UNIVERSIDADE FEDERAL DO RIO DE JANEIRO
ESCOLA POLITÉCNICA
DEPARTAMENTO DE ENGENHARIA ELÉTRICA

LABORATÓRIO MODULAR DE ELETRÔNICA DE POTÊNCIA MICROPROCESSADO

DANIEL SANTOS DE OLIVEIRA CABRAL

Rio de Janeiro, RJ - Brasil
Maio de 2008

LABORATÓRIO MODULAR DE ELETRÔNICA DE POTÊNCIA MICROPROCESSADO

DANIEL SANTOS DE OLIVEIRA CABRAL

PROJETO SUBMETIDO AO CORPO DOCENTE DO DEPARTAMENTO DE ENGENHARIA ELÉTRICA DA ESCOLA POLITÉCNICA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE ENGENHEIRO ELETRICISTA.

Aprovado por:

_____

Luís Guilherme B. Rolim, Dr.-Ing. (orientador)

_____

Richard M. Stephan, Dr.-Ing.

_____

Walter I. Suemitsu, Dr.Ing.

Rio de Janeiro, RJ - Brasil

Maio de 2008

Dedicatória

Dedico este trabalho a uma pessoa muito especial que hoje não pode mais estar aqui reunida comigo, mas que tenho certeza que em algum lugar distante e em paz está olhando por mim: minha querida mãe Consuelo Santos Ribeiro (☼ 1954 – † 2002), que mesmo com uma vida de intensa luta contra uma das maiores enfermidades do nosso século sempre esteve pronta a dar carinho, alegria e jamais deixar de lado o sonho de formar seus filhos cidadãos aptos a contribuirem para um futuro melhor da humanidade.

Professora e pedagoga, profissional de exemplo e sempre dedicada à educação, fez do incentivo ao estudo uma das principais lembranças que levarei para sempre dela, e por isso esta dedicatória tão justa para que ela se sinta bem em saber que seu filho finalmente chega ao término de mais um importante trecho desta longa caminhada que é a vida. O conhecimento e a capacidade do ser humano de recebê-lo e transmití-lo é a maior dádiva da vida de uma pessoa e por isso deixo aqui esta sincera dedicatória.

Agradecimentos

Agradeço em primeiro lugar a Deus, que permitiu hoje estar aqui em condições de realizar este trabalho e completar finalmente, após muitos anos de esforço, minha tão sonhada graduação em engenharia elétrica.

Agradeço também à minha querida esposa Joana, a quem amo muito e é uma pessoa que sempre esteve ao meu lado em todos os momentos, principalmente os mais difíceis, sempre me incentivando e dando motivação para jamais desistir. Sem ela certamente este trabalho não teria chegado ao fim.

Ao meu pai Gérson, minhas avós Dora e Berenice, meu tio Marcelo e meu irmão Diogo, deixo aqui também meu sincero agradecimento pelo apoio dado ao longo de tantos anos e por terem sempre ficado ao meu lado dando o suporte necessário.

Não podendo tabém deixar de agradecer aos meus professores da Escola Politécnica da UFRJ, funcionários e colegas do DEE e principalmente ao meu professor e orientador Luís Guilherme Barbosa Rolim, pela confiança em meu trabalho e em minha capacidade de vencer este desafio proposto, atividade que muito me fez aprender e me mostrou que jamais devemos desistir de nossos objetivos.

E finalizando meus eternos amigos de infância André, André Luís, Ricardo e Thiago, demais amigos e também meus colegas de trabalho do Ministério Público do RJ que também me ajudaram bastante nessa caminhada.

Resumo

Este trabalho propõe o desenvolvimento de uma base de software robusta, escalável e flexível capaz de permitir a reconstrução da atual bancada didática de eletrônica de potência, desenvolvida na década de 90 pelo Departamento de Engenharia Elétrica da UFRJ (DEE) para ser utilizada em seu curso de graduação. Atualmente o hardware e o software disponíveis ficaram ultrapassados e alguns problemas foram surgindo levando à necessidade de modernizar tal equipamento, que é hoje um dos poucos desta categoria disponíveis no Brasil.

Utilizando idéias propostas e experiências adquiridas ao longo dos últimos anos em outros trabalhos e artigos acadêmicos e focando no desafio de manter o custo de implementação pequeno mesmo com a utilização do que existe de mais moderno em termos de tecnologia da informação e controle microprocessado, este projeto tem por meta corrigir as atuais limitações da bancada didática da UFRJ e oferecer uma nova experiência aos seus usuários, os alunos.

Os assuntos aqui estudados formam uma rede de conhecimentos heterogêneos e interconectados que abrangem desde tópicos específicos da eletrônica de potência até tópicos hoje também presentes e necessários à vida profissional de um engenheiro eletricista como: microprocessadores e microcontroladores, programação orientada a objetos, bancos de dados e interface *web* / internet, processamento de sinais e comunicação de dados. Faz parte também deste estudo a utilização do sistema operacional Linux, o desenvolvimento de um protocolo proprietário, a montagem e configuração completa de um ambiente de software distribuído e outro embarcado e a integração entre eles.

Sumário

Lista de abreviações

A/D – Analógico / Digital

API – *Application Programming Interface* (interface de programação de aplicativos)

CA – Corrente Alternada

CC – Corrente Contínua

CRC – *Cyclic Redundancy Code* (código de redundância cíclica)

DAO – *Data Access Object* (objeto de acesso a dados)

DSP – *Digital Signal Processor* (processador de sinais digitais)

ESCP – *Elepot Serial Comunication Protocol* (protocolo de comunicação serial elepot)

FFT – *Fast Fourier Transform* (transformada rápida de Fourier)

GNU – *GNU is not Unix* (GNU não é Unix)[1]

GPL – *General Public License* (licença pública de uso geral)

HTML – *Hiper-Text Markup Language* (liguagem de marcação de hiper-texto)

HTTP – *Hiper-Text Transfer Protocol* (protocolo de transferência de hiper-texto)

I/O – *Input/Output* (entrada/saída)

IDE – *Integrated Development Environment* (ambiente integrado de desenvolvimento)

IGBT – *Insulated Gate Bipolar Transistor* (transistor bipolar de porta embutida)

IP – *Internet Protocol* (protocolo internet)[2]

J2EE – *Java 2 Enterprise Edition* (edição corporativa do Java versão 2)

JDBC – *Java Database Connectivity* (conectividade com bancos de dados em Java)

JDK – *Java Development Kit* (kit de desenvolvimento Java)

JSP – *JavaServer Pages* (tecnologia Java para elaboração de páginas *web* dinâmicas)

JSTL – *JSP Standard Tag Library* (biblioteca padrão de *tags* JSP)

LED – *Light Emitting Diode* (diodo emissor de luz)

([1]) GNU é um conjunto de ferramentas e programas de computador criado pela Free Software Foundation com o intuito de criar um sistema operacional completo e livre. Devido ao atraso no desenvolvimento de seu núcleo o mesmo acabou utilizando o núcleo do Linux, que em conjunto com este pacote de programas e ferramentas formou um completo sistema operacional, hoje conhecido como GNU/Linux. O significado da sigla recursiva "GNU is not Unix" advém do fato que o GNU por ser um software livre não possui qualquer cópia de código do sistema operacional Unix.

([2]) O IP é um dos diversos protocolos utilizados na internet. Existe um outro conceito que é o "endereço IP" e que é um número que representa unicamente um equipamento (normalmente um roteador ou um microcomputador) na internet, como uma espécie de "endereço" daquele equipamento. Sua semelhança com o IP é que este número funciona na mesma camada do protocolo IP e seguinto os padrões deste protocolo.

Lista de abreviações (cont.)

MOSFET – *Metal Oxide Semiconductor Field Effect Transistor* (transistor de metal-óxido semicondutor de efeito de campo)

MVC – *Model-View-Controller* (arquitetura modelo-visão-controle)

NA – Normalmente Aberto (contato)

PC – *Personal Computer* (computador pessoal, criado pela empresa IBM)

PLL – *Phase-Locked Loop* (ciclo de fase travada)

RAM – *Random Access Memory* (memória de acesso aleatório)

RMS – *Root Mean Square* (valor quadrático médio, ou valor eficaz)

SGBD – Sistema Gerenciador de Banco de Dados

SQL – *Structured Query Language* (linguagem de consulta estruturada)

TI – Tecnologia da Informação

TTL – *Transistor-Transistor Logic* (lógica transistor-transistor)

UFRJ – Universidade Federal do Rio de Janeiro

UML – *Unified Modeling Language* (linguagem de modelagem unificada)

URL – *Uniform Resource Locator* (localizador uniforme de recursos, como por exemplo, um endereço *web*)

USB – *Universal Serial Bus* (barramento serial universal)

Lista de figuras

Lista de figuras (cont.)

Lista de figuras (cont.)

Lista de tabelas

1 – Introdução

A disciplina “laboratório de eletrônica de potência”, obrigatória no curso de graduação em engenharia elétrica da UFRJ, é de grande importância para a formação dos alunos permitindo que todo o conhecimento teórico adquirido no curso de eletrônica de potência seja colocado em prática, criando motivação e confrontando os alunos com situações e problemas que somente surgem em atividades práticas.

Quando se fala em energia atualmente não se pode deixar de lado os temas eficiência e conservação, que cada dia tornam-se mais importantes e cruciais em qualquer sistema energético, visto que uma das maiores preocupações da humanidade para as próximas décadas é ser capaz de usufruir de maneira otimizada e ambientalmente correta os recursos energéticos cada vez mais escassos do planeta. A eletrônica de potência cumpre o papel de estudar e desenvolver soluções energéticas capazes de atender a este cenário, sendo então imprescindível que um engenheiro hoje domine seus principais conceitos.

Os laboratórios de eletrônica de potência da UFRJ são alguns dos melhores e mais modernos do país e contam com um excelente corpo docente, composto de renomados professores e pesquisadores. Para os cursos de graduação, a própria universidade desenvolveu seus equipamentos didáticos que permitem aos alunos uma rápida montagem dos conversores. Além disso, fica à disposição dos interessados, um excelente acervo técnico sobre o desenvolvimento e funcionamento dos equipamentos disponíveis no laboratório. Dentre os vários trabalhos existentes, o principal deles, base deste projeto, foi a tese de mestrado Laboratório Modular de Eletrônica de Potência [1]. Há ainda algumas atividades interessantes e pioneiras, como o Laboratório Remoto de Eletrônica de Potência [2].

Com o passar dos anos, o laboratório de eletrônica de potência do curso de graduação em engenharia elétrica da UFRJ, daqui para a frente chamando apenas de laboratório de eletrônica de potência, foi passando por algumas mudanças e enfrentando algumas limitações técnicas, tornando-se então objeto de estudo para que o mesmo pudesse ser aprimorado para permitir uma didática mais eficiente. Este trabalho propõe um novo ciclo de evolução para o laboratório, aproveitando idéias bem sucedidas de trabalhos anteriores.

Ao longo deste trabalho, será proposta, projetada, implementada e homologada uma nova versão da bancada didática do laboratório de eletrônica de potência, composta também de módulos independentes e com controle feito por software. Uma significativa diferença entre esta nova versão e as anteriores é que o módulo de interfaces não mais ficará no microcomputador (e na placa de aquisição de dados) mas sim em um outro software, embarcado, e que ficará residente em um microcontrolador com funções de DSP. A idéia do uso do microcontrolador é poder transferir as funções temporizadas críticas do microprocessador do PC para um outro microprocessador dedicado a esta função, eliminando assim uma das principais limitações atuais do laboratório. Com essa nova estrutura será possível, por exemplo, uma implementação futura, por software, de um PLL para as experiências com retificadores, de modo a eliminar o problema da falta de sincronismo causado pelo excesso de conteúdo harmônico da rede.

Aproveitando ainda que uma nova versão da bancada didática será desenvolvida pelo Departamento de Engenharia Elétrica da UFRJ, o módulo de controle, hoje já todo implementado em software, será reescrito utilizando tecnologia de ponta em sua concepção (substituindo o atual software em Turbo Pascal desenvolvido em 1993 por um completamente novo em Java [18]). Os objetivos desta substituição são: i) utilização de código orientado a objetos, aumentando assim a reusabilidade e a flexibilidade para modificações futuras além do fato que programas de computador escritos de forma estruturada (procedural) estão em pleno desuso atualmente; ii) independência de plataforma (sistema operacional principalmente); iii) utilização da *web* como interface com o usuário, hoje a interface mais intuitiva para qualquer pessoa no mundo e que permite que o módulo de controle seja desmembrado em servidor, que pode ser isolado para uso exclusivo dos professores, e máquinas clientes mais modestas de modo que com um único navegador *web* os alunos possam acessar o laboratório.

Além disso, com este módulo de controle será futuramente mais fácil a implementação de um laboratório remoto [2] onde os alunos poderão acessar e ver os resultados das experiências em suas próprias casas. Como incentivo para trabalhos futuros este assunto e a apresentação de alguns dos seus problemas práticos serão detalhados adiante. Fará parte ainda da nova bancada didática um módulo adicional, de administração do laboratório, com cadastros de alunos, experimentos, configurações da bancada didática, entre outros recursos, que auxiliarão o controle e acesso do mesmo pelos alunos e professores. Também serão citados ao longo deste trabalho diversas possibilidades de estudos futuros envolvendo este módulo. Apesar de não fazer parte do

foco principal, toda interface com o usuário dos módulos de controle e administrativo foi desenvolvida usando um *design* temático e elegante, típico de aplicações *web*, dando assim um excelente acabamento visual à aplicação. Obviamente este *design* poderá ser facilmente modificado no futuro de modo a atender possíveis requisitos do laboratório ou do Departamento de Engenharia Elétrica da UFRJ.

O trabalho está então estruturado da seguinte forma: o capítulo 2 faz um resumo de como funciona e está estruturado atualmente o laboratório de eletrônica de potência. Sobre este tópico não serão fornecidos muitos detalhes já que tal conteúdo está muito bem tratado e detalhado nos trabalhos de Rolim [1] e Fernandes [2]. Este capítulo possui ainda um outro tópico sobre a realidade do laboratório, que atualmente está desativado por motivos técnicos e as experiências vêm sendo realizadas por meio de simulações em computador. O capítulo 3 explica quais são as idéias propostas neste trabalho e confronta tais idéias com as possíveis dificuldades e limitações técnicas que serão encontradas. Os capítulos 4 e 5 são o projeto propriamente dito. Eles foram assim divididos de modo a tratar isoladamente as duas grandes "partes" do trabalho. O capítulo 4 é dedicado ao microcontrolador, seu hardware e software e o capítulo 5 é sobre a aplicação Java, tanto do módulo administrativo como do módulo de controle. Neste capítulo, será tratado em detalhes o projeto do banco de dados e da aplicação *web* com suas respectivas tecnologias empregadas. O capítulo 6 é a implementação de todo o sistema, desde a preparação para a montagem física da nova bancada didática do laboratório até o desenvolvimento do software em linguagem C embarcado no microcontrolador e da aplicação *web* em Java. O capítulo 7 é a homologação do laboratório, ou seja, o sistema será testado e validado de acordo com as necessidades dos cinco experiementos do atual roteiro de atividades do laboratório [3] e o capítulo 8 é a conclusão do trabalho e propostas para projetos futuros.

Assim como nos demais trabalhos de Rolim [1] e Fernandes [2], será mantida a filosofia de software livre e, apesar de extensa, uma listagem de todos os códigos-fontes desenvolvidos será anexada neste trabalho. Acompanhará também o presente trabalho um CDROM contendo todo o material desenvolvido.

