microcomputador. Na Figura 6.1, ilustra-se o modelo do sistema empregando, os módulos desenvolvidos (NCAP e módulo STIM) e como pode ser realizado o acesso para o monitoramento e controle dos transdutores.

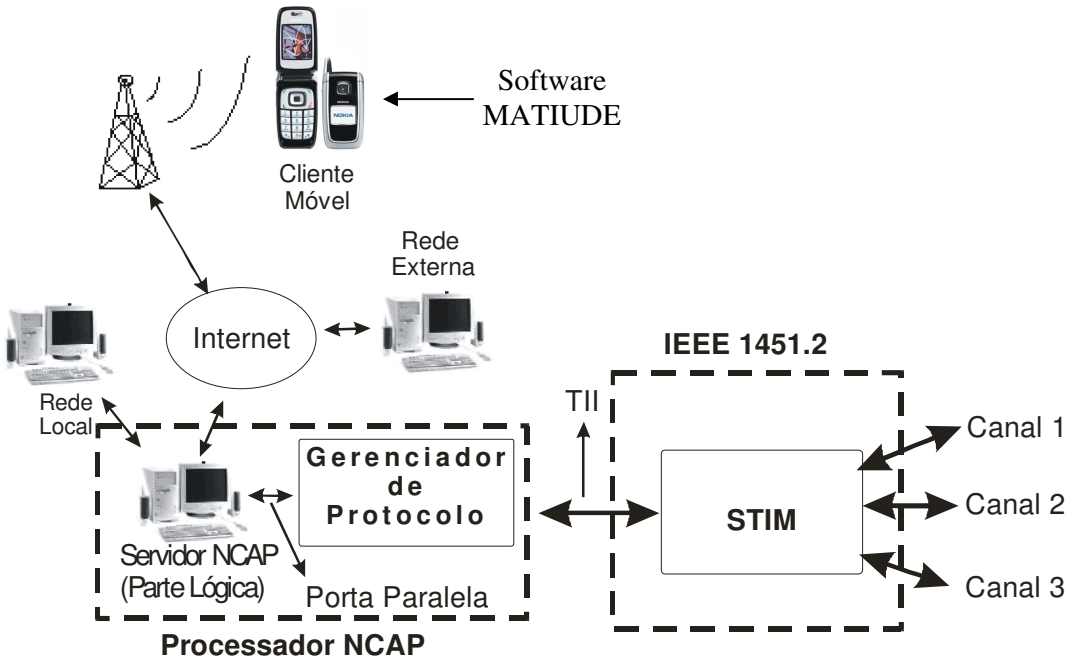

Figura 6.1 - Modelo do projeto.

O acesso ao NCAP pode ser realizado através de uma rede local, internet ou através dos dispositivos embarcados. As diferentes formas de acesso aos transdutores inteligentes apresentarão o mesmo resultado para o usuário.

6.2. Módulo STIM

O módulo STIM pode ser composto por um ou vários transdutores, circuitos condicionamento de sinal, conversores A/D (analógico/digital) e D/A (digital/analógico) necessário para realizar o interfaceamento dos transdutores com o processador local, um dispositivo de memória não volátil para armazenar os formatos TEDS, acessível através do processador local, e a lógica necessária para implementar a TII. A TII é uma interface composta por 10 fios com protocolo serial de transmissão de dados. Cada módulo STIM pode conter ate 255 transdutores de acordo com o padrão IEEE 1451

Para a realização dos testes do módulo STIM, utilizaram-se três ambientes distintos, descrito a seguir:

- Controle de temperatura em ambientes fechados;
- Controle da vazão em sistema de distribuição de água;
- Monitoramento da temperatura ambiente.

Uma especificação detalhada do módulo STIM desenvolvido pode ser encontrado em (8)(9)(10).

6.2.1. Controle de Temperatura em Ambientes Fechados

Os níveis de stress em aves e em animais criados em cativeiros estão diretamente relacionados à temperatura local e outras condições ambientais. Cada vez mais produtores investem em tecnologia para minimizar ou até mesmo eliminar os efeitos que as variações das condições ambientais provocam no plantel. As soluções encontradas para solucionar os problemas são, geralmente, de custo muito elevado e específico para algumas situações pré-estabelecidas. Dessa forma a automação sem nenhuma supervisão humana, apesar de eficiente, pode não prever algumas situações e por em risco a sobrevivência das aves e dos animais.

Neste contexto, desenvolveu-se uma maquete para realização dos estudos, contendo, *coolers*, lâmpadas como atuadores e um sensor de temperatura Dallas DS1821. Apesar da utilização de apenas 3 canais no gerenciador de protocolo, dependendo do ambiente, cabe salientar que outros 252 canais podem ser utilizados para o monitoramento ou controle. Na Figura 6.2 apresenta-se a foto da maquete da granja montada no laboratório para realização do controle da temperatura.

Figura 6.2 - Foto da maquete da granja desenvolvida em laboratório.

6.2.2. Controle da Vazão em Sistema de Distribuição de Água

O sistema de distribuição de água cresce de acordo com a urbanização. Com esse crescimento as redes se tornam mais complexas e em alguns lugares de difícil acesso. Nesse contexto a pressão da água é um fator crucial, pois influenciam diretamente o consumo e as perdas físicas.

Para trabalhar de acordo com o sistema de distribuição de água, realizou-se uma maquete utilizando tubos de PVC de 1 polegada de diâmetro, uma caixa de água como reservatório, uma bomba centrífuga de 1 cavalo de potência da marca WEG, registros de esfera manual (para a simulação de vazamento). Os transdutores inteligentes utilizados foram dois sensores de pressão, um de vazão e dois motores de passo trabalhando como atuadores. Neste trabalho teve como objetivo realizar o monitoramento da pressão da água e através disto, controlar a vazão da água. Na Figura 6.3 apresenta-se a foto do projeto desenvolvido em parceria com o laboratório de Hidrologia e Hidrometria da FEIS.

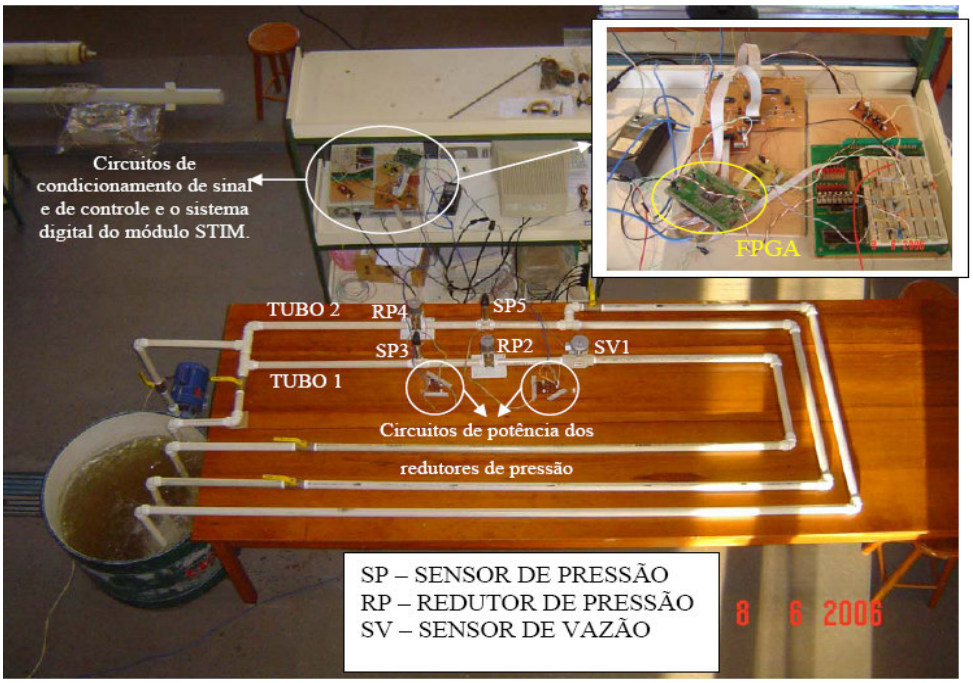

Figura 6.3 - Foto da maquete desenvolvida no laboratório de Hidrologia e Hidrometria da FEIS.

6.2.3. Monitoramento da Temperatura Ambiente

O controle de temperatura ambiente afeta de forma direta e indiretamente no comportamento das pessoas e no desempenho do trabalho. Para realizar esse controle de temperatura, utilizou-se um sensor de temperatura LM35 e um cooler como atuador, sendo que, pode-se trabalhar com qualquer outro tipo de atuador. O que difere este item dos demais testado, é o uso da tecnologia NIOS, onde foi possível integrar o sistema e a parte física do NCAP em um único componente re-configurável. Na Figura 6.4 apresenta-se a foto do circuito desenvolvido em laboratório.

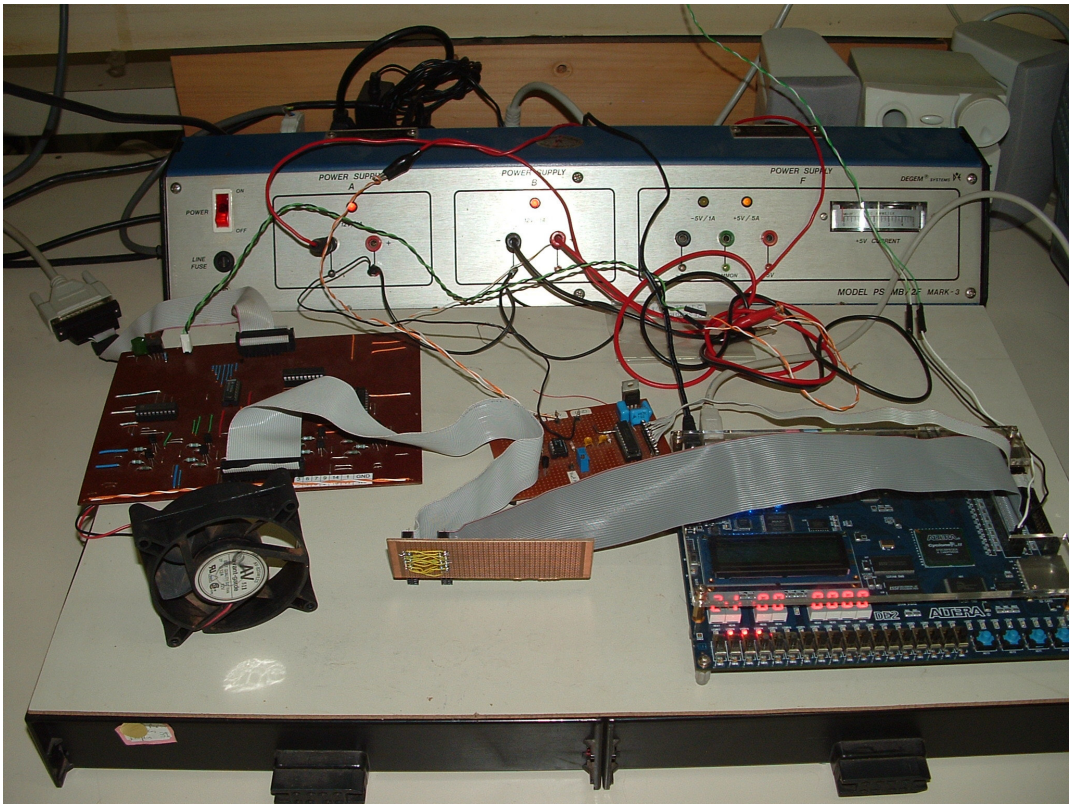

Figura 6.4 - Foto do processador NCAP e do módulo STIM implementado no Kit de desenvolvimento NIOS II.