2 – O laboratório de eletrônica de potência

2.1 – Apresentação

A disciplina laboratório de eletrônica de potência utiliza para a realização de suas cinco práticas de laboratório uma bancada didática desenvolvida na própria UFRJ [1]. Esta bancada didática baseia-se na idéia de desenvolver módulos independentes que conectados de maneira apropriada permitem a montagem de conversores estáticos em diversas configurações.

O diagrama a seguir (figura 2.1) ilustra como é feita essa divisão em módulos. A bancada didática é composta por: módulo de fontes, módulo de cargas, módulo de chaves, módulo de interfaces e módulo de controle.

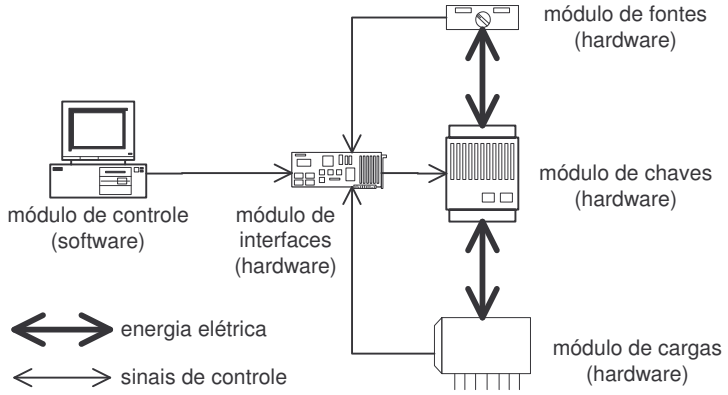


figura 2.1 - Diagrama esquemático da bancada didática

2.1.1 – Módulo de fontes

Este módulo é composto pelas fontes de alimentação de potência. Fazem parte deste módulo fontes lineares ou chaveadas CC e CA independentes. Estas fontes são conectadas no módulo de chaves e são elas que alimentam os conversores que são montados pelos alunos no laboratório. Por questões de segurança, essas fontes são ajustadas para níveis de tensão pequenos, como por exemplo 30V.

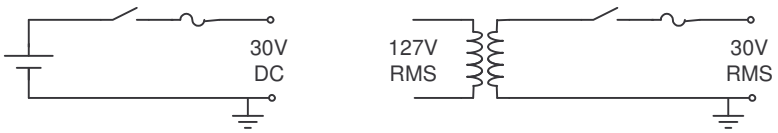


figura 2.2 – Módulo de fontes

2.1.2 – Módulo de cargas

Este módulo é composto por algumas das cargas típicas de conversores estáticos, como cargas resistivas e indutivas, além de motores CC e CA. As cargas são escolhidas dentre as disponíveis no módulo pelos alunos e de acordo com o roteiro da experiência em questão. São então conectadas ao módulo de chaves. As cargas serão alimentadas pelos conversores montados pelos alunos no laboratório.

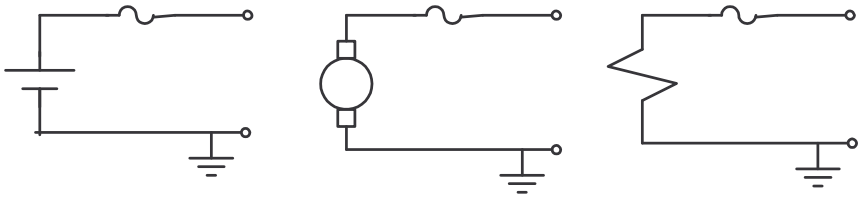


figura 2.3 – Módulo de cargas

2.1.3 – Módulo de chaves

Este é um dos principais módulos da bancada didática. É neste módulo que são conectados os dispositivos semicondutores apropriados para a montagem dos diversos conversores estáticos, além dos demais módulos. Sua função é implementar uma matriz de chaveamento 3x3 [1], permitindo assim a montagem de todos os conversores contidos no roteiro do laboratório [3] e mais alguns, como inversores e retificadores trifásicos, em caso de expansões futuras.

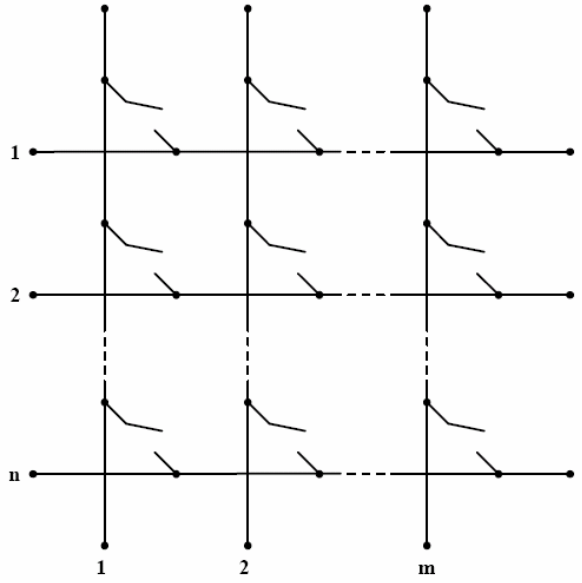


figura 2.4 – Matriz de chaveamento
(fonte: Laboratório Modular de Eletrônica de Potência)

As chaves semicondutoras utilizadas também fazem parte deste módulo e são montadas em unidades independentes no formato de "cartuchos". Segundo Rolim [1], "no estudo de um conversor estático através da matriz-chaveamento, as chaves semicondutoras que compõem este conversor são tratadas de uma maneira generalizada como chaves ideais bidirecionais em corrente e tensão". Ou seja, ficam à disposição dos alunos, unidades completas de chaves usando os diversos dispositivos semicondutores, como diodos, tiristores, MOSFETs de potência ou IGBTs. Mesmo com características diferentes, estas unidades funcionam basicamente da mesma forma: recebem sinais de comando digitais no padrão TTL e operam no modo chaveado (corte e saturação), permitindo ou não a passagem de corrente elétrica para a carga proveniente da fonte de potência. Nestas unidades estão incorporados circuitos analógicos que convertem os sinais digitais em níveis de corrente e tensão necessários ao disparo do dispositivo semicondutor e acopladores ópticos para isolar a parte de comando da parte de potência.

figura 2.5 – Unidades de chaves semicondutoras

O módulo de chaves é então composto das unidades de chaves e de uma base de montagem onde fica a matriz de chaveamento, com todos os bornes de conexão necessários para as diversas montagens. Tanto as unidades de chaves como esta base já passaram por algumas modificações ao longo dos anos, buscando sempre um *design* mais adequado tanto do ponto de vista ergonômico como prático. Em seu trabalho, Rolim [1] cita os quatro itens que resumem esta necessidade: "segurança, facilidade de uso, facilidade de manutenção e expansibilidade". A figura 2.6 mostra as duas versões atuais da base de montagem do módulo de chaves.

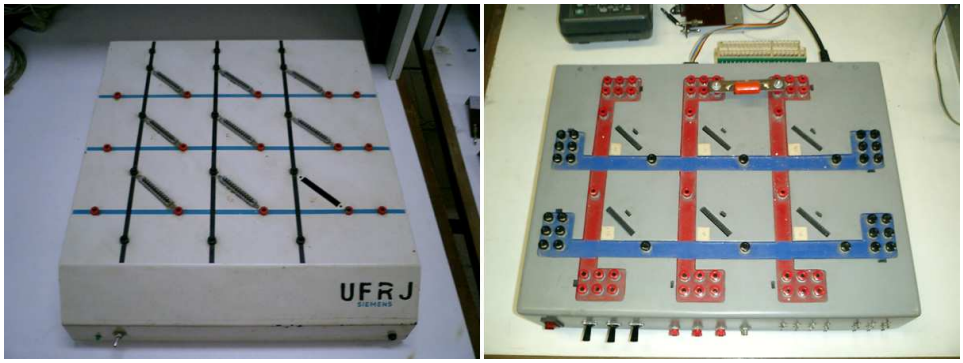


figura 2.6 – Base de montagem do módulo de chaves

Atualmente está sendo desenvolvido pelos professores e técnicos do laboratório uma nova versão da base de montagem, mais compacta ainda que suas antecessoras e com *design* mais eficiente. A figura 2.7 ilustra esta nova base de montagem, que futuramente será adotada no laboratório de eletrônica de potência.

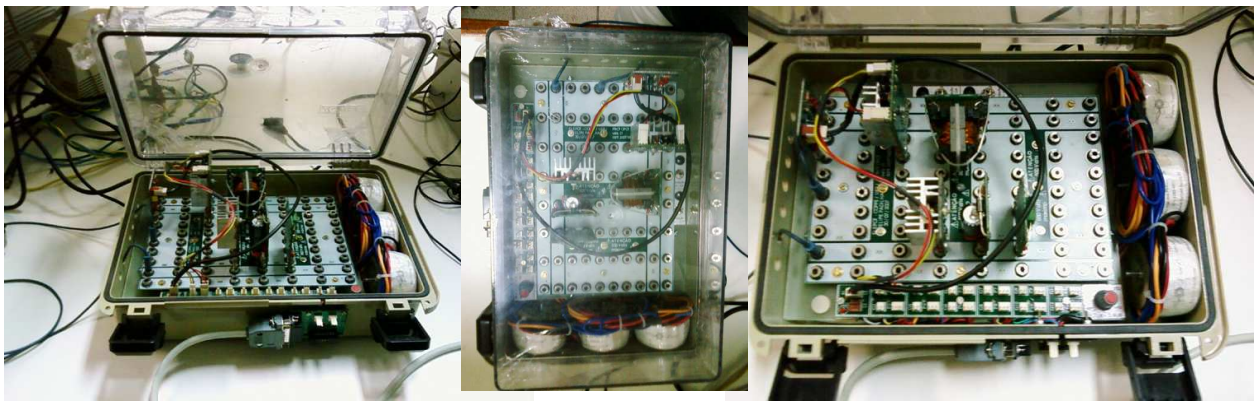


figura 2.7 – Nova base de montagem do módulo de chaves

2.1.4 – Módulo de interfaces

O módulo de interfaces é o responsável por estabelecer o intercâmbio de informações e suas devidas conversões de sinais entre o módulo de controle e o módulo de chaves. Ele transforma dados de controle em sinais de controle apropriados e vice-versa (figura 2.1), além de possuir outras funções como captura e conversão de sinais analógicos usados na realimentação de alguns dos conversores e sincronização de sinais por meio de temporização. Para desempenhar esta função é utilizada uma placa de aquisição de dados. A placa deve possuir pelo menos alguns canais de saída digitais e analógicos, canais de entrada analógicos, um conversor A/D e um temporizador que opere nos modos sincronizado e não sincronizado [1].

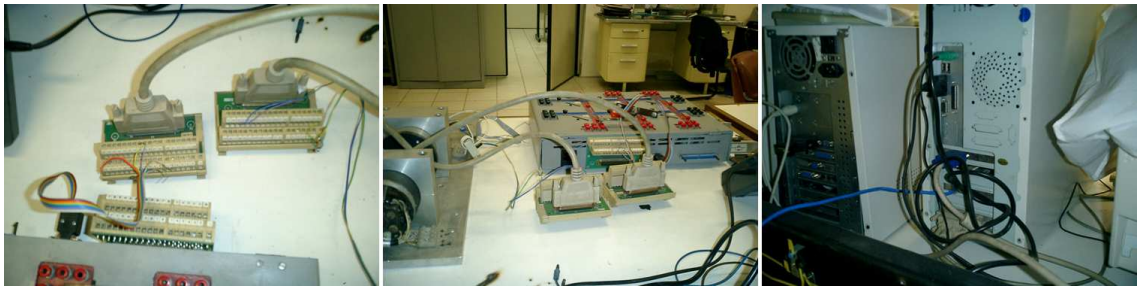


figura 2.8 – Módulo de interfaces

O módulo de interfaces implementado atualmente no laboratório representa um ponto crítico da bancada didática devido aos seguintes fatores: i) custo extremamente elevado das placas de aquisição de dados; ii) dependência do microprocessador do PC, o

que torna o módulo também dependente do sistema operacional (não desejável) o que obriga o uso de um relógio/temporizador externo, acrescentando maior complexidade; iii) necessidade de um circuito de sincronismo externo, atualmente com implementação analógica, mais sujeita a ruídos – este último representa hoje um grande entrave ao uso prático da bancada didática no laboratório.

Uma das principais mudanças que este trabalho propõe é então a substituição da atual placa de aquisição de dados por um microcontrolador dedicado. Além de possuir custo muito inferior ao das placas de aquisição, o microcontrolador permitirá integrar em um único hardware todas as funções do módulo de interfaces.

2.1.5 – Módulo de controle

O módulo de controle é implementado em sua totalidade por software em um microcomputador PC (figura 2.9). Atualmente o módulo de controle funciona com um conjunto de programas, separados em módulos independentes, escritos em Turbo Pascal. Cada programa destes é responsável pelo controle de um ou mais experimentos.

figura 2.9 – Módulo de controle

O funcionamento deste módulo sofre então grande influência do sistema operacional do microcomputador. Em sua primeira versão, a bancada didática utilizava o sistema operacional MS-DOS, que é mono-tarefa. Posteriormente passou-se a usar o sistema operacional Windows 95, que é multi-tarefa, uma evolução natural. Apesar desta mudança representar alguns ganhos, como a possibilidade de se utilizar recursos de rede por exemplo, este sistema trouxe alguns problemas como o fato de não ser capaz de priorizar tarefas (processos) do usuário sobre tarefas do próprio sistema operacional, gerando atrasos de processamento na placa de aquisição de dados e em alguns casos compromentendo os sub-sistemas de temporização e sincronismo. Surgiram então

algumas idéias interessantes como, por exemplo, o uso do sistema operacional de tempo real Real Time Linux (uma variação Linux), que além de ser multi-usuário e multi-tarefa é capaz de priorizar a execução de rotinas críticas do usuário consideradas de tempo real [2].

Outra limitação do módulo de controle é o hardware utilizado pela bancada didática: um antigo microcomputador PC com processador Pentium I de 133MHz com pouca memória RAM – um micro já bastante obsoleto há alguns anos.

Partindo então das idéias anteriores e da necessidade de modernização do laboratório, o Departamento de Engenharia Elétrica da UFRJ adquiriu recentemente um novo microcomputador PC, com processador Sempron 2800+ de 2,8GHz (256Kb de memória cache), 512Mb de memória RAM e 20Gb de disco rígido. O gabinete deste novo microcomputador, conforme pode ser observado na figura 2.10, foi especialmente escolhido para uso na bancada didática. Com este hardware será possível a implementação de um servidor J2EE (Java) e do novo software de controle da bancada didática, além do novo módulo administrativo, que possuirá um banco de dados.

figura 2.10 – Novo módulo de controle

Com o proposto neste trabalho, todas as tarefas que dependem de sincronismo e temporização estarão concentradas no microprocessador dedicado (a comunicação com o microcontrolador será feita de forma assíncrona através das portas seriais deste equipamento e do PC utilizando a interface RS-232C) e como todos os softwares de controle e de administração do sistema serão multiplataforma, será possível então utilizar praticamente qualquer sistema operacional no PC. Levando em conta o fator custo, será adotada uma solução GNU Linux. A escolha feita será pelo Kurumin Linux, uma distribuição Linux bem leve e simples de usar, brasileira, e com extensa documentação disponível.

2.2 – Panorama atual

Durante o primeiro semestre letivo do ano de 2006 algumas das dificuldades técnicas existentes no laboratório de eletrônica de potência, já aqui citadas nos itens anteriores, em conjunto com alguns problemas típicos de circuitos eletrônicos que surgem com o passar dos anos como, por exemplo, a oxidação de contatos elétricos e consequente aumento de ruídos e dificuldade de serem estabelecidas boas conexões dos componentes modulares, levaram o laboratório a ser desativado temporariamente para manutenção, já que os alunos estavam tendo muitas dificuldades práticas em realizar seus experimentos.

Ficou então estabelecido, em caráter provisório, um laboratório de eletrônica de potência "virtual", com todas as atividades práticas sendo realizadas somente através do software de simulação PSCAD. Este laboratório virtual, que permanece funcionando até o momento da conclusão deste trabalho, atende às necessidades didáticas básicas do curso porém não permite aos alunos terem o contato mais prático da realidade da eletrônica de potência. Há detalhes que somente podem ser observados e desafios que somente podem ser superados em situações reais, sem as inúmeras aproximações acrescentadas pelos simuladores digitais, por mais fidedignos com o mundo real que eles possam ser.

O macro-objetivo do laboratório é então projetar, desenvolver e implementar uma nova versão da bancada didática, com novo hardware e software. A atualização do hardware está atualmente sendo executada pelo Departamento de Engenharia Elétrica da UFRJ por meio da aquisição de novos equipamentos e criação própria de uma nova base de montagem. Fica então como motivação e objetivo deste trabalho cuidar do projeto, desenvolvimento e implementação do novo software e permitir sua futura integração com o novo hardware da bancada didática, ainda que com algumas adaptações que se façam necessárias.

3 – Propostas e desafios

Partindo então da necessidade e motivação expostas, novas idéias surgiram e suas viabilidades serão estudadas nos próximos tópicos. A escolha das tecnologias empregadas neste projeto foram feitas com base nos seguintes critérios: domínio e experiência da tecnologia por parte dos integrantes e da universidade; custo para aquisição da tecnologia; e por fim, busca por tecnologias internacionalmente consagradas, de qualidade indiscutível e com amplo suporte e documentação.

3.1 – Idéias e propostas

O principal objetivo da nova bancada didática de eletrônica de potência é corrigir algumas das limitações que somente puderam ser observadas com o passar dos anos em uso no laboratório, principalmente no que diz respeito às tarefas temporizadas e com necessidade de sincronismo. A principal idéia para atingir esta meta é o uso de um microprocessador dedicado, operando em um microcontrolador digital e com software embarcado capaz de se comunicar com uma aplicação de controle.

A partir deste objetivo principal, foram surgindo também outras idéias para a modernização do hardware, a substituição do maior número possível de circuitos analógicos por digitais, a utilização de um novo sistema operacional, e a criação de um novo módulo, administrativo, para oferecer um melhor controle das práticas realizadas pelos alunos (utilizando para isso um SGBD).

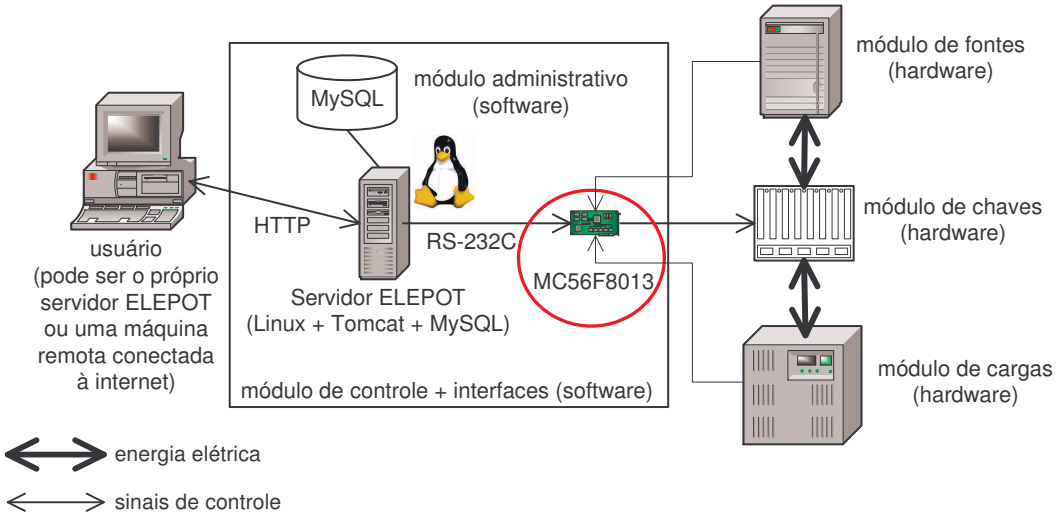


figura 3.1 - Diagrama esquemático da nova bancada didática

3.1.1 – O microcontrolador MC56F8013 da Freescale (Motorola)

A Freescale, divisão de microcontroladores e microprocessadores da Motorola, possui um produto adequado ao projeto da nova bancada didática de eletrônica de potência: o MC56F8013, um microcontrolador de baixo custo e muito completo, com diversos periféricos integrados e funções de processamento digital de sinais. As principais características e periféricos disponíveis neste microcontrolador são:

- possui o eficiente microprocessador de 16-bits 56800E (arquitetura *Harvard*);
- velocidade de 32MHz no núcleo;
- possui funcionalidades de DSP;
- deslocador lógico-aritmético de múltiplos bits;
- acumuladores de 36-bits;
- memória incorporada com 20Kb (sendo 16Kb de memória *flash*);
- 1 módulo PWM com 6 saídas PWM;
- 1 conversor analógico-digital (A/D) de 12-bits com 6 entradas;
- 1 módulo temporizador de 16-bits;
- 1 interface serial padrão RS-232C;
- 26 pinos para uso de portas genéricas de I/O.

Além dos itens já citados, o MC56F8013 é comercializado sobre uma placa de demonstração que possui todos os conectores de acesso aos seus periféricos prontos para uso, LEDs indicadores e programáveis coloridos, uma interface paralela que além de ser usada para programar também permite depuração em tempo real do *softtware* embarcado, entre outras facilidades.

Outro ponto importante é que o MC56F8013 pode ser programado usando-se o excelente ambiente de desenvolvimento integrado (IDE) CodeWarrior, que possui um compilador C de alta qualidade, ferramentas de depuração, componentes de software (*beans*) reutilizáveis e razoável documentação. Com o CodeWarrior fica então possível desenvolver todo o software embarcado diretamente em linguagem C ao invés do típico e complicado *assembly*.

Juntando todas essas características com o preço do produto, ele se torna a opção mais viável para o projeto, sendo assim justificado seu uso daqui por diante.

3.1.2 – O novo microcomputador do laboratório e o Kurumin Linux

O laboratório de eletrônica de potência adquiriu recentemente um novo microcomputador para ser usado na confecção da nova bancada didática. Com este novo equipamento será possível instalar e executar o sistema operacional com os softwares propostos sem quaisquer comprometimentos de desempenho. Apesar de mais moderno, a configuração ainda é muito modesta para os dias atuais mantendo o custo do projeto muito pequeno. Como já citado, o novo microcomputador do laboratório é um PC com processador AMD Sempron modelo 2800+ de 2,8GHz, 512Mb de memória RAM e 20Gb de disco rígido.

Como sistema operacional a escolha foi o GNU Linux, que é um sistema operacional livre e de excelente qualidade. Uma distribuição Linux nacional e muito difundida atualmente é o Kurumin Linux. Sua principal vantagem é ser um sistema operacional Linux voltado para o uso geral e doméstico e, sendo assim, possuir uma curta linha de aprendizado e um pacote de programas bem abrangente de modo que não seja necessário configurar ou instalar muita coisa que não seja apenas o utilizado no projeto. E tudo isso sem abrir mão de todas as características de robustez, segurança e desempenho comuns a qualquer Linux.

A versão do Kurumin utilizada neste projeto será a 4.1, um pouco antiga, porém por ter sido instalada no computador do laboratório de eletrônica de potência logo após sua aquisição, acabou então sendo adotada. Porém, nada impede que qualquer outra versão seja utilizada. Neste caso, basta apenas observar a versão do *kernel* para saber se atende aos pré-requisitos dos softwares a serem instalados. Esta versão 4.1 do Kurumin possui o *kernel* 2.6.8 e dois navegadores *web* diferentes: o Konqueror e o Mozilla Firefox. Mais detalhes sobre o Kurumin Linux podem ser obtidos no sítio oficial na internet [19] ou em Morimoto [5].

3.1.3 – O banco de dados MySQL

Pensar em um sistema hoje que lide com informações é impossível sem junto também pensar em um banco de dados. As informações precisam estar eficientemente organizadas para que no futuro continuem acessíveis e úteis.

O novo módulo administrativo proposto neste trabalho é composto essencialmente de informações sobre os alunos, os experimentos e as configurações da

bancada didática. Numa possível futura expansão poderão ainda ser acrescidas outras informações.

O banco de dados MySQL é hoje uma das soluções mundialmente mais adotadas em sistemas que gerenciam dados pois é um software de código aberto, multiplataforma (com implementações para quase todos os sistemas operacionais existentes), de excelente desempenho e estabilidade apesar da pouca exigência de hardware, fácil de usar e com extensa documentação e suporte. O MySQL é também compatível com a linguagem de consulta a banco de dados SQL.

Para a proposta deste trabalho o MySQL atende perfeitamente sem criar quaisquer ônus financeiros ou técnicos para o projeto, ficando então justificado seu uso daqui por diante.

3.1.4 – A aplicação *web* em Java, o NetBeans e o uso do *framework* Struts

O controle da bancada didática será, conforme já explicado, implementado por software dentro do microcontrolador, porém a interface deste equipamento com o operador da bancada é extremante pobre, com poucas opções para o usuário interagir com o controle. Por este fato será importante existir um segundo componente de software capaz de estabelecer uma camada intermediária entre o operador da bancada (usuário) e o seu sistema de controle (software embarcado). Esta camada deverá ser capaz de comunicar-se com o microcontrolador de forma coerente e precisa, seguindo determinadas regras pré-estabelecidas (chamadas de protocolo), e comunicar-se também com o usuário por meio de uma interface intuitiva e fácil de usar.

O mais apropriado para esta camada é uma aplicação com interface *web*, hoje um padrão mundialmente adotado. E para construir uma aplicação *web* como essa, que também precisará fazer a comunicação com o microcontrolador usando uma interface serial, a melhor opção é o Java, uma completa linguagem de programação orientada a objetos e um ambiente de desenvolvimento rico em recursos, largamente difundido e muito seguro e confiável. Além disso, o pacote de desenvolvimento do Java (JDK) e diversas ferramentas de desenvolvimento e suporte como o NetBeans (IDE) e o servidor de aplicação Apache Tomcat, entre outras, são de livre distribuição, mantendo os custos do projeto extremamente baixos. O Java e suas APIs e ferramentas são multiplataforma, tornando também o projeto mais flexível e independente do sistema operacional.

Com o crescimento constante da complexidade dos sistemas de informação, a metodologia de desenvolvimento estruturado (também conhecido como procedural) tornou-se obsoleta devido a sua enorme dificuldade em reaproveitar e modificar partes de um sistema complexo. Para resolver essa questão surgiu a orientação a objetos, que melhora em muito os aspectos acima citados. Porém, com a utilização cada vez maior de sistemas distribuídos em camadas (como é o caso da *web*) e o desenvolvimento desses sistemas sendo compartilhado com milhares de pessoas ao redor do mundo por meio da internet, surgiu também a necessidade de se ter uma padronização das técnicas utilizadas chamada de *design patterns*, ou padrões de projeto. Os *design patterns* são conjuntos de técnicas e algoritmos usados para solucionar problemas comuns a diversos sistemas. Os *design patterns* mais importantes utilizam a arquitetura de software MVC, hoje adotada mundialmente como um padrão de fato. O MVC é um conjunto pré-estabelecido de componentes de software e seus inter-relacionamentos, sendo que a sua principal característica é manter o isolamento e a independência entre as suas três camadas: de apresentação, ou visualização (*view*); de controle (*controller*) e de dados, ou modelo (*model*).

Usando então técnicas e arquiteturas de software comuns fica mais fácil manter, compartilhar e reaproveitar um software. A partir daí, uma estrutura de software usando esses componentes e técnicas comuns pode ser estabelecida. Seu nome é *framework*. É quase o mesmo conceito das antigas bibliotecas de software (*toolkits*), porém, é muito mais abrangente já que não envolve apenas implementações de funções reaproveitáveis, onde cada desenvolvedor escolhe sua maneira de usá-las, mas sim todo um conjunto de suporte, orientado a objetos e de alto nível com as melhores técnicas de programação envolvidas para facilitar o desenvolvimento e deixar o analista ou programador mais focado nas regras do negócio do que em detalhes técnicos de implementação, tornando ainda o código mais reutilizável, flexível e legível a outras pessoas. Um *framework* muito utilizado em aplicações J2EE (a plataforma Java para aplicações distribuídas, que compreende o conjunto servidor de aplicação + aplicação *web*) é o Struts. O Struts é um *framework* que implementa o MVC, ou seja, mantém todo o código da aplicação bem separado em camadas de apresentação (a parte que o usuário interage com a aplicação), de modelo (onde os dados e regras de negócio são processados) e controle (que organiza o funcionamento da aplicação e cuida da troca de dados entre as outras duas camadas). O Struts facilita em muito o trabalho pois isenta o programador de implementar uma

série de rotinas comuns de uma aplicação *web*, como por exemplo, realizar o processamento dos cabeçalhos HTTP e manter o código bem estruturado e legível.

Por todas as razões acima apresentadas, será então utilizado neste projeto, para o desenvolvimento da aplicação *web* responsável pela interface do usuário com os módulos de controle e administrativo, e também capaz de realizar a comunicação com o microcontrolador, a seguinte estrutura:

a) Ambiente de desenvolvimento, onde a aplicação será construída e testada:

- Sistema operacional: Windows XP ou Kurumin Linux
- Ferramentas de desenvolvimento: JDK versão 1.5 e NetBeans IDE versão 5.0
- Ferramentas de apoio: *driver* JDBC para MySQL e RXTX versão 2.1.7 [1]
- *Framework* de desenvolvimento: Struts versão 1.2.7
- Servidor de aplicação: Bundled Apache Tomcat versão 5.5.9 [2]

b) Ambiente de execução, onde a aplicação será utilizada, ou seja, a bancada didática:

- Sistema operacional: Kurumin Linux
- Servidor de aplicação: Apache Tomcat 6.0.10

O diagrama abaixo mostra resumidamente como ficará estruturada a aplicação *web*. Alguns detalhes deste diagrama que ainda não foram explicados e a descrição dos módulos do pacote br.ufrj.dee.elepot serão mostrados ao longo deste texto.

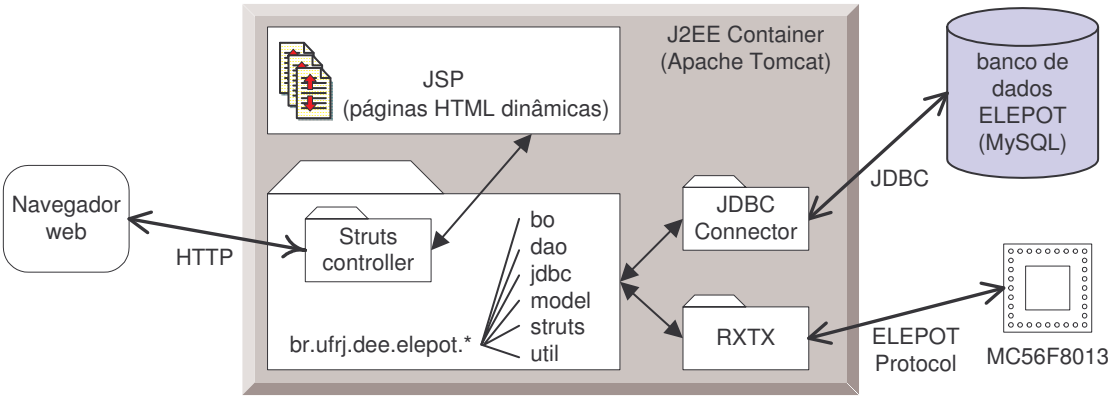


figura 3.2 - Diagrama esquemático da aplicação *web*

([1]) O RXTX é uma API Java para comunicação baseada na API oficial da Sun JavaComm, porém, ao contrário desta é fácil de configurar, possui drivers de comunicação para diversos sistemas operacionais e não necessita de arquivos de configuração, o que a torna mais adequada a sistemas *web* e multiplataforma.