6.3. Implementação do NCAP

O NCAP é composto por uma parte física e uma parte lógica. A parte lógica do NCAP, foi desenvolvida utilizando a linguagem de programação Java. Neste projeto não pretende detalhar a parte física do NCAP, apenas mostrar uma visão geral de como foi implementado.

6.3.1. Parte Física do NCAP

A parte física do processador NCAP é formada por um microcomputador e um dispositivo FPGA, onde é encontrado o gerenciador de protocolo,

O gerenciador de protocolo gerencia a comunicação através da interface TII e este bloco é implementado em um dispositivo FPGA. A comunicação entre o microcomputador e o gerenciador de protocolo é realizada através da porta paralela, utilizando métodos nativos da linguagem Java (9).

A parte de desenvolvimento do gerenciador de protocolo foi estudada em trabalho de doutorado (8), dissertação de mestrado (10) e projeto de iniciação científica (9).

6.3.2. Parte Lógica do NCAP

A parte lógica do processador NCAP foi desenvolvida em um microcomputador utilizando a plataforma Linux distribuição Slackware 10. O microcomputador utilizado foi um Pentium II, com processador de 200 MHz e memória RAM de 128 MBytes. Para desenvolver o *software* do NCAP, inicialmente realizou-se a instalação do pacote J2SDK. As etapas de instalação estão apresentadas no Apêndice A1. Também foi necessária a instalação do servidor Tomcat, apresentada no Apêndice A2.

A parte lógica do processador NCAP está dividida em duas formas de recepção de dados do ambiente externo, ou seja, através da rede ethernet, utilizando o protocolo http e *Socket*. A implementação do processador utilizando duas formas de comunicação através da rede ethernet, faz com que o sistema trabalhe com uma quantidade maior de dispositivos celulares ou sistemas desenvolvido em microcomputadores. A parte lógica do NCAP utilizando o protocolo http foi desenvolvido utilizando servlets, cujos detalhes de implementação estão apresentados no Apêndice B2.

A finalidade da parte lógica do NCAP é receber os dados do cliente e passar essas informações para o gerenciador de protocolo. Isso foi realizado usando-se a porta paralela do microcomputador para realizar a comunicação com o gerenciador de protocolo. A parte lógica do NCAP foi desenvolvida em Java e para realizar a comunicação com a porta paralela utilizaram-se métodos nativos em Java chamada JNI (*Java Native Interface*). A tecnologia Java permite que um código nativo possa ser carregado dentro de um programa escrito em Java através do emprego do JNI. Na prática, trabalhar com métodos nativos não é simples e exige certo conhecimento do programador, para isto, a solução é o pacote de domínio público *parport* desenvolvido por J.C. Del Cid Portillo. As etapas de instalação do pacote *parport* podem ser acompanhadas em detalhes no Apêndice B1.

A comunicação com o gerenciador de protocolo utilizando a porta paralela foi realizada de forma padronizada conforme o protocolo EPP (IEEE 1284). Esta iniciativa resultou num aperfeiçoamento da parte física deste processador. Os estudos realizados sobre a padronização de comunicação com o gerenciador de protocolo utilizando o protocolo EPP, já vinha sendo realizado no Laboratório de Processamento de Sinais e Sistemas Digitais. A manipulação dos registradores da porta paralela eram realizadas utilizando a linguagem Python (9). Para esta dissertação, o sistema foi reescrito e

implementado em Java, dessa forma, tornando o sistema padronizado em apenas uma única plataforma de programação.

Em sistema de controle e monitoramento dos transdutores inteligentes, na leitura manual pode ocorrer falha deixando de realizar as precauções necessárias, podendo gerar prejuízo no ambiente de produção ou até mesmo ocorrer acidentes. Como exemplo, podemos citar o ambiente de uma granja, que necessita de uma temperatura constante para analise da variação da temperatura gerando automaticamente relatórios de variação da mesma.

Para automatizar a parte lógica do NCAP desenvolvido nesta dissertação é realizada a leitura dos sensores, automaticamente gerando assim relatórios precisos e eficazes. Por exemplo, em um sistema de distribuição de água a leitura da pressão da água pode ser determinada pelo usuário de acordo com o tempo, podendo ser em, segundos, minutos ou horas. Essas informações podem ser acessadas através de um arquivo que é fornecido para o cliente, dessa forma, torna-se eficaz o monitoramento e o controle do sistema de distribuição de água. O processador NCAP realiza está função independente do ambiente em que esteja trabalhando, pois desde que o sensor seja conectado ao módulo STIM a leitura pode ser realizada normalmente.

A geração de relatórios não interfere na leitura do sensor realizado pelo usuário, pois a parte lógica do NCAP trabalhar com fluxo de *thread*, realizando assim um controle entre os processos. Ao iniciar a parte lógica do NCAP o *software* dispara dois processos, um para requisição pelo usuário e o outro para a geração de relatórios. O processo gerador de relatório aguarda um tempo pré-determinado, que após finalizado realiza a leitura do sensor. Neste processo, envia-se primeiramente a função, o canal, escrita 1 e escrita 2. Antes de dar início ao envio das funções de gerenciamento do NCAP é feita a verificação se o usuário realizou uma requisição de leitura ou controle. Caso exista um processo de requisição de leitura em andamento, o processo gerador de relatório volta à etapa inicial e aguarda o tempo pré-determinado para iniciar o novo processo de leitura. Caso não ocorra nenhuma requisição referente ao usuário, a parte lógica do NCAP obtém a resposta do gerenciador de protocolo e grava em arquivo o valor de leitura do sensor.

O arquivo de relatório gerado é criado de acordo com a data no seguinte formato: sensor e data como, por exemplo, 3-7-2-2007.TASF. O primeiro caractere contendo o valor três apresenta o número do canal do sensor que está sendo realizado à leitura, seguido pela data em que foi gerado o relatório.

Na Figura 6.5 apresenta-se o relatório de um sensor de temperatura disponibilizado através da web, o período de tempo para realizar a leitura deste relatório foi de 10 a 11 segundos.

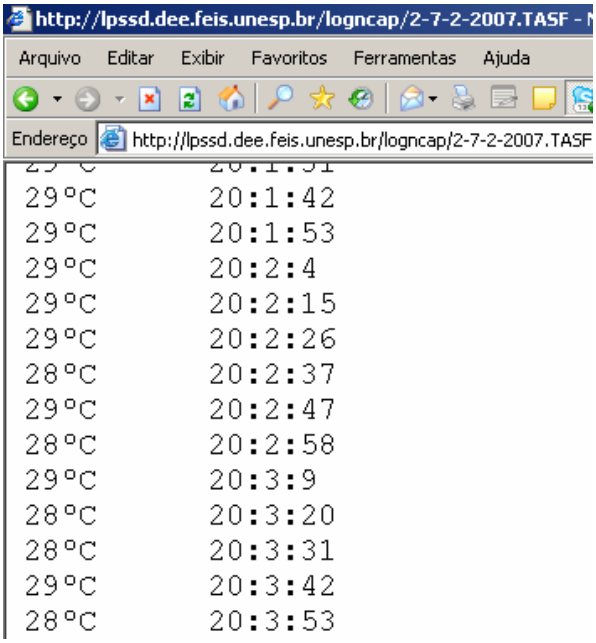

Figura 6.5 - Relatório do sensor de temperatura canal 3.

Pode-se visualizar melhor a parte lógica do processador NCAP através do fluxograma apresentado na Figura 6.6.

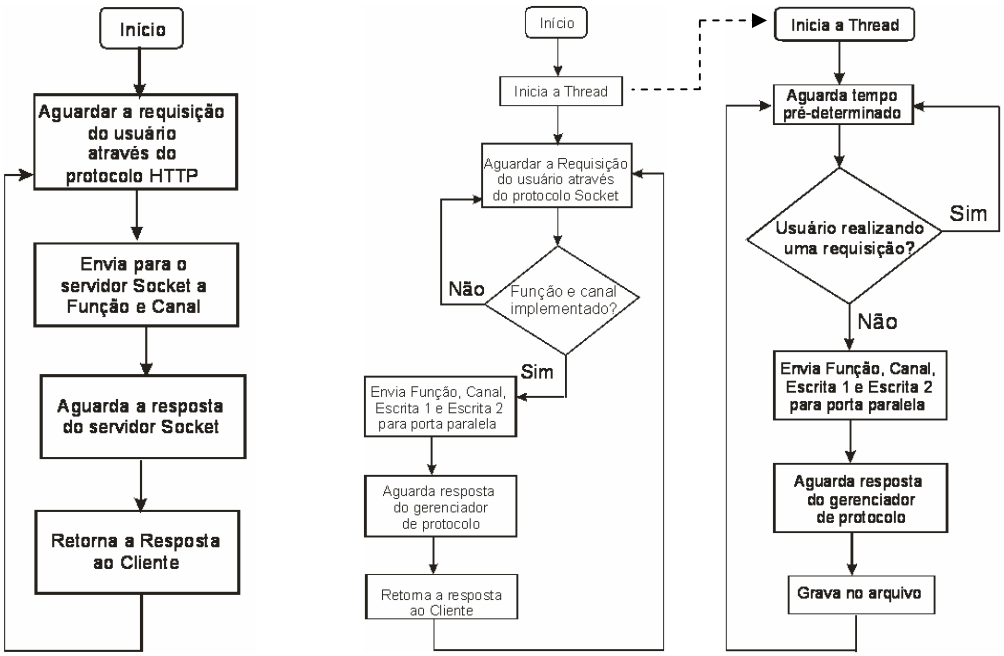

Figura 6.6 - Fluxograma da parte lógica do processador NCAP.