([2]) O Bundled Tomcat é uma versão simplificada do servidor de aplicação Apache Tomcat que vem embutida no NetBeans. Apesar de mais limitado é ideal para a etapa de desenvolvimento e realização de testes.

3.2 – O desafio do novo laboratório

Em seu trabalho, Rolim [1] conclui sobre a atual bancada didática do laboratório de eletrônica de potência: "foi projetado e construído um protótipo segundo a concepção desenvolvida, e foram apresentados diversos exemplos de utilização, a título de validação experimental. Os exemplos apresentados comprovaram a versatilidade do equipamento, tendo sido feitas implementações de diferentes funções de conversão com grande rapidez e facilidade. Contudo, os testes de validação revelaram também algumas limitações no projeto do protótipo, principalmente no que se refere à temporização. A demora do computador do módulo de controle no processamento de interrupções limita a freqüência de acionamento das chaves e restringe as possibilidades de implementação de malhas de realimentação rápidas, como por exemplo para controle de corrente".

A principal limitação da bancada didática está em seus processos temporizados. Construir um sistema onde a temporização possui resposta muito rápida e que seja independente do restante do controle é o maior desafio deste trabalho. Por esta razão foi escolhido manter o controle temporizado da bancada em um microprocessador dedicado e fazer com que sua comunicação com o restante do controle fosse feito de forma assíncrona.

Porém, outros pontos também serão um grande desafio neste projeto, a citar:

- desenvolver um módulo administrativo onde as experiências sejam suficientemente flexíveis para que os usuários mantenedores (professores) possam realizar algumas modificações em seu roteiro;

- implementar no sistema um mecanismo capaz de controlar também o acesso à bancada didática, impedindo por exemplo o acesso simultâneo de mais de um usuário sobre o módulo de chaves;

- desenvolver uma aplicação que siga as mais eficientes e modernas técnicas tornando o sistema flexível (capaz de sofrer modificações sem grande esforço ou impacto), escalável (permitindo um futuro crescimento de suas funcionalidades) e reaproveitável (mesmo que o sistema precise de grandes modificações boa parte do código é bem aproveitada);

- montar um sistema que seja seguro e robusto o suficiente para permitir a plena realização das práticas laboratoriais, com máxima eliminação de ruídos externos;

- desenvolver um protocolo de comunicação assíncrono capaz de estabelecer as regras corretas de comunicação entre o microcontrolador e o microcomputador, permitindo a plena troca de informações e comandos, assim como a correta interpretação dos números, já que diferentes processadores lidam com números de forma diferente também;

- Implementar o software de controle embarcado usando ao máximo os recursos oferecidos no microcontrolador e ao mesmo tempo sendo capaz de contornar todas as limitações do equipamento;

- Trabalhar com aritmética de ponto fixo no software embarcado mesmo com o uso de funções trigonométricas (tipicamente de ponto flutuante), necessárias por exemplo no controle de inversores. O uso da aritmética de ponto fixo é de extrema importância quando se trabalha com a necessidade de alto desempenho do processador, conforme explica Rolim [8] em seu estudo;

- Tornar possível a futura utilização de um PLL por software – um PLL é um sistema implementável por software ou hardware capaz de manter sincronizados dois sinais senoidais usando a passagem pelo zero por meio da extração de uma réplica "limpa" de um sinal com ruídos. O PLL será capaz de eliminar o circuito de sincronismo externo com implementação analógica que existe atualmente e é a causa de um grande entrave ao uso da bancada didática no laboratório. Devido à sua grande complexidade, este trabalho será deixado como sugestão para um estudo mais aprofundado;

- Testar e homologar o controle dos cinco experimentos do laboratório de eletrônica de potência do curso de graduação em engenharia elétrica da UFRJ, para que este trabalho não fique apenas como uma análise teórica do problema mas se torne uma realidade viável para o uso prático do laboratório pelos alunos.

3.3 – Novo roteiro de experimentos do laboratório

O roteiro de experimentos do laboratório de eletrônica de potência, usado pelos alunos em suas práticas foi escrito em 2004. Uma das metas da nova bancada didática é permitir a flexibilidade para que o roteiro seja futuramente alterado, já que normalmente o material didático dos alunos é revisto periodicamente. Para tal é necessário então que o sistema como um todo possua as seguintes características:

a) A aplicação embarcada deverá ser composta de módulos de software capazes de controlar cada um dos 3 tipos de conversores[1] utilizados no curso de graduação de eletrônica de potência, a saber: CC-CC (*buck-boost* e variações), CC-CA (inversor) e CA-CC (retificador). Cada um desses módulos não poderá ser dependente de detalhes específicos do roteiro atual, de modo que alterando-se o experimento (na aplicação *web*) seja possível utilizar o mesmo software no microcontrolador.

b) A aplicação *web* deverá também ser composta de um conjunto de classes capazes de representar de modo mais genérico possível os conversores, permitindo que mudanças no roteiro não exijam reescrita de código-fonte.

Com as características acima cumpridas, ficará então possibilitado ao professor modificar os experimentos sem alterar as aplicações, bastando editar os arquivos das páginas *web* correspondentes.

Uma novidade na bancada didática é o módulo administrativo, que gerencia os cadastros de alunos e experimentos. Para facilitar o acesso do aluno ao material didático e modernizar sua forma de consulta, o roteiro de experimentos foi inserido no módulo administrativo da bancada didática (cadastro de experimentos), ficando possível ao professor não apenas editar o experimento em si, mas atualizar o conteúdo escrito do roteiro utilizando os vários recursos do HTML como *hiperlinks*, imagens e animações gráficas, entre muitos outros.

([1]) Os conversores CA-CA (ciclo-conversores) não fazem parte do conteúdo programático do curso de graduação da UFRJ, sendo assim, fica como sugestão uma futura implementação desde conversor na bancada didática, já que o sistema será bem modularizado.

3.4 – Hardware da nova bancada didática

Resumindo as idéias propostas, a nova bancada didática será então composta do seguinte hardware:

- Um microcomputador PC com uma porta serial disponível;
- Um microcontrolador MC56F8013, conectado ao PC por meio da porta serial. Para a programação e depuração do software embarcado poderá ser necessário também o uso da porta paralela, conforme será visto mais adiante;
- Uma base de montagem (módulo de chaves), conectada ao microcontrolador por meio de um cabo de dados próprio, a serem especialmente confeccionados pelo Departamento de Engenharia Elétrica da UFRJ;
- Chaves semicondutoras de potência com *slot* de encaixe compatível com a base de montagem (serão aproveitadas as mesmas chaves atualmente disponíveis no laboratório);
- Circuitos analógicos e outros dispositivos auxiliares e de medição como cargas (resistores, indutores, motores elétricos), fontes de tensão, transformadores, osciloscópio, filtros etc.

Para manter a compatibilidade com os componentes que serão reaproveitados, alguns detalhes deverão ser levados em conta, como por exemplo, o fato das chaves trabalharem somente com níveis lógicos TTL.

Devido ao foco principal deste trabalho ser apenas o desenvolvimento do software necessário à nova bancada didática, ficará considerado que este novo hardware que está sendo construído pelo Departamento de Engenharia Elétrica da UFRJ seguirá os mesmos padrões do atual, sendo assim mínimo o impacto quando o novo equipamento entrar em operação.

Possíveis melhorias poderão ser feitas posteriormente neste hardware, como por exemplo a integração física do microcontrolador junto da base de montagem, ou ainda, melhorias nos cabos de controle utilizando conectores especiais.

4 – Controle microprocessado

Neste capítulo será visto como será projetado o controle microprocessado da bancada didática. Conforme já explicado, o sistema utilizará o microcontrolador MC56F8013 da Freescale. Nada impede porém que em futuras expansões do sistema este modelo de microcontrolador não possa ser substituído por outro com mais recursos e/ou custo mais adequado. O capítulo se divide em 3 partes: uma que trata do hardware do microcontrolador (periféricos usados, configurações, pinagens etc.) – seçao 4.1; uma outra que trata do protocolo de comunicação que será utilizado – seção 4.2; e por fim, uma parte que trata do software, ou seja, o programa embarcado que ficará sendo executado no microcontrolador – seção 4.3.

4.1 – Hardware

O microcontrolador MC56F8013 utilizado neste projeto é fabricado sobre uma placa que permite fácil acesso a seus periféricos. Essa placa é chamada de placa de demonstração do microcontrolador. A sua principal vantagem é tornar dispensável o manuseio de dispositivos eletrônicos de pequena escala, realização de soldas, uso de *proto-boards* e adaptadores. O acesso aos principais recursos do microcontrolador está disponível na placa, permitindo a redução do tamanho físico do conjunto e redução do nível de ruído externo. Os itens mais importantes estão mostrados na figura e tabela seguintes (4.1).

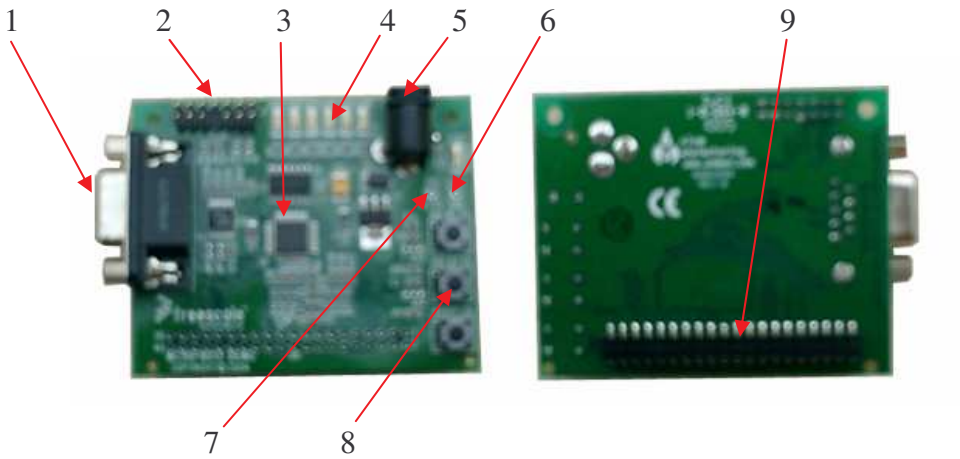


figura 4.1 – Placa de demonstração do microcontrolador MC56F8013

Item	Descrição
1	Conector serial padrão RS-232C
2	Conector JTAG
3	Microcontrolador MC56F8013
4	LEDs genéricos (várias cores)
5	Conector de alimentação (+9V_{CC} / 450mA)
6	Pino de referência "nível baixo" (*ground*) → 0V
7	Pino de referência "nível alto" → 3,3V
8	Botões genéricos (*push-buttons*)
9	Conector das portas de I/O e demais periféricos (*daughter card connector*)

tabela 4.1 – Principais elementos da placa de demonstração do microcontrolador MC56F8013

A subseções a seguir contém informações mais detalhadas sobre alguns dos itens listados. Para detalhes técnicos mais aprofundados consultar [11].

4.1.1 – Conector serial

Será utilizado para conectar o MC56F8013 ao PC e permitir a comunicação entre o programa embarcado e a aplicação *web*. Por utilizar a interface RS-232C, a placa de demonstração do microcontrolador já possui um conversor interno que transforma os níveis de tensão elétrica do periférico SCI (*Serial Communications Interface*) em níveis de tensão do padrão RS-232C, compatível com as portas seriais de micros PC. Para que se estabeleça a comunicação em nível de hardware é necessário apenas que os pinos de transmissão (TxD) e recepção (RxD) sejam conectados por meio de um cabo aos pinos de recepção e transmissão, respectivamente, da porta serial do micro PC.

Pin #	Signal	Pin #	Signal
1	Jumper to 6 & 4	6	Jumper to 1 & 4
2	TXD	7	NC
3	RXD	8	NC
4	Jumper to 1 & 6	9	NC
5	GND		

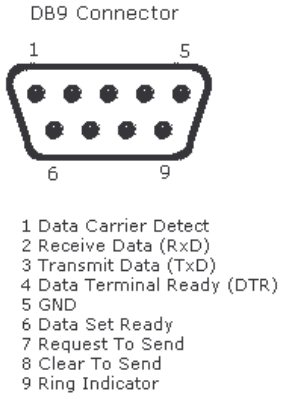


figura 4.2 – Pinagem da porta serial do microcontrolador (esquerda) e do PC (direita)
(fonte: 56F8013 Demonstration Board User Guide e http://www.electronicsteacher.com/)

De acordo com a figura 4.2 é possível notar que a pino 2 do conector serial do MC56F8013 deve ser ligado ao pino 2 do PC (TxD ←→ RxD) e o pino 3 de um com o pino 3 do outro também (RxD ←→ TxD), fechando assim o circuito de comunicação.

Como ambos os conectores são do tipo 9 pinos (padrão DB9) um cabo serial padrão sem inversão de pinos pode ser usado. Normalmente os cabos seriais conhecidos como *data-link* possuem os pinos 2 e 3 invertidos pois são destinados à conexão de dois micros apenas – este tipo de cabo não funcionará. Um cabo que não possui inversão de pinos são os cabos seriais de extensão (usado em *mouse* por exemplo). Outra vantagem do cabo de extensão é que ele normalmente possui um conector macho e um conector fêmea, que casa perfeitamente com a placa de demonstração do microcontrolador, cujo conector é fêmea e com qualquer microcomputador PC-compatível, cujo conector serial padrão DB9 é macho, dispensando o uso de adaptadores auxiliares.

4.1.2 – Conector JTAG

O conector JTAG será utilizado para a programação do microcontrolador por meio do CodeWarrior. Para ser ligado ao PC ele utiliza um adaptador, chamado de adaptador JTAG (fornecido com o kit do MC56F8013) e um cabo paralelo de 25 pinos (DB25), o mesmo utilizado em algumas impressoras. A utilização do conector JTAG é necessária apenas durante as etapas de desenvolvimento e depuração do programa.

É por meio deste conector que o CodeWarrior transfere os dados do programa (*download*) e monitora o mapa de memória do microcontrolador, permindo ao programador utilizar os recursos de depuração.

4.1.3 – *Daughter card connector*

Este conector é o mais importante da placa de demonstração, pois permite acesso aos principais periféricos do microcontrolador como por exemplo o módulo PWM com suas 6 saídas, o conversor analógico-digital (A/D) com suas 6 entradas, as diversas portas genéricas de I/O, entre outros.

Este conector possui 40 pinos e a utilização destes pinos fica a critério de cada aplicação sendo então necessária a confecção de cabo próprio – muitos desses pinos inclusive compartilham funções diferentes.

A figura a seguir descreve a pinagem completa deste conector. A configuração de quais funções do conector serão utilizadas é feita via software, conforme será visto na seção 4.3.

Pin #	Signal	Pin #	Signal
1	+3.3V	2	NC
3	GND	4	GPIOA7 / $\overline{\text{RESET}}$ / V_{PP}
5	GPIOB7 / TXD / SCL	6	NC
7	GPIOB6 / RXD / SDA / CLKIN	8	NC
9	GPIOA0 / PWM0	10	GPIOC0 / ANA0
11	GPIOA1 / PWM1	12	GPIOC1 / ANA1
13	GPIOB4 / T0 / CLK0	14	GPIOC2 / V_{REFH} / ANA2
15	GPIOB5 / T1 / FAULT3	16	NC
17	GPIOB3 / MOSI / T3	18	GPIOC4 / ANB0
19	GPIOB2 / MISO / T2	20	GPIOC5 / ANB1
21	GPIOB0 / SCLK / SCL	22	GPIOC6 / V_{REFL} / ANB2
23	GPIOB1 / $\overline{\text{SS}}$ / SDA	24	NC
25	GPIOD0 / TDI	26	GPIOB1 / $\overline{\text{SS}}$ / SDA
27	GPIOD1 / TDO	28	GPIOB0 / SCLK / SCL
29	GPIOD2 / TCK	30	GPIOA2 / PWM2
31	GPIOD3 / TMS	32	GPIOA3 / PWM3
33	GPIOA6 / FAULT0	34	GPIOA4 / PWM4 / FAULT1 / T2
35	NC	36	GPIOA5 / PWM5 / FAULT2 / T3
37	NC	38	NC
39	NC	40	NC

figura 4.3 – Pinagem do *daughter card connector*
(fonte: 56F8013 Demonstration Board User Guide)

Para o projeto da bancada didática este conector será então utilizado para ligar o microcontrolador com o módulo de chaves por meio das saídas PWM e saídas de I/O genéricas em cabo desenvolvido no laboratório.

Em uma implementação futura de um circuito de sincronização digital dentro do microcontrolador (PLL) outros periféricos deverão ser usados, como por exemplo o conversor analógico-digital (neste caso a comunicação será no sentido inverso, ou seja, com os sinais sendo recebidos pelo microcontrolador).

4.2 – Protocolo de comunicação

Para estabelecer a comunicação entre o MC56F8013 e o microcomputador PC é preciso que determinadas regras sejam definidas. São essas regras que irão definir, por exemplo, o que significa cada bit que é transmitido, levando em consideração inclusive a sua ordem na cadeia de dados, já que os mesmos são transmitidos de forma serial, ou seja, um bit por vez em sequência ordenada. Esse conjunto de regras é conhecido e chamado de "protocolo de comunicação".

Para que dois dispositivos possam comunicar-se com sucesso é necessário então que ambos "entendam" o mesmo protocolo de comunicação definido e a forma de como esse "entendimento" é implementado depende das características de cada dispositivo. O importante é que as regras sejam as mesmas. No caso deste trabalho um exemplo típico é o tratamento numérico. No microcontrolador um número do tipo inteiro é formado por 2 bytes porém na linguagem Java o mesmo número inteiro é formado por 4 bytes. A ordem de como esses bytes compõem o número também é muito importante e pode variar de dispositivo para dispositivo – o nome desta característica dos sistemas de representação numérica é *endianness*.

Um protocolo pode ser síncrono ou assíncrono, uni ou bi-direcional. No caso da bancada didática ele será assíncrono e unidirecional. Assíncrono para que a temporização dos processos no PC não comprometa a temporização dos processos no microcontrolador (se o pacote de dados a trafegar sofrer atrasos não influenciará em nada o programa embarcado) e unidericional pois para cumprir o objetivo desse trabalho é suficiente que a aplicação *web* somente envie comandos ao microcontrolador mas este não precisará se comunicar com a aplicação. Deste modo o protocolo ganha em simplicidade. Como sugestão para um trabalho futuro fica a idéia de tornar o protocolo bi-direcional para que o microcontrolador possa também transmitir informações à aplicação, como por exemplo, transmitindo informações de *status*.

Neste item 4.2 será então definido este protocolo de comunicação que será usado na bancada didática apenas, ou seja, o conjunto de regras e sua estrutura. As suas duas implementações, no MC56F8013 e no microcomputador PC, serão vistas no item 4.3 e no capítulo 5, respectivamente. Nas implementações do protocolo é que serão tratadas as representações numéricas e *endianness* de cada dispositivo, de acordo com suas particularidades.

O protocolo, batizado de ESCP (*Elepot Serial Comunication Protocol*) será composto de uma cadeia de bytes, podendo representar ou não valores numéricos, de acordo com a estrutura (também chamada de pacote) a ser apresentada. Cabe aqui lembrar que um byte representa 256 valores diferentes (1 byte = 8 bits = 2^8 valores).

Para atender a todas as experiências do laboratório de eletrônica de potência e permitir também que futuras expansões sejam feitas, as seguintes informações devem compor o pacote do protocolo:

- 1 byte para indicar qual o conversor a ser usado;
- 1 byte para informar qual o comando desejado;
- 4 números inteiros (formados por 4 bytes cada) com parâmetros do comando;
- 1 byte para cálculo de CRC, muito importante para garantir a integridade do que está sendo transmitido.

Totalizando então, o tamanho do pacote deverá ser pelo menos de 19 bytes. Este tamanho é compatível com o *buffer* de recebimento do MC56F8013 (cuja capacidade é de 65.536 bytes). Porém, há um detalhe importante: experimentalmente ficou comprovado que para receber o pacote de forma íntegra no MC56F8013 é necessário que os bytes que sofrerão conversão de tipo implícito (*cast*) sejam transmitidos primeiro e que sejam sempre transmitidos 4 bytes extras no final do pacote que não podem ser usados para nada pois serão perdidos ao chegarem no microcontrolador. A razão para tal comportamento não é descrita na documentação e muito provavelmente trata-se de um *bug* (erro de programação) do equipamento, contornável ao adotar-se os procedimentos aqui descritos. Considerando então este caso, o pacote ESCP terá 23 bytes, organizados de acordo com a seguinte estrutura:

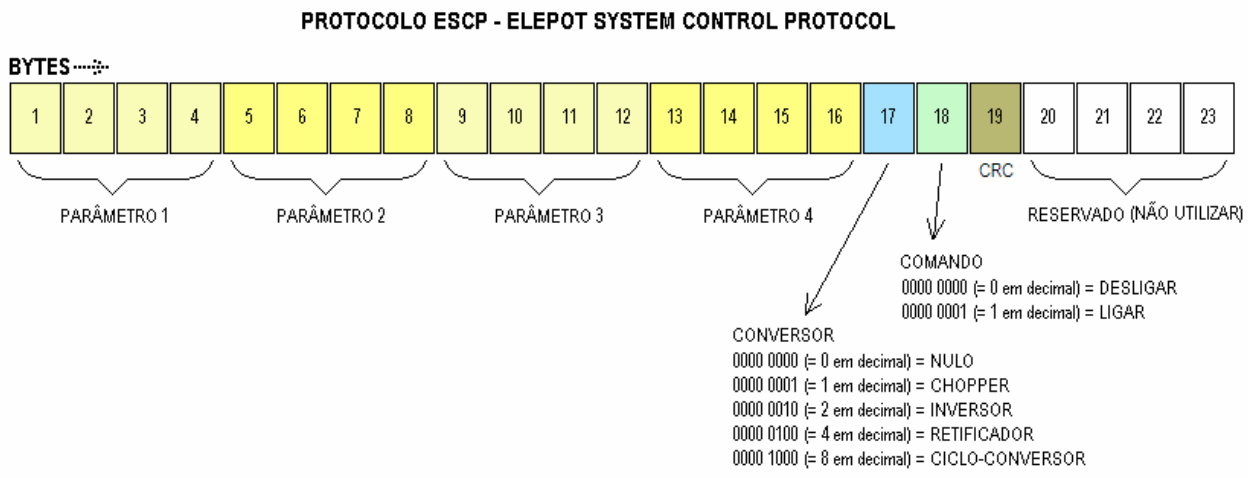


figura 4.4 – Estrutura do protocolo de comunicação

4.3 – Software

O software embarcado, escrito em linguagem C, será composto de um módulo principal, responsável por receber e validar os comandos enviados pela aplicação *web*, organizar o código e as subrotinas de apoio e gerenciar o acesso aos periféricos do microcontrolador, e outros três módulos, responsáveis pela produção dos sinais de controle dos três tipos de conversores estudados nos cinco experimentos do laboratório de eletrônica de potência: *chopper*, inversor e retificador.

Além dos módulos há também os *beans* – componentes de software que acessam os periférios e funções do microcontrolador em alto nível e que são configuráveis por meio de propriedades e utilizados nos módulos de software em métodos e eventos.

A figura 4.5 mostra os arquivos correspondentes aos módulos e os *beans*. A listagem completa de todos os módulos de código (arquivos com extensão ".h" e ".c") encontra-se no apêndice B[1]. As propriedades dos beans serão mostradas na seção 4.3.1.

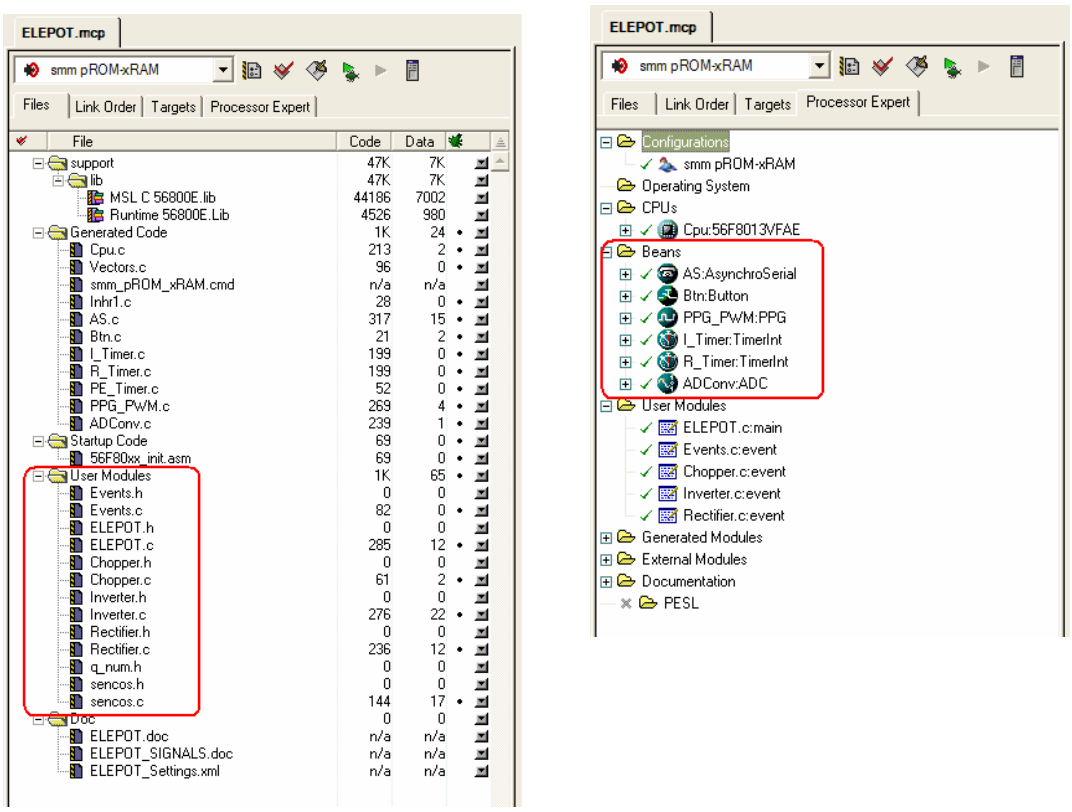


figura 4.5 – Módulos de código do software embarcado (esq.) e *beans* utilizados (dir.)

([1]) O ambiente integrado CodeWarrior trabalha com diversos outros arquivos que são relacionados ao projeto (contém informações de compilação, posição das janelas, pontos de depuração etc.) porém não serão listados porque são automaticamente gerados pelo próprio ambiente. O que de fato interessa são apenas os código-fontes em linguagem C e as configurações dos *beans*.

Diferentemente dos demais códigos deste projeto, todo o software embarcado foi escrito em inglês (inclusive comentários). A razão desta escolha foi porque praticamente não existe documentação em português sobre o microcontrolador MC56F8013 e suas ferramentas de desenvolvimento (ao contrário da linguagem Java, utilizada na aplicação *web*, por exemplo). Em trabalhos futuros, sendo necessária a troca de conhecimento sobre este código (principalmente pela internet) ficará mais fácil com o mesmo escrito somente em inglês.

4.3.1 – Os componentes de software (*beans*)

4.3.1.1 – O componente *Asynchronous Serial Communication*

Este componente permite configurar e utilizar a interface de comunicação serial do microcontrolador. Utilizando este *bean* é possível então implementar a comunicação serial assíncrona entre o programa embarcado e a aplicação *web*. Será utilizado o modo de interrupção que trabalha com eventos gerados por interrupções do microprocessador cada vez que o *buffer* de recepção de dados é completamente preenchido.

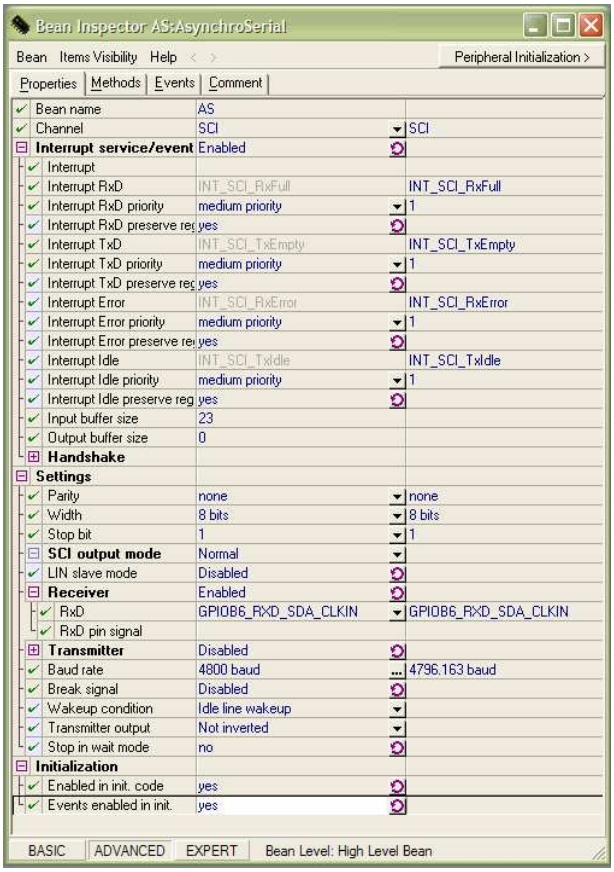


figura 4.6 – Propriedades do *bean Asynchronous Serial Comunication*

Seu principal evento é *OnFullRxBuf*, que é disparado sempre que um pacote de dados é recebido e seu principal método é *RecvBlock*, que permite ler os dados recebidos no *buffer*. Entre as propriedades mais relevantes temos a prioridade da interrupção, ajustada para média, o tamanho do *buffer* (em bytes), ajustado para 23 e a taxa de transmissão dos dados, configurada para 4.800baud. Este valor foi utilizado devido ao resultado de testes práticos em que alguns dos bytes não eram recebidos na velocidade de 9.600baud. Um estudo mais apurado das causas deste problema deverá ser feito posteriormente a fim de permitir o uso de velocidades mais elevadas.

4.3.1.2 – O componente *Button*

Este componente oferece acesso aos botões de pressionamento (tipo NA) localizados na placa de demonstração do microcontrolador. Utilizando este *bean* é possível codificar eventos associados ao pressionamento de um botão. No caso desta aplicação este componente irá implementar uma função de *reset* do conversor, muito importante para que em eventuais situações de travamento ou emergência o conversor possa ser rapidamente desligado sem necessidade de programação.

figura 4.7 – Propriedades do *bean Button*

Seu principal evento é *OnButton*, que é disparado sempre que o botão é apertado. Este evento já possui mecanismos de anti-selo que impede o evento de disparar infinitas vezes mesmo com o operador mantendo o botão pressionado.