O principal enfoque deste trabalho foi a utilização do telefone celular capaz de realizar o gerenciamento e controle dos transdutores conectados em rede de acordo com o padrão IEEE 1451. Dessa forma, apresenta-se a seguir como o aparelho celular foi configurado.

6.4. Configurações do Aparelho Celular

6.4.1. Conexão com a Rede

O *software* MATIUDE trata-se de um aplicativo desenvolvido para os dispositivos celulares capaz de comunicar com uma rede de transdutores conectados de acordo com o padrão IEEE 1451, realizando assim, o controle e o monitoramento.

Para realizar a comunicação entre o dispositivo móvel e a parte lógica do NCAP, deve ser feita à configuração do aparelho celular para acesso a internet. A configuração é realizada de acordo com a operadora do usuário.

Neste projeto, utilizou-se a operadora CLARO que trabalha com a tecnologia de transmissão GSM. As configurações que foram efetuadas são apresentadas no Apêndice C3.

Para a realização dos testes, utilizou-se o celular modelo Motorola C385, CLDC 1.0 e MIDP 2.0 e o Motorola V220, CLDC 1.0, MIDP 2.0 com conexão s*ocket* ativa. Apesar dos testes serem realizados utilizando dois modelos de aparelhos celulares distintos, nesta dissertação, será apresentado apenas às imagens do modelo V220. Os dois aparelhos celulares apresentam a mesma forma de configuração para a comunicação com a rede ethernet, porém, o V220 possui comunicação *socket* com a rede.

Na Figura 6.7 apresenta-se os aparelhos utilizados e as configurações apresentadas pelos dispositivos móveis.

Figura 6.7 - Dispositivos móveis utilizados para teste do MATIUDE.

6.4.2. Transmissão de Dados

A transferência dos arquivos dos aplicativos para os dispositivos móveis podem ser realizadas das seguintes formas:

- Rede sem fio através da internet - utilizou-se o seguinte endereço para *download* do arquivo utilizado para a rede sem fio: <http://lpssd.dee.feis.unesp.br/*download*/MATIUDE.jar>;
- USB – para a visualização da configuração da interface USB para transferência dos dados, está descrito no Apêndice C2;
- Infravermelho - A transmissão de informações por meio de radiação na faixa do infravermelho.
- Bluetooth - é uma tecnologia de baixo custo para a comunicação sem fio entre dispositivos eletrônicos a curta distância.

Os modelos Motorola V220 e C385 com a configuração padrão vem bloqueada a opção de *upload* de aplicativos Java através da interface USB. Na opinião do autor os aparelhos celulares são bloqueados para que os usuários sejam forçados a realizar o *download* dos aplicativos Java através das redes sem fio, dessa forma, tendo gastos para a transferência dos dados. Neste contexto houve a necessidade de desbloquear o

aparelho celular, e para isto, utilizou-se o *software* PST(7.1.1_GENERAL) - Phone Programmer. A configuração do aparelho celular está apresentada no Apêndice C1.

Os aplicativos desenvolvidos para os dispositivos móveis são instalados automaticamente dentro da opção **Jogos & Aplicações**. Na Figura 6.8 apresenta-se a interface de acesso aos aplicativos disponível pelo celular.

Figura 6.8 - Acesso aos aplicativos desenvolvidos para o dispositivo celular.

Na Figura 6.9 apresenta-se a opção **Jogos & Aplicações** do aparelho celular. É nesta opção que o *software* MATIUDE foi instalado.

Figura 6.9 - Software MATIUDE no aparelho celular.

6.5. O Software MATIUDE

Ao acessar o *software* MATIUDE, o formulário de abertura denominado Principal oferece as seguintes opções:

- **MATIUDE**;
- **Configurações;**
- **Manual;**
- **Sobre;**
- **Sair**.

Na Figura 6.10 apresenta-se a interface do *software* MATIUDE.

Figura 6.10 - Interface do software MATIUDE.

A opção **MATIUDE** do formulário Principal será detalhada após a opção **Configurações**, já que, o *software* deve estar configurado corretamente para que sejam enviados os dados. As opções apresentadas no formulário Principal serão explicadas com detalhes nos Tópicos 6.5.1, 6.5.2, 6.5.3, 6.5.4.

6.5.1. Configurações

A opção **Configurações** permite ao usuário definir a forma em que deseja realizar a transferência dos dados: s*ocket* ou http. Neste projeto definiram-se duas formas de comunicação com a parte lógica do processador NCAP devido alguns aparelhos celulares possuírem apenas um tipo de conexão, à http. Estas duas formas são apresentadas na Figura 6.11.

Figura 6.11 - Opções de comunicação do software MATIUDE.

O NCAP pode ser desenvolvido em qualquer rede de computadores. As redes de computadores possui uma grande quantidade de endereçamento IP, sendo assim, o NCAP pode variar suas configurações de rede de acordo com o ambiente de implementação. Neste caso, para que os dispositivos móveis possam transmitir os dados para a parte lógica do NCAP é necessário estar configurado o endereço IP da rede em que se encontra a parte lógica do processador e a porta de comunicação que o processador NCAP está oferecendo o serviço.

Para realizar as configurações no *software* MATIUDE, após definir a forma de comunicação http ou s*ocket,* o usuário terá disponível dois campos de entrada, o Endereço e Porta de Comunicação. De acordo com as configurações do processador NCAP o usuário deverá inserir o endereço IP e a porta de comunicação no *software*

MATIUDE de acordo com a Figura 6.12. Neste projeto, as configurações para comunicação, são IP 200.145.248.238 e a porta 2000 para *socket* e o protocolo http, o IP 200.145.248.238 e a porta 8080.

Figura 6.12 - Configuração do software MATIUDE.

6.5.2. Manual

Na opção Manual apresentada na Figura 6.13 descreve os procedimentos para a utilização do *software*.

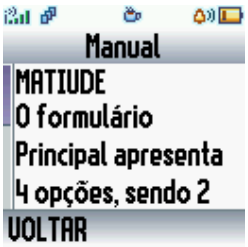

Figura 6.13 - Manual do sistema MATIUDE.

6.5.3. Sobre

A opção **Sobre** apresenta a equipe do LPSSD que fazem parte deste projeto. Na Figura 6.14 apresenta-se a interface informando a equipe do projeto MATIUDE.

Figura 6.14 - Menu Sobre apresentando a equipe do projeto MATIUDE.

6.5.4. MATIUDE

A opção **MATIUDE** apresentado na Figura 6.10 permite acesso ao formulário Opções onde o usuário terá a disposição as seguintes opções de manipulação:

- **Leitura dos Transdutores** – realiza a leitura de um sensor;

- **Atuador** – realiza a abertura ou o fechamento do canal atuador;
- **Atuador - 2** – liga ou desliga um determinado atuador;
- **Relatório** – realiza a leitura dos relatórios realizados pelo processador NCAP;
- **Gráfico** – apresenta o gráfico dos valores lidos nos canais de transdutores. Esta opção encontra-se em fase de finalização, sendo que, será tratado em trabalho futuro;
- **Lista de Transdutores** – apresenta os canais dos transdutores conectados ao modulo STIM;
- **Leitura dos Transdutores** – realiza a leitura dos transdutores em uma faixa de valores.

Na Figura 6.15 apresenta-se as opções definidas para monitoramento e controle dos transdutores inteligentes no formulário **Opções**.

Figura 6.15 - Opções de manipulação do software MATIUDE.

6.5.4.1. Leitura dos Transdutores

A opção **Leitura dos Transdutores** permite acessar a interface e escolher em qual canal será realizada a leitura, e esta opção realiza apenas uma leitura. Ao escolher esta opção **Leitura dos Transdutores**, o usuário estará definindo a função 128 para leitura do transdutor inteligente de acordo com o padrão IEEE 1451.

No formulário **Sensor** o campo **Escreva o Canal**. Este campo indica qual o canal que se deseja realizar a leitura. Na Figura 6.16 apresenta-se a interface de inserção do canal de leitura.

Figura 6.16 - Escolha do canal de leitura.

O MATIUDE dispõe de uma opção **Canais** que é acessada através da tecla **Menu** do celular. Na Figura 6.17 apresenta-se a opção de **Canais** que apresenta quais os canais disponíveis no NCAP.

Figura 6.17 - Opção de canais disponíveis no processador NCAP.

A opção **Canais** realiza a leitura de um arquivo contido na parte lógica do processador NCAP. Este arquivo contém quais os canais fazem parte do módulo STIM e qual o transdutor inteligente interligado. Na Figura 6.18 apresenta-se os canais disponíveis pelo módulo STIM.

Figura 6.18 - Canais implementado no módulo STIM.

Ao inserir o valor do canal e confirmar a opção, o valor da função e do canal são enviados à parte lógica do processador NCAP. Através da rede ethernet e repassa em forma de sinais digitais ao gerenciador de protocolo utilizando a porta paralela do microcomputador. As informações transmitidas do NCAP para o gerenciador de protocolo são enviadas de acordo com o protocolo EPP. O gerenciador de protocolo ao receber os dados realiza comunicação com os módulos inteligentes por meio da interface física padronizada TII. Quando o gerenciador de protocolo finaliza a comunicação com o módulo STIM, retorna a resposta através da porta paralela de acordo com o protocolo EPP. O NCAP ao receber os dados do gerenciador de protocolo retorna a resposta ao usuário utilizando a rede ethernet.

O cliente MATIUDE por estar trabalhando com uma comunicação *socket* bloqueante fica aguardando a resposta da parte lógica do processador NCAP até que seja retornado o resultado. Caso o valor de resposta não retorne em um determinado tempo é apresentada a mensagem de erro ao usuário. Na Figura 6.19 apresenta-se a

interface do aparelho celular recebendo a resposta do servidor NCAP. A leitura apresentada é proveniente do módulo STIM apresentada na Figura 6.4 onde realizou-se a leitura de temperatura em ambientes fechado. A MATIUDE disponibiliza a leitura do transdutor de qualquer outro tipo de sensor, desde que esteja conectado ao módulo STIM. Outros testes realizados utilizaram sensores de pressão e de vazão.

Figura 6.19 - Interface de resposta da MATIUDE.

Caso os dados não sejam transmitidos, o MATIUDE informa ao usuário que não foi possível realizar a comunicação com a parte lógica do processador NCAP. Os erros podem ocorrer quando o celular não estiver configurado corretamente, por falta de créditos ou por não encontrar a parte lógica do NCAP pelo endereço indicado pelo usuário. Na Figura 6.20 apresenta-se a interface de erro do *software* MATIUDE.

Figura 6.20 - Interface de erro do software MATIUDE.

6.5.4.2. Atuador

Para o controle e manipulação dos atuadores foram definidas duas formas, sendo **Atuador** e **Atuadores - 2**. A opção **Atuador** corresponde ao controle dos atuadores que necessitam das funções de controle para o posicionamento, por exemplo, os motores de passo[7]. A opção **Atuadores - 2** é utilizada para atuadores que tem a função de apenas ligar ou desligar, por exemplo, as lâmpadas. A primeira opção, **Atuador,** foi testada e analisada utilizando o sistema de distribuição de água que possui 2 redutores de pressão. Na Figura 6.3 foi apresentada à foto da maquete do sistema desenvolvido.

[7] Motor de passo - desloca-se por impulsos ou passos discretos e exibem três estágios: parados, ativados com rotor travado (bobinas energizadas) ou girando em etapas. Este movimento pode ser brusco ou suave, dependendo da freqüência e amplitude dos passos em relação à inércia em que ele se encontre.

Para realizar o controle dos atuadores a opção **Atuador** apresentada na Figura 6.15 e fornece ao usuário o controle do motor de passo. Para o controle do motor de passo foram definidas as seguintes funções e valores de controle:

- **Ligar:** Função - 16 e escrita1 - 1;
- **Desligar:** Função - 16 e escrita1 - 2;
- **Selecionar modo de menor potência:** Função - 16 e escrita1 - 4;
- **Selecionar modo de maior potência:** Função - 16 e escrita1 - 8;
- **Modificar a posição do redutor:** Função 1 - 1, escrita1 - 1 e escrita2 - valor da posição desejada.

Os valores de escrita1 e escrita2 são os valores de controle. Por exemplo, a função 16 refere-se ao acionamento do redutor e o valor de controle liga, desliga ou seleciona o modo de potência do redutor. Já na função 1, o valor de controle informa a posição onde o redutor deve se posicionar. Para acionar os atuadores, seguem o mesmo exemplo realizado para os sensores, enviando os valores de Função, Canal, Escrita1 e Escrita2 (9).

Após a escolha da opção Atuador, o usuário define qual o canal e o estado do redutor de pressão, ligado ou desligado. Na Figura 6.21 apresenta-se as opções disponíveis no *software* MATIUDE para o controle do motor de passo.

Figura 6.21 - Escolha do canal e o estado do redutor de pressão.

A opção **Desligar** desativa o redutor de pressão. A opção **Ligado** habilita o atuador para o modo ativo e aguarda a próxima instrução. Ao confirmar a opção, o usuário estará enviando a primeira seqüência de manipulação do atuador para o NCAP. A próxima interface apresenta a força em que o atuador irá exercer para executar a tarefa, sendo disponíveis em dois estados; **Mínimo** e **Máximo**. Os valores de função e canal serão os mesmo, pois o sistema estará dando continuidade ao processo de manipulação. Na Figura 6.22 apresenta-se a interface de definição da força do motor de passo.

Figura 6.22 - Interface de definição da força do redutor de pressão.

Para indicar o posicionamento do motor de passo, os valores são apresentados através de porcentagem proporcionalmente aos valores definidos no processador NCAP, os respectivos valores são: 0 (aberto) no MATIUDE é representado como 0%, e 255 (fechado) no MATIUDE é representado como 100%. Para facilitar a visualização e controle pelo usuário foi criada através de barra a definição do posicionamento do motor de passo. Na Figura 6.23 apresenta-se a interface de posicionamento do redutor de pressão.

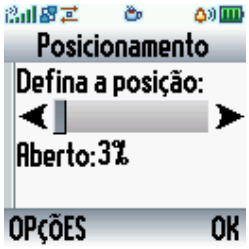

Figura 6.23 - Posicionamento do redutor de pressão.

Como finalização do processo de controle é apresentada na Figura 6.24 o modo do atuador, força e a posição.

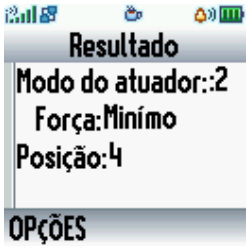

Figura 6.24 - Estado do motor de passo definido pelo usuário.

6.5.4.3. Atuador - 2

A opção **Atuador - 2** foi desenvolvida para ligar ou desligar os atuadores. Neste projeto, os testes foram realizados utilizando o sistema de controle de temperatura em ambientes fechados e no monitoramento de temperatura. As fotos dos sistemas podem ser visualizadas nas Figura 6.2 e Figura 6.4. Os valores da função, canal, escrita1 e

escrita2 foram definidos da seguinte forma para o sistema de controle de temperatura em ambientes fechados:

- **Ligar *cooler*** – função 16, canal 4, escrita1 1, escrita 2 3;
- **Desligar *cooler*** - função 16, canal 4, escrita1 2, escrita 2 1;
- **Ligar lâmpada** – função 16, canal 5, escrita1 1, escrita 2 3;
- **Desligar lâmpada** – função 16, canal 5, escrita1 2, escrita 2 1.

Para os testes no sistema de monitoramento de ambiente foram definidas as seguintes funções:

- **Ligar *cooler*** – função 16, canal 2, escrita1 1, escrita 2 3;
- **Desligar *cooler*** – função 16, canal 2, escrita1 2, escrita 2 1.

Na Figura 6.25 apresenta a interface para o acionamento e a desativação dos atuadores.

Figura 6.25 - Campo de entrada para acionamento dos atuadores.

Após confirmar os valores de controle os dados são enviados para o processador NCAP e chama a próxima interface que apresenta o estado em que o atuador se encontra. Na Figura 6.26 apresenta-se a interface do estado e o envio dos dados para o processador NCAP.

Figura 6.26 - Interface de envio e estado do atuador.

6.5.4.4. Relatório

O relatório do monitoramento dos transdutores inteligentes é de grande importância em qualquer tipo de sistema, seja controle de vazão, monitoramento da temperatura em ambientes fechados ou abertos, entre outras áreas. Através do relatório

pode-se verificar o comportamento do sistema e realizar as precauções necessárias evitando acidentes ou prejuízos.

Na organização dos relatórios na parte lógica do processador NCAP a cada dia é gerado um novo arquivo. Esses arquivos são nomeados de acordo com o sensor em que se realizou a leitura e de acordo com a data da seguinte forma: 3-10-07-2006.TASF. Para a praticidade do usuário, esses dados são repassados para o aparelho celular através do *software* MATIUDE. Na Figura 6.27 apresenta-se a opção de relatório disponível para o usuário.

Figura 6.27 - Interface com a opção relatório.

Para realizar leitura do relatório, é necessário indicar qual o sensor que se deseja obter as informações, pois o sensor é indicado de acordo com o canal. Na Figura 6. 28 apresenta-se o campo de inserção do canal para realização da leitura do relatório. Na inserção dos canais para leitura dos relatórios também é disponibilizada a opção de visualização dos canais, disponível através da tecla **Menu** do celular.

O campo destinado à inserção do canal do sensor permite ao *software* MATIUDE enviar os dados do sensor para o qual se deseja realizar a leitura do relatório.

Figura 6. 28 - Interface para inserção do canal para leitura do relatório.

Ao inserir o canal, a interface referente ao formulário **Relatório** disponibiliza o campo **Escolha a Data**. Este campo, apresentado na Figura 6.29, obtêm-se a data que se deseja realizar a leitura do relatório. Para a tranqüilidade de manipulação do usuário, o campo já vem preenchido com um exemplo de como deve ser inserido a data.

Figura 6.29 - Informações para requisição do relatório.

De acordo com a leitura dos sensores, podem-se ter vários arquivos de relatório facilitando para o usuário a identificação dos relatórios que estão disponíveis. Para visualizar os relatórios disponíveis, definiu-se uma nova opção que é acessada através do **Menu** do dispositivo celular chamada **Relatório**. As datas são obtidas pela parte lógica do NCAP e gravadas em arquivos (exemplificado no Tópico 6.3.2), onde são repassadas para o *software* MATIUDE utilizando o protocolo http. Na Figura 6.30 apresenta-se a interface dos relatórios disponíveis pela parte lógica do NCAP.

Figura 6.30 - Interface de relatórios disponíveis.

Após o usuário confirmar a data, o MATIUDE realiza a requisição para a parte lógica do NCAP retornando os dados do relatório. Os valores retornados são: leitura do sensor e a hora. Na Figura 6.31 apresenta-se a interface de leitura do relatório.

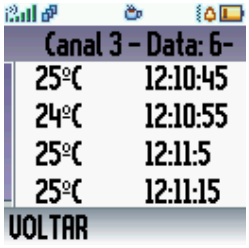

Figura 6.31 - Interface de retorno do relatório do sensor de temperatura.

6.5.4.5. Gráfico

A próxima opção disponível no formulário **Opções** refere-se ao gráfico. O desenvolvimento da parte da visualização dos valores dos sensores em formato de gráfico encontra-se em fase de desenvolvimento e algumas implementações já foram

realizadas, porém, devido aos diversos dispositivos celulares disponíveis no mercado terem características diferentes como o *display,* a implementação de gráficos para a representação dos valores dos sensores se torna um tanto complexa.

6.5.4.6. Lista de Transdutores

A opção **Lista de Transdutores** disponível no formulário **Opções**, refere-se à lista de canais de transdutores inteligentes interligados ao módulo STIM. Ao escolher a opção é realizada uma requisição para a parte lógica do NCAP retornando a lista de canais de transdutores inteligentes contendo o canal e o transdutor pertencente.

6.5.4.7. Leitura dos Sensores

A opção **Leitura dos Sensores** disponível no formulário **Opções**, disponibiliza ao usuário a leitura dos sensores através de uma faixa de valores. A leitura dos sensores utilizando uma faixa de canais facilita na inserção dos valores e na leitura de uma grande quantidade de canais. Pode-se notar no formulário **Leitura dos Sensores** apresentado na Figura 6.32 dois campos, **Mínimo** e **Máximo**.

Figura 6.32 - Formulário de leitura dos sensores de acordo com a faixa de valor.

Ao inserir os valores e confirmar a opção de leitura, para os canais que estiverem entre a faixa de valores **Mínimo** e **Máximo** serão realizadas as leituras apresentando o resultado no próximo formulário chamado **Leitura dos Sensores**. Na Figura 6.33 apresenta-se a resposta da leitura dos sensores. Para os canais com resposta 0 significa que são canais não utilizados ou canais dos atuadores.

Figura 6.33 - Resposta da leitura dos sensores.

Capítulo 7

Conclusões Gerais

7.1. Conclusões

Neste trabalho foi apresentado um *software* desenvolvido no LPSSD chamado MATIUDE capaz de monitorar e controlar transdutores inteligentes interligados em rede de acordo com o padrão IEEE 1451. A MATIUDE foi desenvolvida utilizando a linguagem J2ME, por ser ferramenta gratuita e padronizada, e por atingir um grande número de dispositivos embarcados (celulares e PDA`s).