4.3.1.3 – O componente *Programable Pulse Generation* (PPG)

O componente PPG oferece acesso ao gerador de sinais do microcontrolador. Os sinais produzidos são pulsos quadrados com frequência e largura configuráveis, ou seja, um sinal PWM, utilizado nos convesores do tipo *chopper* (abaixador ou *buck*, elevador

ou *boost* e abaixador-elevador ou *buck-boost*) dos experimentos 1 e 2 do laboratório de eletrônica de potência. A principal diferença entre o PPG e o *bean* PWM é que somente o PPG permite alteração na frequência e no ciclo de trabalho em tempo de execução (*runtime*) e por essa razão o mesmo será utilizado no módulo *Chopper*.

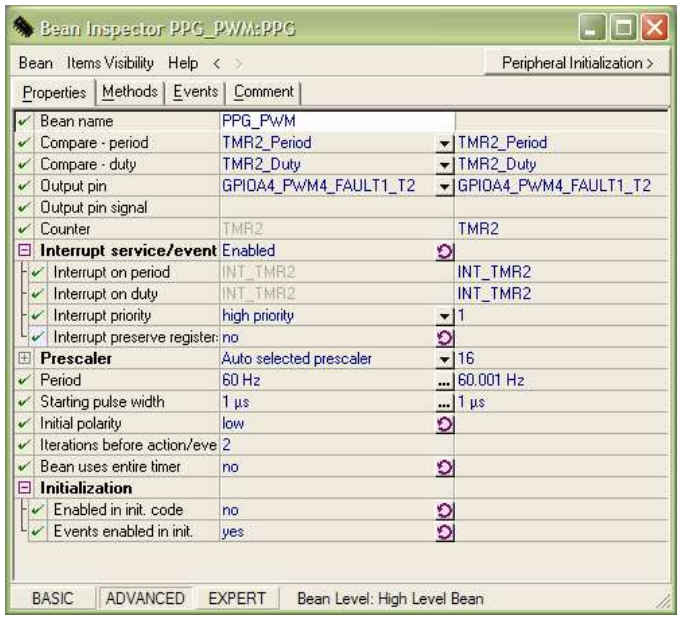


figura 4.8 – Propriedades do *bean* PPG

Seu principal evento é *OnEnd*, disparado sempre que um determinado número de ciclos é completado (configurado na propriedade "*Iteractions before action/event*"). Dentro deste evento é ajustado o valor do ciclo de trabalho (*duty cycle*) do conversor.

Seus principais métodos são *SetFreqHz* e *SetRatio16*, que permitem ajustar, respectivamente, a frequência (em Hz) e o ciclo de trabalho (que é representado por um inteiro de 16 bits, ou seja, um valor de 0 a 65.535 correspondente ao intervalo de 0 a 100%).

Entre as propriedades mais relevantes temos a prioridade da interrupção, ajustada para alta, a polaridade inicial, ajustada para baixa (nível de tensão 0V) e o período do sinal a ser gerado, configurado para o tipo "*from time interval*" com limite inferior em 20Hz e limite superior em 2.000Hz. Estes limites foram assim escolhidos apenas como demonstração. Apesar do nome da propriedade ser "período", a mesma permite que sejam especificados valores de tempo ou frequência.

Um importante detalhe sobre o ajuste do ciclo de trabalho é que quando o componente tem sua polaridade inicial ajustada para baixa (*low*) o valor do ciclo de trabalho representa o tempo em que o sinal ficará também em nível baixo, contrário à convenção tipicamente usada em eletrônica de potência (figura 4.9).

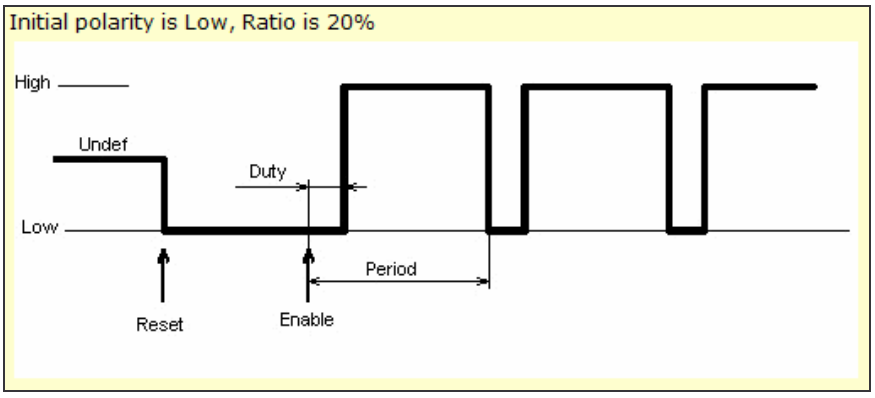


figura 4.9 – Sinais gerados pelo *bean* PPG com polaridade inicial baixa
(fonte: CodeWarrior Help)

Para contornar este detalhe o valor do ciclo de trabalho a ser transmitido para o microcontrolador será o complemento de 1 do valor original. Por exemplo: o usuário deseja um ciclo de trabalho de 80% (0,8). A saída desejada no conversor será então igual à da figura 4.9 e para tal a aplicação deverá determinar o complemento de 1 de 0,8 (que neste caso é 0,2) e colocar o mesmo em Q16, ou seja, *round*(0,2 x 2^{16}) = 13.107, que é a representação de um ciclo de trabalho de 20%.

4.3.1.4 – O componente *Periodic Interrupt* (*TimerInt*)

O componente *TimerInt* permite produzir interrupções periódicas utilizando o *timer* interno do microcontrolador. As interrupções possuem frequência fixa e permitem que algoritmos iterativos sejam executados dentro de seu principal evento *OnInterrupt*. Não há nenhum método em especial para este componente.

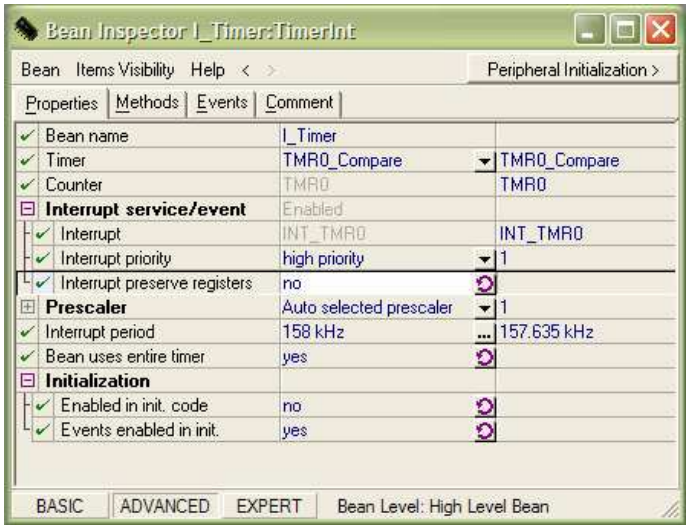


figura 4.10 – Propriedades do *bean TimerInt*

Como propriedade mais importante há apenas o período de interrupção. Para este projeto haverá um *TimerInt* para ser usado no módulo *Inverter* e outro para o módulo *Rectifier*. Dependendo do algoritmo de cada módulo, que pode gastar mais ou menos tempo para ser executado, será escolhido um valor apropriado (o menor possível) para o período de interrupção (ver seções 4.3.4 e 4.3.5).

4.3.1.5 – O componente *A/D Converter*

O componente *A/D Converter* permite utilizar o conversor analógico-digital do microcontrolador. Sua configuração é mais complexa que os demais componentes e sua finalidade no projeto é servir apenas de meio para uso futuro em que um sinal proveniente de um circuito de sincronização analógico ou um sinal analógico a ser usado por um módulo de sincronização via software (um PLL por exemplo) possa ser usado nos experimentos com retificadores. Mais detalhes na seção 4.3.5.

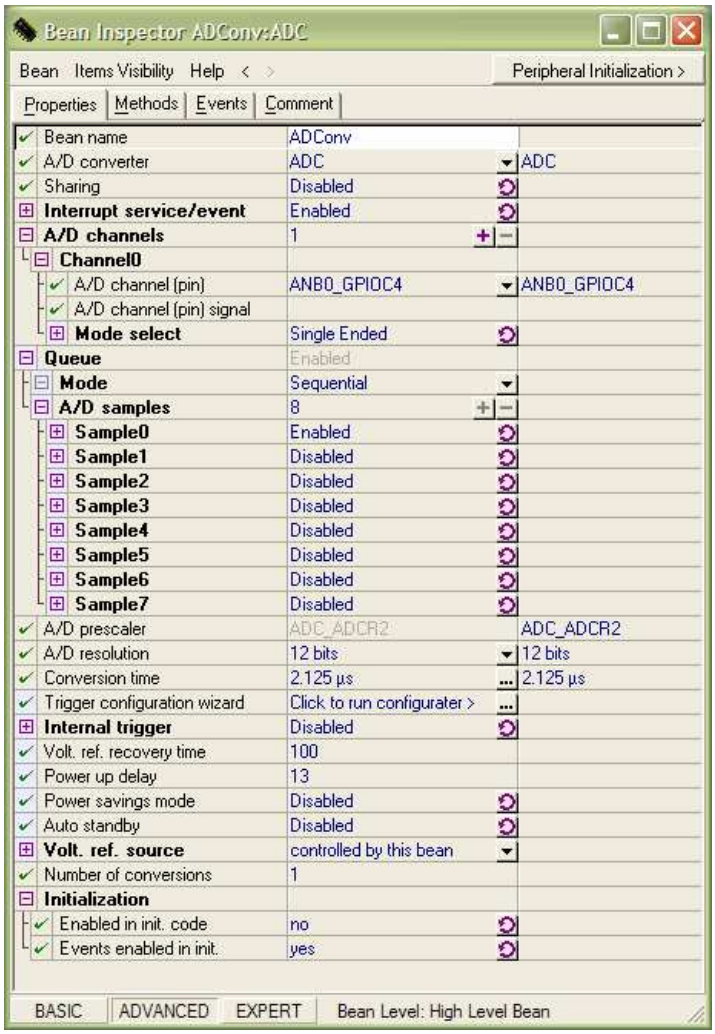


figura 4.11 – Propriedades do *bean A/D Converter*

4.3.2 – Módulo ELEPOT

O módulo ELEPOT possui as definições globais do sistema e métodos e eventos que permitem trabalhar com o tratamento dos pacotes de dados enviados pela porta serial do equipamento, além de procedimentos gerais de inicialização, *reset* etc.

Entre as definições há a estrutura do protocolo ESCP, endereços de memória para acesso a determinadas portas de I/O (e LEDs) e máscaras de bits que são utilizadas em quase todos os módulos.

Os eventos do módulo ELEPOT, localizados no arquivo "Events.c" tratam da recepção de dados pela porta serial e o pressionamento do botão de *reset*. Os métodos deste módulo são resumidamente descritos a seguir:

- O método *initProgram* limpa qualquer conteúdo na variável global que representa o pacote de dados, acende o LED vermelho, apaga os demais e efetua o desligamento de todos os conversores;

- O método *main* (obrigatório) apenas inicia o programa e o mantém em *loop* infinito para que sua execução não termine;

- O método *configureGPIOA* serve para ajustar alguns registradores das portas de I/O dependendo do seu parâmetro *default_configuration*. Para se utilizar o *bean* PPG na porta genérica A, por exemplo, os registradores DDIR e PEREN devem ter os valores 0x3F e 0x10, respectivamente, porém, para acesso direto a esta porta os mesmos registradores devem ser ajustados para 0xC0 e 0x3F;

- Os métodos *setESCP* e *clearESCP* são para carregar e limpar, respectivamente, o pacote de dados na variável global *mainPack*;

- O método *processESCP* serve para direcionar o programa ao conversor correto. Ao ser chamado, este método lê o código do conversor em questão e repassa os parâmetros necessários ao método de controle do módulo correspondente ao conversor desejado;

- O método *checkCRC* efetua a validação do byte de paridade do pacote, garantindo assim que os dados transmitidos não possuam erros. Quando um pacote é recebido (o evento *OnFullRxBuf* é disparado sempre que 23 bytes entram no *buffer* da interface serial), todos os bytes são primeiramente passados para o método *checkCRC* que utilizando um algoritmo matemático calcula qual deverá ser o CRC do pacote e o compara com o valor do 19º byte (B_{19}) do pacote. Somente se os valores coincidirem é que o pacote é utilizado, caso contrário o mesmo é descartado.

Algoritmo de verificação[1]:

$s = 0x000000B_{18} + 0x000000B_{17} + B_1B_2B_3B_4 + B_5B_6B_7B_8 + \ldots + B_{13}B_{14}B_{15}B_{16}$

$CRC_{calculado} = (s \text{ AND } 0x000000FF)$

A listagem completa dos 4 arquivos deste módulo (ELEPOT.h, ELEPOT.c, Events.h e Events.c) encontra-se no Apêndice B.

4.3.3 – Módulo *Chopper*

Para efetuar o controle do *chopper*, independentemente de sua configuração externa (*buck*, *boost* ou *buck-boost*), é preciso apenas um sinal periódico capaz de manter aberta (t_{off}) ou fechada (t_{on}) a chave semicondutora durante um tempo proporcional ao ciclo de trabalho (DC) desejado. São as configurações externas do circuito de potência combinadas com o ciclo de trabalho do conversor que definem se a tensão de saída será menor, igual ou maior que a tensão de entrada.

Um algoritmo de controle tipicamente utilizado no *chopper* é a produção de um sinal rampa com frequência igual à frequência de chaveamento desejada e amplitude unitária, que é constantemente comparado com um sinal constante de valor entre 0 e 1 de acordo com o ciclo de trabalho desejado. Se o valor da rampa for superior ao sinal constante, a chave recebe valor lógico ON (chave fechada) e caso contrário, valor lógico OFF (chave aberta). Conforme é possível observar na figura 4.12 o resultado deste algoritmo, será um sinal quadrado, com período fixo e que varia apenas entre os 2 níveis lógicos (ON e OFF), ou seja, um sinal PWM.

([1]) Para que a soma do algoritmo seja realizável, é necessário que os 2 primeiros elementos possuam também 32 bits (4 bytes). Na implementação do algoritmo será utilizada uma conversão (*cast*) para o tipo inteiro longo, acrescentando 24 bits zero à esquerda.

Como o CodeWarrior oferece um *bean* que produz este tipo de sinal, o controle deste conversor será implementado apenas recebendo os 2 parâmetros do *chopper* (ciclo de trabalho e frequência de chaveamento) e configurando o parâmetro *Period* e chamando o método *SetRatio16* do *bean* PPG a cada comando do usuário. Haverá ainda o comando desligar em que o método *Disable* é chamado, desligando o gerador de sinais PWM do microcontrolador e colocando a sua saída em nível lógico OFF (0 V).

Para mais detalhes acerca do funcionamento e características do *chopper* consultar [14].