A parte lógica do NCAP foi desenvolvida através da tecnologia Java que realiza a comunicação com o gerenciador através da porta paralela do microcomputador. O processador NCAP é interligado à rede Ethernet trabalhando de acordo com o modelo cliente-servidor.

Para o desenvolvimento deste projeto, houve a necessidade de se entender o funcionamento da parte lógica do processador NCAP, e como realizar a comunicação com a rede Ethernet utilizando os dispositivos embarcados. Primeiramente foi desenvolvido o *software* MATIUDE utilizando a linguagem J2ME e o simulador *Wireless Toolkit* disponibilizado gratuitamente pela Sun. Os estudos realizados com o simulador demonstraram a praticidade de se desenvolver aplicações para celular e realizar a comunicação com os servidores, porém, utilizando *Socket* alguns aparelhos não suportam tendo que migrar para o protocolo http.

Para a transferência de dados com o aparelho celular, optou-se pela porta USB do microcomputador com o cabo USB AxB-mini conectado ao dispositivo celular, realizando assim quantas vezes necessárias a transferência do arquivo sem custo algum.

Os testes realizados utilizando o aparelho celular do fabricante Motorola modelo V220 e C385, demonstraram a necessidade de desenvolver um sistema otimizado e dinâmico, pois, a quantidade de aparelhos com diferentes características torna a programação um tanto complexa. O tempo de resposta do processador NCAP ao

dispositivo celular pode variar de acordo com a qualidade do sinal, sendo que quanto melhor o sinal mais rápido será o valor de retorno. Os custos podem variar de acordo com a operadora. Informação apresentada no Apêndice C5.

Como sugestão de trabalhos futuros tem-se:

- Desenvolvimento de gráficos para analise dos relatórios;
- Melhoramento do acesso à porta paralela do servidor NCAP utilizando a linguagem de programação Java;
- Monitoramento de transdutores inteligentes através de *stream-video* utilizando dispositivos embarcados;
- Comunicação com a porta serial do dispositivo embarcado para a realização da comunicação com o gerenciador NCAP.

Referências

(1) TABLELESS. Tendências da internet móvel. Disponível em: **<http://www.tableless.com.br/aprenda/tendencias-da-internet-movel/>** Acesso em: 17 ago. 2006.

(2) AMORIN, D. E.; SHIMA, W. T. **Convergência tecnológica e a formação de novos tipos de alianças estratégicas:** uma análise do desenvolvimento dos Personal Digital Assistent (PDA´s). **Revista Brasileira de Inovação,** Rio de Janeiro, v.5, p. 273-314, 2006.

(3) WIKIPÉDIA - GSM (Global System for Mobile Communications). Disponível em:**<http://pt.wikipedia.org/wiki/GSM#Estat.C3.ADstica_de_Assinantes_de_Telefonia_Celular>**. Acesso em: 16 dez. 2006.

(4) OLIVEIRA, D. S. **GPS com JavaME - Web Mobile. Edição 11**. Ano 2002. Publicação Bimestral. p. 16-36.

(5) AMORIM, A. R.; BORGES, S. K. Desenvolvimento de aplicações móveis com J2ME. In: SEMINÁRIO DE INFORMÁTICA – RS (SEMINFO-RS´2005), JORNADA INTEGRADA DE TRABALHOS DE CONCLUSÃO DE CURSO EM COMPUTAÇÃO (JIT3C), 2005, Torres. **Seminário...**, 2005. p. 155-161.

(6) JARDIM, F. M. **Framework para implementação de serviços transmissores de vídeos voltados a PCs e dispositivos móveis, perceptivo à mudança contextual de localização**. 2004. Dissertação (Mestrado) - Universidade Federal de São Carlos. Centro de Ciências Exatas e de Tecnologia Programa de Pós-Graduação em Ciência da Computação, São Carlos, 2004.

(7) WILLIAN, T.; CLEM, G.; PARRA, M.; MAXWELL, J. Early warning environmental protection system. **Florida Gulf Coast University**, Flórida-EUA, Novembro, 2005.

(8) ROSSI, S. R. **Implementação de um nó IEEE 1451, baseado em ferramentas abertas e padronizadas, para aplicações em ambientes de instrumentação distribuída.** 2005. 231 f. Dissertação (Doutorado em Engenharia Elétrica) – Faculdade de Engenharia de Ilha Solteira, Universidade Estadual Paulista, Ilha Solteira, 2005.

(9) PRADO, T.A.; SILVA, A.C.R.; SOBRINHO, M.D. **Implementação de sistema para automatizar o processo de medição de vazão e de pressão numa rede de distribuição de água empregando o padrão IEEE 1451.** Ilha Solteira: UNESP/FEIS/DEE, 2006. 49p (Relatório final de curso).

(10) BATISTA, E.A. **Emprego da tecnologia java para implementar a parte lógica de um processador de aplicação com capacidade de operar em rede de comunicação (NCAP), em conformidade com o padrão IEEE 1451**. 2004. 104 f. Dissertação (Mestrado em Engenharia Elétrica) - Faculdade de Engenharia de Ilha Solteira, Universidade Estadual Paulista, Ilha Solteira, 2004.

(11) PIROPO, B. Multimídia no bolso. Jornal o Estado de Minas. Disponível em: **<http://www.bpiropo.com.br/em20040506.htm#PCA>.** Acesso em: 02 set. 2006.

(12) INTEL. Processador Intel PXA800F. Disponível em: **<http://www.intel.com/portugues/design/pca/prodbref/252336.htm>**. Acesso em: 15 ago. 2006.

(13) INTEL. Processador Intel PXA800F. Disponível em: < **http://www.intel.com/design/pca/prodbref/253820.htm**>. Acesso em: 15 ago. 2006.

(14) FAQ. Vivo. Mais cobertura em voz, mais veloz em dados. Disponível em: **<https://www2.tco.net.br/vivo/hotsites/cdma/cdmaoverlay_co/faq.htm>**. Acesso em: 20 jul. 2005.

(15) MUNDO SEM FIO. O portal dos entusiastas da tecnologia móvel. Disponível em**: <http://www.mundosemfio.com.br/2004/08/040802_o_tdma_ja_morreu_par.html>**. Acesso em: 20 jun. 2005.

(16) TELEFONICA CELULAR. TDMA. Disponível em: **<http://proenca.uel.br/curso-redes-graduacao/1998/trab-05/equipe-01/tdma.htm>.** Acesso em: 20 set. 2005.

(17) TERRA CELULAR. Tecnologia CDMA. Disponível em: **<http://www.terra.com.br/celular/oquee_cdma.htm>**. Acesso em: 02 set. 2005.

(18) TERRA CELULAR. Tecnologia GSM. Disponível em: <**http://www.terra.com.br/celular/oquee_gsm.htm**>. Acesso em: 30 set. 2005.

(19) AREHART, C.; CHIDAMBARAM, N.; GURUPRASAD, S.; HOMER, A.; HOWELL, R.; KASIPPILLAI, S.; MACHIN, R.; MYERS, T.; NAKHIMOVSKY. A.; PASSANI. L.; PEDLEY. C.; TAYLOR. R.; TOSCHI. M. De Programador para Programador. In: **____. Professional WAP**. São Paulo: Makron Books, 2001. 774 p.

(20) MICROSOFT CORPORATION. Msdn. Disponível em: **<http://msdn.microsoft.com/404/default.aspx>.** Acesso em: 04 dez. 2005.

(21) TEIXEIRA. C.; JULIANO. C.; JEVEAUX M. C. P. Tudo sobre SuperWaba e derivados. Disponível em: **<https://www.dev.java.net/files/documents/353/13788/tudo_superwaba.pdf>.** Acesso em: 09 jan. 2006.

(22) MUCHOW. J. W. Programação Java/J2ME. In: **____.Core J2ME tecnologia & MIDP.** São Paulo: Makron Books, 2004. 588 p.

(23) SILVA, V.F.; OLIVEIRA, J. A.; FONSECA, J.A. Ambiente de execução de aplicações Java. Disponível em: < **http://www.fccn.pt/crc2001/pdf/artigos/crc2001_123_a32vf.pdf>.** Acesso em: 21 set. 2005.

(24) CESTA, A. A. Tutorial "a linguagem de programação Java, orientação a objetos**"**. Disponível em: **<http://www.apostilando.com>**. Acesso em: 15 jan. 2006.

(25) NETBEANS. Bem-Vindo ao NetBeans e ao Site www.netbeans.org – NetBeans**.** Disponível em: **<http://www.netbeans.org/index_pt.html** >. Acesso em: 12 fev. 2007.

(26) NETBEANS. Notas da versão do mobility pack NetBeans 5.5 – NetBeans. Disponível em: **<http://www.netbeans.org/community/releases/55/relnotes-mobility_pt_BR.html>**. Acesso em: 12 fev. 2007.

(27) PITER PUNK'S. **Por que usar Slackware.** Disponível em: **<http://www.piterpunk.hpg.ig.com.br/artigos/porque.html.>** Acesso em: 02 out. 2005.

(28) D´ÁVILA, M. Tutorial tomcat – instalaçâo e configuraçâo. Disponível em: **<http://www.mhavila.com.br/topicos/java/tomcat.html>**. Acesso em: 22 ago. 2006.

(29) OLIVEIRA, A. P. Apostila de Servlets/JSP. Disponível em: **<http://www.apostilando.com/*download*.php?cod=2176>.** Acesso em: 15 jul. 2006.

(30) UNIVERSO CELULAR – CONEXÃO DE DADOS – UCEL. Disponível em: **<http://www.ucel.com.br/dados.asp>**. Acesso em: 20 fev. 2007.

Apêndice A - Instalação do J2SDk e do Servidor Tomcat na plataforma linux

A1 - Instalação do J2SDK

Para a instalação do J2SDK é necessário realizar o *donwload* do pacote. Este pacote é disponível pelo *site* da *Sun Microsystems* através do *link* <http://java.sun.com/javase/*downloads*/index.jsp> na opção de *download*. Nesta dissertação, apresenta-se a instalação dos componentes em ambiente Linux distribuição Slackware 10. Depois de escolher a opção para a plataforma Linux em formato *.bin*[8] o arquivo deve ter permissão de execução, para isso é realizado o seguinte comando em modo *shell*[9], chmod 777 <nome do pacote java>. Após, a execução do comando ./<nome do pacote Java> é realizada a instalação do pacote Java, como sugestão, sugere realizar a instalação no seguinte diretório /usr/local/.

É útil deixar configuradas algumas variáveis de ambiente relacionadas à Java. Para realizar a configuração é necessário editar o arquivo *profile* dentro da pasta /etc/, as alterações pode ser realizada utilizando qualquer editor à escolha do usuário. As variáveis de ambiente relacionadas são:

JAVA_HOME

Local de instalação do JDK (Kit de Desenvolvimento Java).

CLASSPATH

Caminhos (pacotes e diretórios) de localizações de classes Java; o classpath deve incluir o(s) jar(s) dos pacotes Servlet e JSP do Tomcat.

PATH

Caminhos (diretórios) de localizações de executáveis no sistema operacional, deve incluir o diretório `bin` das ferramentas do Java SDK.

Na linha 10 do arquivo profile insira as seguintes linhas de comando:

- JAVA_HOME=/usr/local/j2sdk1.4.2_13
- CLASSPATH=.:%CLASSPATH%:$JAVA_HOME/lib:$JAVA_HOME/bin: $JAVA_HOME/jre/bin:
- export JAVA_HOME CLASSPATH

[8] .bin – extensão dos arquivos executáveis para a plataforma Linux
[9] Shell – ambiente modo texto na plataforma Linux.

A2 - Instalação do Servidor Tomcat

Antes de realizar a instalação do servidor Tomcat é necessário que se tenha o J2SDK instalado, sugere a versão mais atual disponibilizada no *site* da *Sun Microsystems*. Para realizar a instalação, é efetuado o *download* do pacote de instalação através do *site* <http://tomcat.apache.org/*download*-55.cgi> na opção de *download*. Nesta dissertação, utilizou-se o pacote de instalação apache-tomcat-5.5.20.tar.gz para plataforma Linux. Ao baixar o pacote, descompacte dentro do diretório /opt/ utilizando o seguinte comando `tar -xzvf apache-tomcat-?.?.*.tar.gz`, a seguir, efetue os seguintes linhas de comando para realizar a instalação.

- `ln -s apache-tomcat-?.?.?? tomcat` – para facilitar a manipulação do diretório é recomendável criar um link simbólico;
- `cd tomcat/bin` – acesso ao arquivos executáveis do servidor Tomcat;
- `ls -l *.sh -` listar os arquivos .sh do diretório bin;
- `chmod +x *.sh -` criar permissão de execução aos arquivos .sh;
- `cd ../.. -` sair do diretório.

Ao finalizar a instalação, as variáveis de ambiente devem ser configuradas no arquivo profile, o arquivo se encontra dentro da pasta /etc. Para editá-lo, execute os seguintes comandos:

- cd /etc
- pico profile – pode ser utilizado outros editores a preferência do usuário.

As variáveis de ambiente devem ser editadas da seguinte forma, junto ao J2SDk:

```
JAVA_HOME=/usr/local/j2sdk1.4.2_13
CATALINA_HOME=/opt/tomcat
CLASSPATH=$CATALINA_HOME/common/lib/servlet-api.jar:.:$CLASSPATH
CLASSPATH=$CATALINA_HOME/common/lib/jsp-api.jar:$CLASSPATH
CLASSPATH=.:%CLASSPATH%:$JAVA_HOME/lib:$JAVA_HOME/bin:$JAVA_H
OME/jre/bin
export JAVA_HOME CATALINA_HOME CLASSPATH PATH
```

Salve o arquivo, sai do terminal e abra outro para que as configurações possam ser efetuadas.

A configuração inicial mínima recomendada após a instalação dos arquivos do Tomcat é adicionar, no arquivo `conf/tomcat-users.xml`, um usuário e senha de

admininistrador com autorizações para uso das ferramentas admin e manager do Tomcat. Estando no diretório base /opt/apache-tomcat<versão>/ após a instalação, edite o arquivo `tomcat-users.xml` que se encontra dentro do diretório conf para realizar as seguintes alterações. Adicione a linha em destaque a seguir, defindo um nome e senha para o usuário administrativo do Tomcat (altere a senha, e opcionalmente também o nome, à sua escolha):

<tomcat-users>

<user username="tomcat" password="tomcat" roles="tomcat"/>

<user username="role1" password="tomcat" roles="role1"/>

<user username="both" password="tomcat" roles="tomcat,role1"/>

<user username="root" password="*" fullName="Iako" roles="admin,manager"/>**

</tomcat-users>

Para inicializar o servidor Tomcat é necessário entrar na pasta /opt/<diretório do apache tomcat>/bin/, após acessar, os seguintes comandos terão os controles sobre o servidor:

- `./startup.sh` - inicializa o servidor;
- `./shutdown.sh` - para o servidor.

Para testar o funcionamento do servidor, abra o navegador de internet e digite no endereçamento http://localhost:8080/ , na Figura A.1 apresenta-se a interface inicial do servidor Tomcat. Na interface, aparecerão links de exemplos de servlets e JSP (*Java Server Pages*) disponibilizará o controle do servidor e tutoriais de instalação de novas ferramentas.

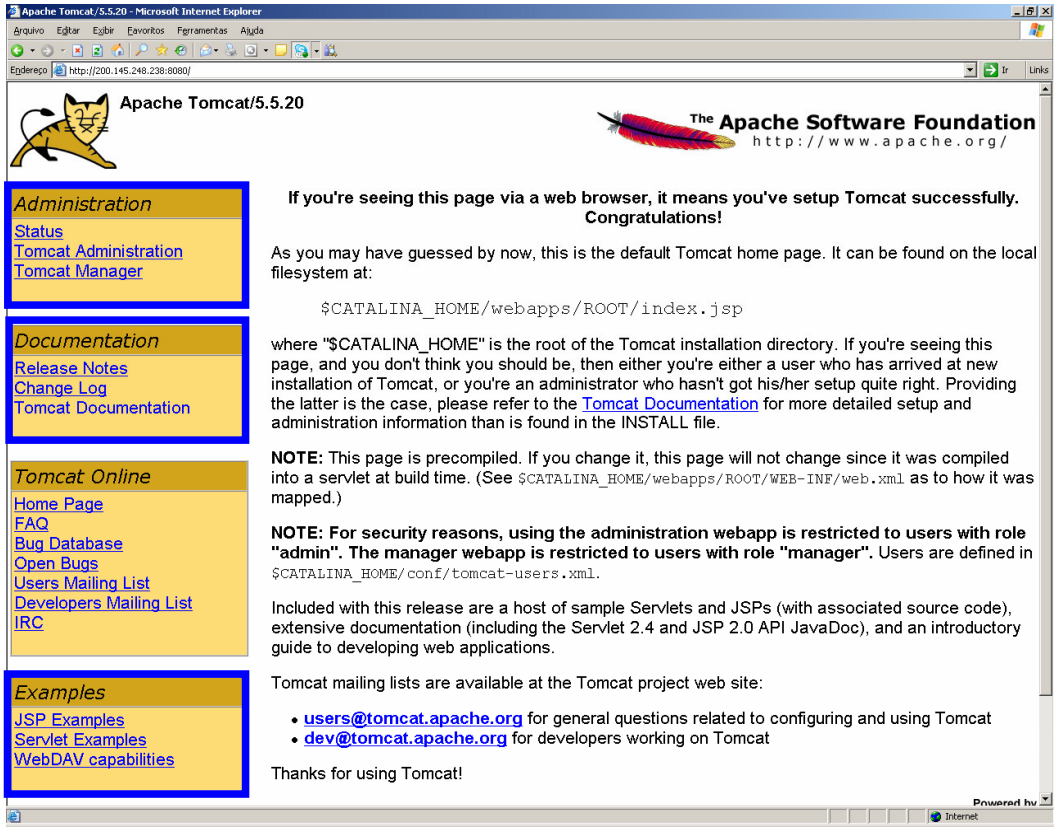

Figura A.1 - Interface de gerenciamento do Tomcat.

Apêndice B - Acesso à porta paralela utilizando métodos nativos JNI (Java Native Interface)

B1 - Instalação e Configuração da Biblioteca Parport para Acesso a Porta Paralela.

Parport é um pacote que pode ser instalados quanto em sistemas operacionais *Linux* ou *Windows,* o pacote permite realizar a programação utilizando Java, porém o parport emprega a linguagem C para realizar a comunicação direta com a porta paralela.

O sistema operacional utilizado para instalação foi o Linux distribuição Slackware 10, para instalação, foi efetuada os seguintes passos:

- cd /usr/local/j2sdk1.4.2_13
- mkdir classes
- cd classes

Após criar a pasta classes baixe o arquivo do pacote *parport*, o arquivo pode ser encontrado através do *site* <http://www.geocities.com/Juanga69/parport/parport-linux.zip>. Descompacte o arquivo dentro da pasta classes utilizando o comando:

- unzip parport-linux.zip

Entre na pasta *parport* e copie o arquivo libparport.so dentro da pasta ../j2sdk1.4.2_13/lib/ , o comando utilizando nesta dissertação foi:

- cp libparport.so /usr/local/j2sdk1.4.2_13/lib/

O comando pode variar de acordo com o local de instalação do Java. Após copiar o arquivo, devem-se configurar as variáveis de ambiente através do arquivo *profile*, o arquivo encontra-se dentro da pasta /etc/, para editar o arquivo execute os seguintes comandos:

- cd /etc/
- pico profile – (pode ser utilizado qualquer editor)

Neste projeto, foram inseridas as seguintes linhas no arquivo *profile*:

- LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/local/j2sdk1.4.2_13/classes/parport
- CLASSPATH=.:%CLASSPATH%:$JAVA_HOME/lib:$JAVA_HOME/bin:$JAVA_HOME/jre/bin:$JAVA_HOME/classes/parport
- CLASSPATH=.:%CLASSPATH%:$JAVA_HOME/classes

- export LD_LIBRARY_PATH JAVA_HOME CATALINA_HOME CLASSPATH PATH

Nesta dissertação foram configurados o servidor TOMCAT, biblioteca parport e o Java, as configurações podem variar de acordo com a instalação. As variáveis de ambiente foram configuradas da seguinte forma neste trabalho:

- JAVA_HOME=/usr/local/j2sdk1.4.2_13
- CATALINA_HOME=/opt/tomcat
- CLASSPATH=$CATALINA_HOME/common/lib/servlet-api.jar:.:$CLASSPATH
- CLASSPATH=$CATALINA_HOME/common/lib/jsp-api.jar:$CLASSPATH
- CLASSPATH=$CLASSPATH:/$JAVA_HOME/lib:$JAVA_HOME/bin:$JAVA_HOME/jre/bin:$JAVA_HOME/classes/parport
- CLASSPATH=$CLASSPATH:/$JAVA_HOME/classes
- PATH=$JAVA_HOME/bin:$PATH
- LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/local/j2sdk1.4.2_13/classes/parport
- export LD_LIBRARY_PATH JAVA_HOME CATALINA_HOME CLASSPATH PATH
- PATH="/usr/local/bin:/usr/bin:/bin:/usr/local/j2sdk1.4.2_13/lib:/usr/local/j2sdk1.4.2_13/bin: /usr/local/j2sdk1.4.2_13/jre/bin

B2 - Criação do Contexto de Desenvolvimento da Servlet

Para executar seus servlets e JSPs, você precisa colocá-los dentro de um contexto de aplicação web (ServletContext). Cada contexto é uma unidade de aplicação web Java (servlet/JSP) que possui suas próprias configurações.