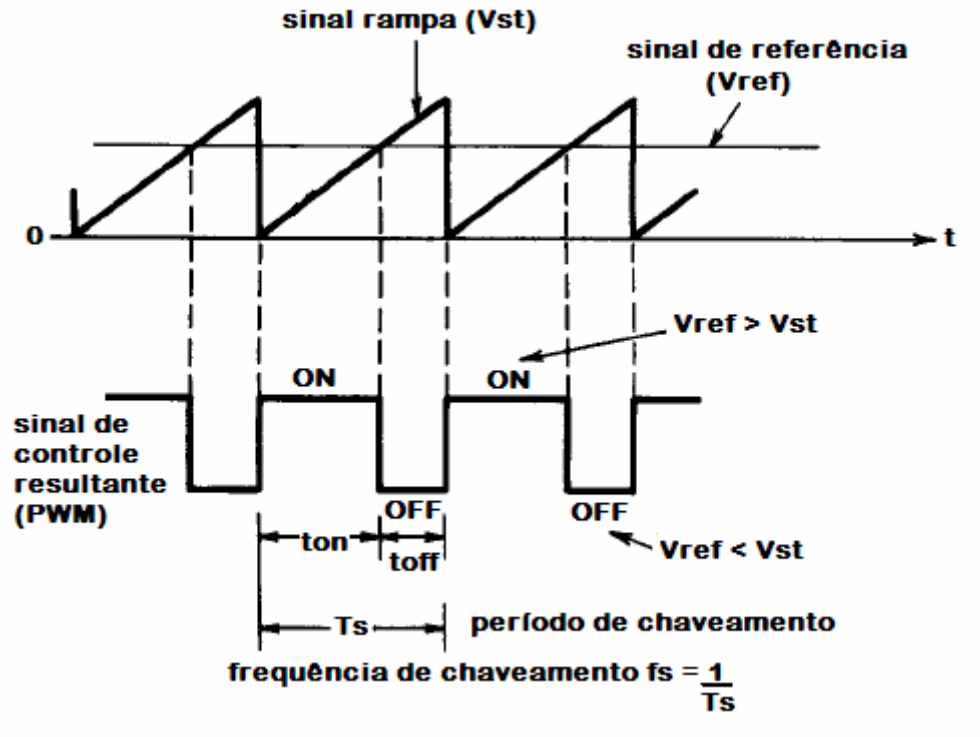


figura 4.12 – Algoritmo de controle do *chopper*

Os únicos métodos e eventos então existentes no módulo *Chopper* são o *setChopper* e o *OnEnd*, respectivamente responsáveis por receber o comando de ligar ou desligar o conversor e ajustar o ciclo de trabalho. A título de sinalização para o usuário, quando o conversor for ligado o LED vermelho apagará e o verde acenderá, além do LED laranja que ficará "piscando" de acordo com o sinal de saída.

O sinal de controle do *chopper* será fornecido no pino 34 (como referência de terra poderá ser usado o pino 3 do *daughter card connector* ou o terminal TP2 / GND da placa de demonstração).

4.3.4 – Módulo *Inverter*

O controle do inversor é similar ao controle do *chopper*, porém, ao invés de se comparar a rampa a um sinal constante o mesmo é comparado à um sinal senoidal de referência. O resultado desta comparação é um sinal similar ao PWM porém com a largura dos pulsos variando constantemente dentro de cada período, conforme ilustrado na figura 4.13. Por essa variação ser muita rápida será inviável a utilização do *bean* PPG como no caso anterior. Um algoritmo iterativo deverá então ser implementado para o controle do inversor.

Como o algoritmo possui também comparações constantes no tempo, um processo temporizado com menor período possível deverá ser usado para que o sinal de saída discreto seja o mais próximo possível de um sinal contínuo. Para isso surge aqui a necessidade do uso do *bean TimerInt*, ajustado com período de interrupção maior que o tempo necessário à execução de uma iteração do algoritmo.

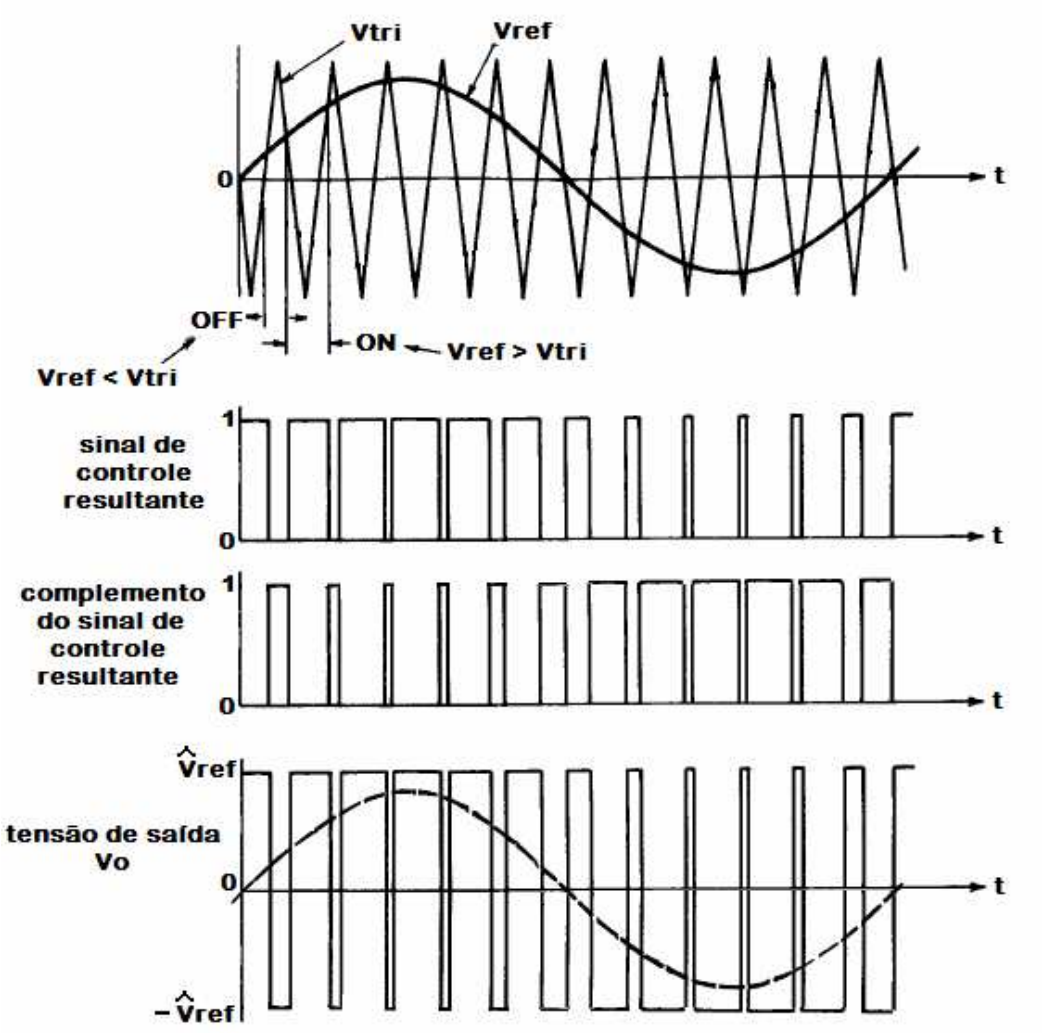


figura 4.13 – Algoritmo de controle do inversor

O algoritmo de controle escolhido utilizará uma rampa completa, cuja função é:

$f(t) = A \cdot \frac{4}{T_{chav}} \cdot t + B$ onde T_{chav} é o período de chaveamento desejado.

E sendo {A, B} = {1, 0} para $t \in \left[0, \frac{T_{chav}}{4}\right]$ ou {A, B} = {1, -4} para $t \in \left[\frac{3T_{chav}}{4}, T_{chav}\right]$

ou {A, B} = {-1, 2} para $t \in \left[\frac{T_{chav}}{4}, \frac{3T_{chav}}{4}\right]$.

O algoritmo irá então, a cada iteração, determinar o valor de t (que evoluirá de acordo com o período de amostragem T_{amos}), calcular a posição da rampa usando a função acima e calcular o valor do sinal senoidal de referência, dado por:

$f(t) = A \cdot \cos(\omega \cdot t)$ onde A é a amplitude, em pu, e $\omega = 2\pi \cdot f$ sendo f a frequência de referência. Tanto A como f são parâmetros do conversor. Como será visto mais adiante, a função que calculará o coseno, desenvolvida no laboratório de microprocessadores da UFRJ, será uma função que trabalha em ponto fixo e seu parâmetro de entrada, o ângulo, em radianos, deverá ser normalizado de [-2π, +2π] para o intervalo [-1, +1]. Por este detalhe será possível eliminar o π do ω evitando assim as operações aritméticas de divisão, que possuem elevado custo de processamento mesmo em ponto fixo.

Neste algoritmo o tempo de processamento é de extrema importância já que quanto maior for a frequência de amostragem mais preciso será o sinal de saída. Será preciso então reduzir ao máximo a quantidade de operações aritméticas a serem realizadas.

Existe uma considerável diferença de tempo entre as operações aritméticas básicas e as operações de deslocamento de bits, comparações, substituição de valores em registradores etc., sendo estas últimas desprezíveis se comparadas com a primeira. E também existe uma grande diferença de tempo entre as operações em ponto fixo e ponto flutuante, conforme ensina em seus trabalhos, os autores de [8] e [13]. Uma medição realizada em laboratório com o microcontrolador MC56F8013 trouxe os seguintes resultados:

	+ , -	*	/
Inteiro	0,2 µs	0,3 µs	1,3 µs
Inteiro longo	0,3 µs	0,4 µs	5,7 µs
Ponto flutuante (real)	7,9 µs	7,3 µs	11,2 µs

tabela 4.2 – Tempo necessário às operações aritméticas do MC56F8013 (resumo)

Para auxiliar na otimização do algoritmo, alguns valores redundantes são calculados dentro do método *setInverter* (que é executado apenas quando um comando é recebido) como por exemplo as frações do período de chaveamento (¼, ½ e ¾ de T_{chav}). Outra característica do algoritmo voltada para sua otimização é concentrar ao máximo todas as operações aritméticas no método *setInverter* e somente as que dependem de t dentro do evento *OnInterrupt* do *bean TimerInt*.

Os resultados apresentados na tabela 4.2 deixam claro que o uso de aritmética de ponto flutuante será inviável neste projeto e por isso todas as variáveis do módulo *Inverter* serão do tipo inteiro ou inteiro longo (com ou sem sinal). Como o sistema trata de grandezas reais, será utilizada a aritmética de ponto fixo em formato Q. Para maiores esclarecimentos sobre o formato Q, consultar [8] e [13].

O tempo necessário à execução do algoritmo será:

a) função que calcula seno e co-seno:

11 somas/subtrações (inteiro)	Tempo gasto = 0,2 μs x 11 = 2,2 μs
2 multiplicações (inteiro longo)	Tempo gasto = 0,4 μs x 2 = 0,8 μs
	Total = 3,0 μs

b) algoritmo de controle periódico:

3 somas/subtrações (inteiro longo)	Tempo gasto = 0,3 μs x 3 = 0,9 μs
6 multiplicações (inteiro longo)	Tempo gasto = 0,4 μs x 6 = 2,4 μs
	Total = 3,3 μs

Tempo total necessário = 6,3 μs

Frequência máxima que poderá ser utilizada = $(6{,}3 \times 10^{-6})^{-1}$ = 158,73 kHz

Será então escolhida como frequência de amostragem o valor de 158 kHz.

O módulo *Inverter* será então composto das rotinas para cálculo do seno e coseno em ponto fixo (arquivos q_num.h, sencos.h e sencos.c), os métodos *cos* e *setInverter* e do evento *OnInterrupt*. No caso do inversor monofásico, objeto do laboratório de eletrônica de potência, são necessários 2 sinais complementares na saída, então sempre que o controlador ajustar uma das saídas do microcontrolador para nível lógico ON a outra será ajustada para nível lógico OFF. Assim como no *chopper*, quando o conversor for ligado o LED vermelho apagará e o verde acenderá e os LEDs laranjas ficarão acendendo ou apagando de acordo com os sinais de saída.

Os 2 sinais de controle do inversor serão fornecidos nos pinos 11 e 34 (como referência de terra poderá ser usado o pino 3 do *daughter card connector* ou o terminal TP2 / GND da placa de demonstração).

4.3.5 – Módulo *Rectifier*

O controle do retificador monofásico, de transformador com *tap* central ou de onda completa, é feito por meio de 2 ou 4 sinais, respectivamente. No caso de 4 sinais, há dois pares iguais, ou seja, permanecem apenas 2 sinais distintos. Os retificadores controlados são montados usando-se tiristores como chave de potência [14] – esses dispositivos semicondutores funcionam com um simples pulso para serem fechados e sua abertura é realizada pela inversão de polaridade do sinal de potência. Sendo assim, o sinal de controle do retificador é composto de pulsos sequenciais e espaçados de α graus, onde α é o principal parâmetro do conversor, responsável pelo valor da tensão de saída. A figura 4.14 ilustra como funciona o algoritmo de controle que será implementado no módulo *Rectifier*.

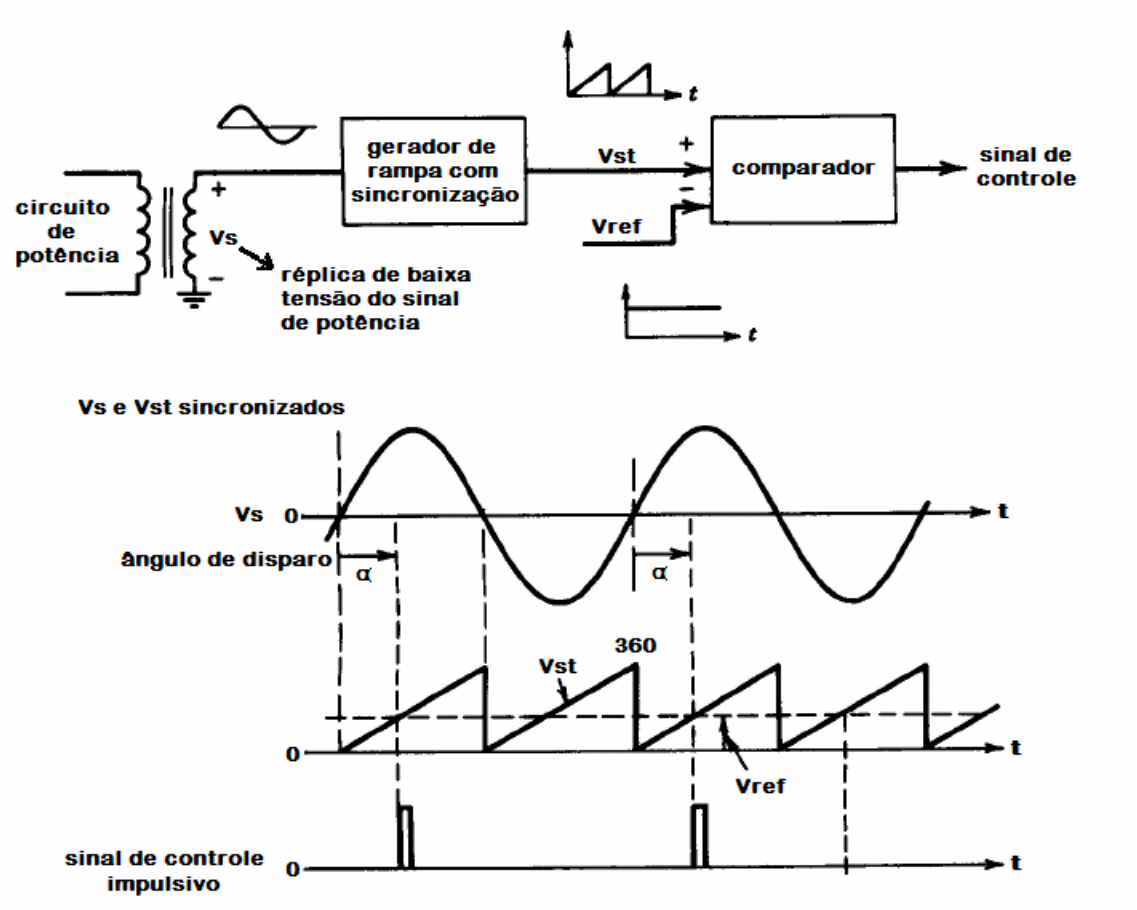


figura 4.14 – Algoritmo de controle do retificador

Na figura, somente um dos sinais está representado. O segundo sinal é disparado sempre 180º após o primeiro. Por esta razão basta apenas um único parâmetro, chamado de ângulo de disparo do retificador. O segundo sinal é obtido a partir do primeiro.