Para organizar o desenvolvimento, é interessante criar um contexto novo e ativar sua opção reloadable (recarga automática das classes modificadas). Para isso, faça o seguinte:

- **Estrutura de diretórios**

Crie um diretório que será a sua estrutura de desenvolvimento web Java. Uma organização simples de diretórios é apresentada de acordo com a Figura B.1:

Figura B.1 - Estrutura de diretório das servlets.

- **Criar contexto de aplicação web**

Usar um dos mecanismos de Deployment Automático de Aplicação do Host no Tomcat. Este é o meio mais recomendado, pois permite configuração automática do contexto na inicialização e atualização dinâmica da aplicação web durante a execução do Tomcat. Uma das formas mais fácil é criando um arquivo XML separado com as configurações do contexto.

- **Criando o arquivo dev.xml:**

Criaremos um arquivo XML, para o novo contexto chamado "dev". O arquivo deve ficar em:

Tomcat 5:

CATALINA_HOME/conf/Catalina/localhost/dev.xml

Catalina é o mecanismo e localhost (máquina local) é o hostname padrão.

Crie o arquivo `dev.xml` na localização já descrita, com o conteúdo do quadro a seguir. O conteúdo é a definição do `Context` e seu `Logger`, precedida pela tag de identificação de arquivo XML:

```
<? xml version="1.0" encoding="iso-8859-1"?>
<Context path="/dev" docBase="C:/dir/dev/web" reloadable="true" crossContext="true" debug="3">
<Logger className="org.apache.catalina.logger.FileLogger" prefix="localhost_dev_log." suffix=".txt" timestamp="true" verbosity="4" />
</Context>
```

- **Configurar contexto: web.xml**

O arquivo WEB-INF/web.xml é o descritor do contexto de aplicação web, segundo a especificação Java Servlet/J2EE. As informações nele contidas são as configurações específicas da aplicação.

Crie o arquivo web.xml descritor para o novo contexto de aplicação web criado, dentro do diretório dev/web/WEB-INF/. Um conteúdo mínimo para ele, com as configurações apresentadas, é listado a seguir.

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="2.4" xmlns="http://java.sun.com/xml/ns/j2ee"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee
http://java.sun.com/xml/ns/j2ee/web-app_2_4.xsd">
    <display-name>ncap</display-name>
    <servlet>
      <servlet-name>ncap</servlet-name>
      <servlet-class>ncap</servlet-class>
    </servlet>
    <servlet-mapping>
      <servlet-name>ncap</servlet-name>
      <url-pattern>/ncap</url-pattern>
    </servlet-mapping>
    <session-config>
      <session-timeout>30</session-timeout>
    </session-config>
    <welcome-file-list>
        <welcome-file> index.jsp</welcome-file>
    </welcome-file-list>
</web-app>
```

Para garantir a ativação do novo contexto criado, reinicie o Tomcat.

Apêndice C - Instalação das Ferramentas para Acesso aos Dispositivos Celulares

C1 - Instalação do PST 6.7 e Configuração do Aparelho Celular

A seguir, apresenta-se uma maneira de desbloquear o aparelho celular modelo Motorola para realização de *upload* de arquivos. O *software* utilizado é o PST 6.7_GENERAL que pode ser baixado do *site* <http://lpssd.dee.feis.unesp.br/*downloads*.php>. Para realizar a instalação, descompacte o arquivo dentro de um diretório desejado, dê um duplo clique no arquivo executável e siga os procedimentos de instalação necessários. Após a instalação execute os seguintes procedimentos:

- Conecte o cabo USB na entrada do micro-computador e no aparelho celular;
- Abra o *software* PST 6.7_GENERAL, entre em NEW e depois selecione KJava Files;
- Entre em Menu Phone e clique em READ FROMPHONE;
- Quando a leitura estiver finalizada, os dois botões ficaram habilitados, clique no botão ENABLE/DISABLE JAVA APP LOADER MENU, note que abaixo deste botão estará mostrando o status atual do aparelho, após clicar, aparecerá JAVA APP MENU IS NOW ENABLED;
- Operação finalizada, seu menu Java estará habilitado;
- Reinicie o sistema operacional do aparelho celular e verifique a opção citada acima.

Na Figura C.1 apresenta-se a interface do programa.

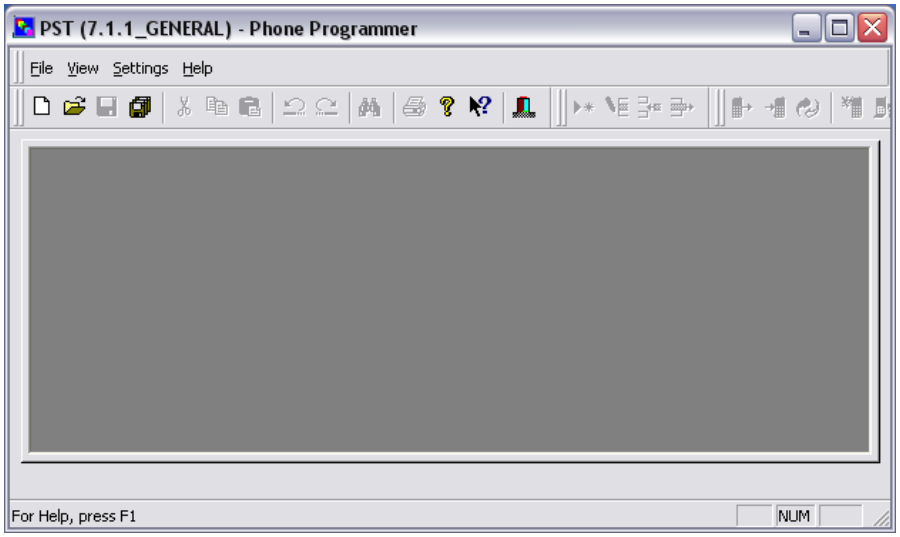

Figura C.1 - Interface do software PST (7.1.1_General) – Phone Programmer.

C2 - Instalação do MIDway 2.8 e Upload dos Arquivos JAR e JAD

A seguir, apresenta-se a configuração do *software* utilizado para enviar aplicativos para o dispositivo móvel modelo Motorola, a plataforma utilizada para a programação do celular foi Windows XP. Ao baixar o *software* através do *site* <http://zemaracutaia.no.sapo.pt/Motorola_MIDway_v28.zip> , deve-se descompactar dentro de uma pasta de acordo com a preferência. O MIDway 2.8 é um *software* utilizado para enviar aplicativos para o celular e funciona mediante a porta COM, porém utilizando o cabo USB AxB-mini de 5 pinos. Na Figura C.2 apresenta-se o modelo de cabo utilizado para a transferência de arquivos.

Figura C.2 - Modelo de cabo USB.

A seguir as etapas para a configuração do cabo USB AxB-mini para *upload* dos arquivos para os dispositivos celulares modelo Motorola:

- Conecte o cabo USB no dispositivo celular e no microcomputador;
- Vá à opção Meu Computador → Gerenciador de *Hardware*;

- Encontre a opção Modems e verifique que as opções Motorola USB Modem esteja ligado;
- Clique com o botão direito e entre em Propriedades;
- Clique na aba Avançadas;
- Clique em Configurações Avançadas de Porta;
- Escolha a porta COM que será utilizada pelo cabo USB;
- No dispositivo celular vá em: Configurações → Configurações Java → Load App Java, será pedido para inserir o cabo USB, então insira o cabo, após inserir, aparecerá JAL Link ATIVO.
- No microcomputador, abra o programa Midway, clique no ultimo botão (Config) e selecione a porta COM que é utilizada pelo cabo USB. Não altere as opções de velocidade.
- Clique em File;
- Open File;
- Selecione o arquivo JAD a enviar;
- Clique em SEND FILE, será enviado o cabeçalho do arquivo para o celular.
- Após o recebimento do cabeçalho, apareceram no celular as informações do Java.
- Clique em *Download*;
- Será feito o *upload* do arquivo e instalado automaticamente no aparelho celular.

Na Figura C.3 apresenta-se a interface de configuração MIDway 2.8, onde é disponível a porta de comunicação e a taxa de transferência.

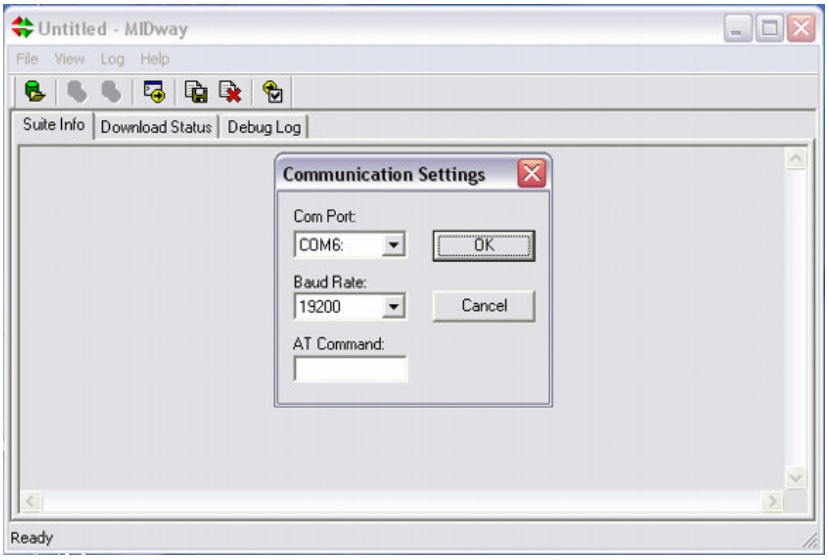

Figura C.3 - Programa MIDway 2.8.

Ao realizar a conexão entre o computador e o celular, o arquivo JAD criado pelo simulador deve ser referenciado. O *software* MIDway apresenta todas as características do aplicativo a ser enviado para o dispositivo celular. Na Figura C.4 apresenta-se a interface e a configuração do *software* MATIUDE. Ao requisitar o envio do *software*, o aparelho celular requer autorização ao usuário permitindo o *download* ou cancelamento do envio do arquivo.

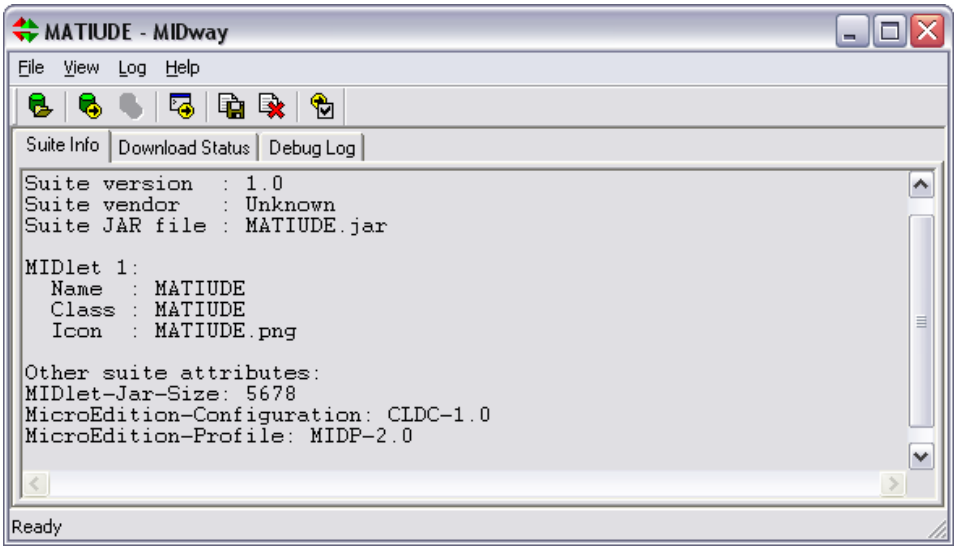

Figura C.4 - Configuração do aplicativo apresentado pelo MIDway.

C3 - Configuração do Dispositivo Celular para Acesso a Rede Ethernet

Para realizar a comunicação entre o aparelho celular e a parte lógica do NCAP é necessário realizar algumas configurações no aparelho celular. No presente projeto a operadora utilizada para realização dos testes deste projeto foi a CLARO.

Para realizar as configurações do foi realizada através da interface Sessões Web disponibilizado no aparelho celular. Na Figura C.5 apresenta-se à interface de acesso às configurações do dispositivo.

Figura C.5 - Interface de acesso a configurações da web.

Após a opção Sessões Web o usuário tem a disponibilidade de editar e alterar os campos de configuração de acordo com os dados da operadora. Para a configuração com o operadora CLARO os dados foram:

- Nome – Claro;
- Página Principal – http://www.claro.com.br;
- Tipo de serviço 1 – http;
- Gateway IP 1 – 200.169.126.11;
- Porta – 9201;
- Nome de usuário:
    - wap.claro.com.br - disponibilizado pelo operador da Claro (utilizado para navegação na web);
    - claro.com.br – de acordo com a internet (utilizado para aplicações Java).
- Senha 1 – CLARO.

Na Figura C.6 apresenta-se a interface de configuração do dispositivo celular.

Figura C.6 - Interface de configuração do aparelho celular Motorola C385.

C4 - Modelos de Aparelho com Tecnologia J2ME

A seguir na Tabela C.1, apresenta-se alguns modelos de aparelho celular disponível no mercado, que possui a tecnologia J2ME.

Tabela C.1 – Exemplo de dispositivos móveis com tecnologia J2ME.

Fabricante	Modelo	Tecnologia Sem-fio	Freqüência (MHz)	Software	Interface
Nokia	3560	AMPS, TDMA	800, 1900	MIDP 1.0, CLDC 1.0, Nokia UI API	96x65/12 bits
Nokia	3570	CDMA2000 1X	1900	MIDP 1.0, CLDC 1.0	96x65/2 bits
Nokia	3585	AMPS, CDMA2000 1X	800, 1900	MIDP 1.0, CLDC 1.0	96x65/2 bits
Nokia	3585i	AMPS, CDMA	800, 1900	MIDP 1.0, CLDC 1.0, Nokia UI API	96x65/2 bits
Nokia	3586i	AMPS, CDMA	800, 1900	MIDP 1.0, CLDC 1.0, Nokia UI API	96x65/12 bits
Nokia	3587i	AMPS, CDMA	800, 1900	MIDP 1.0, CLDC 1.0, Nokia UI API	96x65/12 bits
Nokia	3590	GSM	900, 1800	MIDP 1.0, CLDC 1.0, Nokia UI API	96x65/1 bit
Nokia	3595	GSM	850, 1900	MIDP 1.0, WMA 1.0 / JSR 120, CLDC 1.0, Nokia UI API	96x65/12 bits
Nokia	3600	GSM/GPRS	850, 1900	MMAPI / JSR 135, MIDP 1.0, WMA 1.0 / JSR	176x208/12 bits

				120, CLDC 1.0, Nokia UI API	
Nokia	3650	GSM	900, 1800, 1900	MMAPI / JSR 135, MIDP 1.0, WMA 1.0 / JSR 120, CLDC 1.0, Nokia UI API	176x208/12 bits
Nokia	3660	GSM	900, 1800, 1900	MIDP 1.0, MMAPI 1.1 / JSR 135, WMA 1.0 / JSR 120, CLDC 1.0, Nokia UI API	176x208/16 bits
Nokia	5100	GSM	900, 1800, 1900	MIDP 1.0, CLDC 1.0, Nokia UI API	128x128/12 bits
Nokia	6010	GSM	850, 1900	MIDP 1.0, WMA 1.0 / JSR 120, CLDC 1.0, Nokia UI API	96x65/12 bits
Nokia	6012	AMPS, CDMA	800, 1900	MIDP 1.0, CLDC 1.0, Nokia UI API	96x65/12 bits
Nokia	6015	AMPS, CDMA	800, 1900	MIDP 1.0, CLDC 1.0, Nokia UI API	96x65/12 bits
Motorola	Accompli 009	GSM/GPRS	900, 1800, 1900	MIDP 1.0, CLDC 1.0	240x160/8 bits
Motorola	C300	GSM	900, 1800	MIDP 1.0, CLDC 1.0	98x64/12 bits

Motorola	C370/C 450	GSM/GPRS	850, 900, 1800, 1900	MIDP 1.0, CLDC 1.0	96x64/1 bit
Motorola	C380	GSM/GPRS	900, 1800, 1900	MIDP 2.0, MMAPI 1.1 / JSR 135, WMA 2.0 / JSR 205	128x128/1 bit
Motorola	C550	GSM/GPRS	900, 1800	MIDP 1.0, CLDC 1.0	96x64/1 bit
Motorola	C650	GSM/GPRS	900, 1800, 1900	MIDP 2.0, MMAPI 1.1 / JSR 135, WMA 2.0 / JSR 205	128x128/1
Motorola	C975			MIDP 1.0, CLDC 1.0	176x220/1 bit
Motorola	E380	GSM/GPRS	900, 1800	MIDP 1.0, CLDC 1.0	96x65/12 bits
Motorola	E398	GSM/GPRS	850, 900, 1800, 1900	MIDP 2.0, MMAPI 1.1 / JSR 135, CLDC 1.0, WMA 2.0 / JSR 205	176x220/16 bits
Motorola	V180	GSM/GPRS	850, 900, 1800, 1900	MIDP 1.0, MIDP 2.0, MMAPI 1.1 / JSR 135, WMA 2.0 / JSR 205	128x128/12 bits
Motorola	V220	GSM/GPRS	850, 900, 1800, 1900	MIDP 1.0, MIDP 2.0, MMAPI 1.1 / JSR 135, WMA 2.0 / JSR 205	128x128/12 bits

Motorola	V300	GSM/GPRS	850, 900, 1800, 1900	MIDP 2.0, MMAPI 1.1 / JSR 135, WMA 2.0 / JSR 205	176x220/12 bits
Motorola	V323	AMPS	800, 1900	MIDP 1.0, CLDC 1.0	176x220/12 bits
Motorola	V325	AMPS	800, 1900	MIDP 1.0, CLDC 1.0	176x220/12 bits
Motorola	V360	GSM/GPRS	850, 900, 1800, 1900	MIDP 1.0, Bluetooth / JSR 82, CLDC 1.0	176x220/12 bits
Motorola	V500	GSM/GPRS	850, 900, 1800, 1900	MIDP 1.0, MIDP 2.0, MMAPI 1.1 / JSR 135, WMA 2.0 / JSR 205	176x220/12 bits
Motorola	V550	GSM/GPRS	850, 900, 1800, 1900	MIDP 1.0, MIDP 2.0, MMAPI 1.1 / JSR 135, WMA 2.0 / JSR 205	176x220/12 bits
Motorola	V600	GSM/GPRS	850, 900, 1800, 1900	MIDP 2.0, MMAPI 1.1 / JSR 135, CLDC 1.0, WMA 2.0 / JSR 205	176x220/16 bits

C5 - Tabela de Preço

A Tabela C.2 – apresenta os valores de transferência de dados de acordo com a quantidade dados (R$/Mbyte), tecnologia de transmissão e os serviços dos planos pós ou pré-pagos (30). Nesta dissertação, a pesquisa foi realizada no mês de abril de 2007.

Tabela C.2 – Tabela de preços de acordo com a operadora.

Fonte: UCEL – Universo Celular

Operadora	Tecnologia	Pós (MB)	Pré (MB)
Vivo	CDMA 1x ou EVDO	R$ 6,00	R$ 6,00
Claro	GPRS ou EDGE	R$ 6,00	R$ 6,00
Tim	GPRS ou EDGE	R$ 5,99	R$ 15,73
Oi	GPRS	R$ 8,00	R$ 8,00
Amazônia Celular	GPRS ou EDGE	R$ 8,30	R$ 8,30
Telemig Celular	GPRS ou EDGE	R$ 8,30	R$ 8,30
BrT GSM	GPRS ou EDGE	R$ 6,00	R$ 6,00
CTBC Telecom	GPRS	R$ 8,00	R$ 14,00
Sercomtel	GPRS	R$ 7,00	R$ 12,00

Livros Grátis

(http://www.livrosgratis.com.br)

Milhares de Livros para Download:

Baixar livros de Literatura
Baixar livros de Literatura de Cordel
Baixar livros de Literatura Infantil
Baixar livros de Matemática
Baixar livros de Medicina
Baixar livros de Medicina Veterinária
Baixar livros de Meio Ambiente
Baixar livros de Meteorologia
Baixar Monografias e TCC
Baixar livros Multidisciplinar
Baixar livros de Música
Baixar livros de Psicologia
Baixar livros de Química
Baixar livros de Saúde Coletiva
Baixar livros de Serviço Social
Baixar livros de Sociologia
Baixar livros de Teologia
Baixar livros de Trabalho
Baixar livros de Turismo