O algoritmo de controle do retificador, aparentemente é muito simples. Basta produzir um sinal do tipo rampa com amplitude 360 e compará-lo constantemente com dois sinais constantes: um deles com o mesmo valor, em graus, do ângulo de disparo desejado α e o outro, com valor igual a (α + 180). A partir do momento em que o valor da rampa for superior ao sinal de referência constante, um trem de pulsos é produzido. Na teoria (tiristores sendo considerados dispositivos ideais) um único pulso seria necessário, porém em aplicações práticas deve-se gerar um trem de pulsos pois é comum que em determinados momentos a chave não feche com um único pulso de duração muito curta.

Assim como no caso do inversor, o algoritmo do retificador também deverá ser implementado com o uso de um *bean TimerInt* para que as iterações possam ser executadas periodicamente. Também como no caso do inversor, quanto maior for a frequência de amostragem do temporizador mais preciso será o sinal de saída. E para isso deverá ser adotada a mesma estratégia de otimização usada no caso anterior: usar somente aritmética de ponto fixo em formato Q, reduzir ao máximo a quantidade de operações aritméticas e concentrar, dentro do possível, o máximo delas fora do evento *OnInterrupt* do *bean TimerInt*, usando inclusive, se necessário, variáveis com valores redundantes.

O tempo necessário à execução do algoritmo será:

a) algoritmo de controle periódico:

3 somas/subtrações (inteiro longo)	Tempo gasto = 0,3 μs x 3 = 0,9 μs
3 multiplicações (inteiro longo)	Tempo gasto = 0,4 μs x 3 = 1,2 μs
2 divisões (inteiro)	Tempo gasto = 1,3 μs x 2 = 2,6 μs
	Total = 4,7 μs

Tempo total necessário = 4,7 μs

Frequência máxima que poderá ser utilizada = $(4{,}7 \times 10^{-6})^{-1}$ = 212,76 kHz

Será então escolhida como frequência de amostragem o valor de 200 kHz.

Um outro detalhe, muito importante, no controle do retificador é que para a determinação precisa do momento de disparo do sinal de controle é necessário que o sinal rampa tenha seu cruzamento pelo zero sincronizado com o sinal de potência, externo ao controlador. Esse é o principal ponto crítico do controle do retificador, conforme mencionado nos capítulos anteriores.

No sistema atual, um circuito de sincronização analógico, desenvolvido por Rolim [1] serve para informar ao controle que o sinal de potência cruzou o zero (figura 4.15). Este circuito possui algumas limitações em relação à rejeição de ruídos. A idéia deste projeto é tornar independente a utilização do circuito sincronizador e por não fazer parte do escopo deste trabalho, uma implementação digital e por software de um circuito de sincronismo (um PLL por exemplo) ficará como sugestão para estudos futuros sobre o tema em uma possível expansão deste sistema. Ficará como indicação o uso do sincronizador analógico nos experimentos 4 e 5 do laboratório.

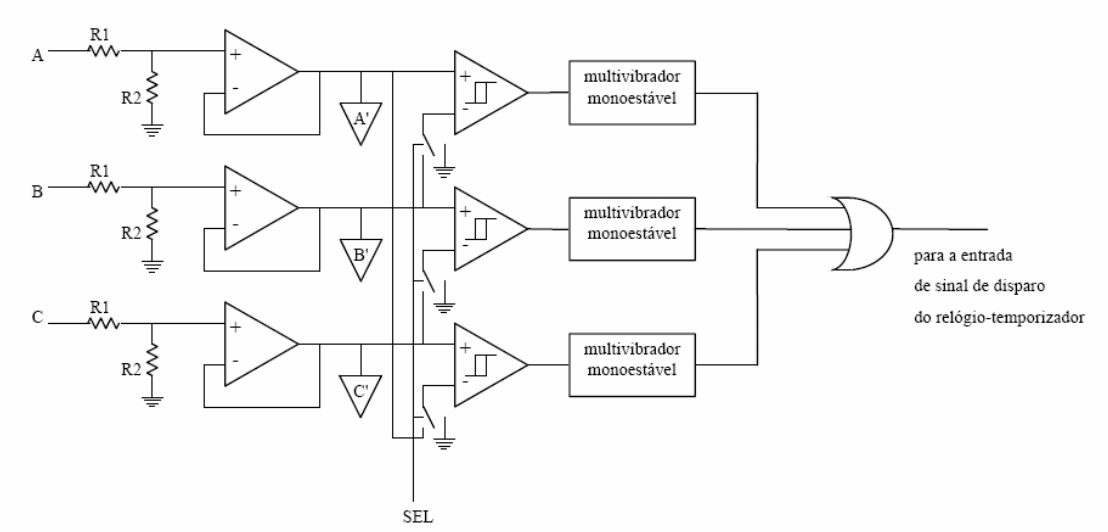


figura 4.15 – Circuito analógico de sincronismo
(fonte: Rolim – Laboratório Modular de Eletrônica de Potência)

Para permitir a utilização do circuito de sincronimo será usado no módulo *Rectifier* o *bean A/D Converter*, que dá acesso ao conversor analógico-digital e permitirá receber sinais de controle de um circuito de sincronismo externo qualquer (um sinal não nulo será suficiente para "resetar" o sinal rampa) ou então ser usado para receber uma réplica em baixa tensão do sinal de potência e processá-lo digitalmente dentro do próprio módulo (neste caso o evento *OnEnd* do *bean* A/D Converter deverá ser apropriadamente alterado). A leitura dos sinais analógicos recebidos pelo conversor será feita de forma contínua e acessada por meio do pino 18.

O módulo *Rectifier* será então composto do método *setRectifier* e dos eventos *OnInterrupt* e *OnEnd*. Assim como nos outros conversores, quando o mesmo for ligado o LED vermelho apagará e o verde acenderá e os LEDs laranjas ficarão acendendo ou apagando de acordo com os sinais de saída.

Os 2 sinais de controle do retificador serão fornecidos nos pinos 11 e 34 (referência de terra no pino 3 do *daughter card connector* ou no terminal TP2 / GND).

5 – Aplicação ELEPOT

O desenvolvimento da aplicação ELEPOT, tópico deste capítulo, engloba as atividades de análise de sistemas e programação orientada a objetos com implementação em linguagem Java. Por este motivo este capítulo foge um pouco do tema eletrônica de potência e alguns aspectos serão então resumidos. Para mais detalhes o recomendado é o estudo de análise e desenvolvimento de sistemas em TI, usando orientação a objetos e UML. De qualquer modo, o texto deste capítulo pode ser entendido mesmo para o leitor que não possua prévios conhecimentos de TI.

Conforme já citado, a aplicação ELEPOT ficará residente no microcomputador PC e se comunicará com o microcontrolador MC56F8013. Neste micro ficará instalado e será executado o servidor de banco de dados MySQL com a base de dados do sistema e o servidor de aplicação Apache Tomcat com a aplicação *web* – sendo esta a responsável por fazer a interface com o usuário e a comunicação com o microcontrolador.

5.1 – Banco de dados

O banco de dados a ser projetado deverá representar as entidades principais do sistema e manter a persistência[1] de informações de configuração, usuários etc. O tipo de banco de dados mais adequado a isso é um banco de dados relacional, onde a estrutura das informações é mantida em um conjunto de tabelas com relacionamentos entre si e com determinadas regras pré-definidas.

Para o sistema da bancada didática temos 2 entidades bem identificáveis:

- Usuário (alunos e professores);
- Experimento.

E como entidade auxiliar temos 1 entidade identificável:

- Configuração (o conjunto de configurações do sistema).

([1]) Persistência no jargão técnico de TI significa manter determinada informação gravada na memória não-volátil do sistema, como por exemplo, gravar a informação em um disco rígido.

Em um modelo de dados relacional cada entidade é representada por uma tabela e dependendo do tipo de associação entre as entidades, é possível que exista mais alguma(s) tabela(s) para representar o relacionamento.

As entidades usuário, experimento e configuração serão então representadas por uma tabela cada[1], sendo que entre usuário e experimento há um relacionamento, representado pela linha na figura a seguir:

figura 5.1 – Modelo de dados preliminar do sistema da bancada didática

O relacionamento que existe entre as entidades usuário e experimento é do tipo muitos-para-muitos, ou seja, um usuário pode realizar nenhum ou vários (infinitos) experimentos assim como um experimento pode ser realizado por nenhum ou vários (infinitos) usuários. Este tipo de relacionamento, cuja cardinalidade, no jargão técnico de TI é chamada de N..N (pronuncia-se "n para n") requer que uma nova tabela seja criada para representar a entidade "experimento sendo realizado por usuário" (ou ainda, "usuário realizando experimento". Este relacionamento receberá o nome de "prática", ficando então o modelo de dados com a seguinte estrutura:

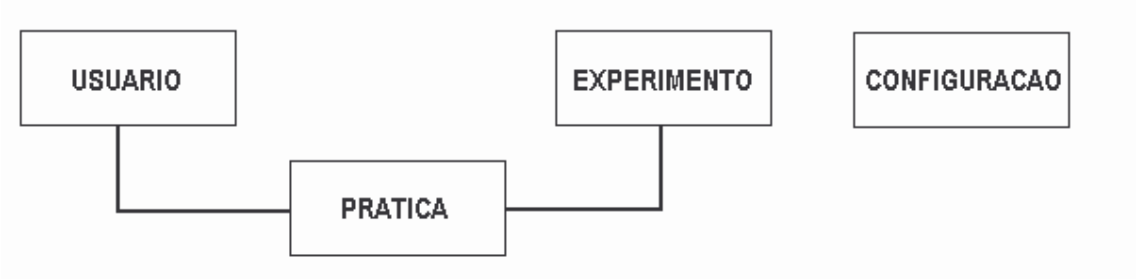


figura 5.2 – Modelo de dados do sistema da bancada didática

Com a estrutura das entidades e relacionamentos definida, a próxima etapa é a confecção do dicionário de dados, ou seja, o conjunto de atributos de cada entidade (ou colunas de cada tabela no banco de dados). As tabelas 5.1, 5.2, 5.3 e 5.4 definem este conjunto de atributos para as entidades usuário, experimento, prática e configuração, respectivamente.

([1]) Em projeto de bancos de dados não se utiliza, por convenção, caracteres acentuados (e cedilha) para nomear os objetos.

5.1.1 – Tabela Usuário

A tabela de usuários, cujo nome no banco de dados será USUARIO, possuirá os seguintes atributos: id, que será o identificador único (chave primária) de cada registro; *login*, que representará o nome que o usuário utiliza para entrar no sistema; nome e senha, que representam o nome e a senha do usuário; identificação, que pode ser a matrícula do aluno ou do professor; e um indicador se o usuário tem sua conta ativa ou não – este atributo pode ser útil por exemplo para retirar o acesso de alunos que já cursaram a disciplina laboratório de eletrônica de potência porém sem perder seu histórico no sistema. Por convenção, é comum em projetos de banco de dados utilizar-se um prefixo para cada tabela. Neste caso então o prefixo será USU. A estrutura completa desta tabela é apresentada na tabela 5.1.

USUARIO			
atributo	**tipo**	**Tamanho**	**Obrigatório**
USU_ID	INTEIRO	–	SIM
USU_LOGIN	CARACTERE	10	SIM
USU_SENHA	INTEIRO	–	SIM
USU_NOME	CARACTERE	50	SIM
USU_IDENTIFICACAO	CARACTERE	20	NÃO
USU_SN_ATIVO	BOOLEANO	–	SIM

tabela 5.1 – Tabela USUARIO

O campo USU_ID é um campo para uso interno no sistema (o usuário nunca irá acessá-lo) e por isso seu valor em si, desde que único, não interessa muito. Por conta disso tal campo será definido como um inteiro auto-incrementável. Um outro detalhe está no tipo do atributo USU_SENHA, que é inteiro. Neste campo não será guardada a senha em si pois não é um procedimento aceitável armazenar senhas em banco de dados sem qualquer tipo de transformação ou criptografia. O método adotado neste projeto chama-se *hash* que é uma transformação matemática, irreversível, que consiste de transformar um texto alfanumérico qualquer (e de qualquer tamanho) em uma cadeia numérica de comprimento fixo. O método de *hash* a ser usado neste projeto é o mesmo utilizado na classe Java.Lang.String (método *hashCode*) do Java, cujo algoritmo é:

$$s_0 \cdot 31^{(n-1)} + s_1 \cdot 31^{(n-2)} + \ldots + s_{(n-1)}$$

onde s_i é cada caractere do texto a ser transformado e n o tamanho deste texto.

5.1.2 – Tabela Experimento

A tabela de experimentos, cujo nome no banco de dados será EXPERIMENTO, possuirá os seguintes atributos: id, que será o identificador único (chave primária) de cada registro; número, que representará o número do experimento no roteiro; nome e objetivo do experimento; descrição e instruções, que armazenarão os detalhes teóricos de cada experimento e seu roteiro (material utilizado, procedimentos, questões que devem ser abordadas no relatório de cada aluno etc.). O prefixo desta tabela será EXP e a sua estrutura completa é apresentada na tabela 5.2. Assim como já explicado no item anterior, o atributo id será um inteiro auto-incrementável.

EXPERIMENTO			
Atributo	**Tipo**	**Tamanho**	**Obrigatório**
EXP_ID	INTEIRO	–	SIM
EXP_NUMERO	INTEIRO	–	SIM
EXP_NOME	CARACTERE	50	SIM
EXP_OBJETIVO	CARACTERE	150	SIM
EXP_DESCRICAO	CARACTERE	1000	NÃO
EXP_INSTRUCOES	CARACTERE	2000	NÃO

tabela 5.2 – Tabela EXPERIMENTO

5.1.3 – Tabela Prática

A tabela de práticas, cujo nome no banco de dados será PRATICA, possuirá os seguintes atributos: id, que será o identificador único (chave primária) de cada registro; exp_id e usu_id, que representarão o experimento e o usuário, respectivamente, de cada prática; um indicador para informar se a prática está sendo realizada ou não, de modo a impedir acessos simultâneos ao mesmo equipamento; e data, que armazenará a data em que a prática foi realizada pelo aluno. O prefixo desta tabela será PRA e sua estrutura completa é apresentada na tabela 5.3.

PRATICA			
Atributo	**Tipo**	**Tamanho**	**Obrigatório**
PRA_ID	INTEIRO	–	SIM
PRA_EXP_ID	INTEIRO	–	SIM
PRA_USU_ID	INTEIRO	–	SIM
PRA_SN_USO	BOOLEANO	–	SIM
PRA_DATA	DATA	–	NÃO

tabela 5.3– Tabela PRATICA

5.1.4 – Tabela Configuração

A tabela de configuração, cujo nome no banco de dados será CONFIGURACAO possuirá os seguintes atributos: sistema operacional, que será usado para armazenar o nome do sistema operacional usado na bancada didática (Linux ou Windows, por exemplo); porta serial, que representará qual o nome (e/ou caminho) da porta serial do PC que será usada na comunicação com o microcontrolador (COM1, COM2, /dev/ttyS0, /dev/ttyS1, por exemplo); e tempo de sessão, que indicará quanto de tempo de sessão será dado ao usuário da bancada didática, de modo a impedir que um usuário ao "abandonar" seu navegador *web* aberto na página de controle da bancada não trave os demais usuários – após um determinado tempo, configurado nesta tabela, a aplicação *web* irá abandonar a sessão do usuário e liberar o controle. Um detalhe importante sobre esta tabela é que ela sempre terá apenas um único registro já que representa informações da bancada didática como um todo, e não apenas informações específicas de determinados experimentos e/ou usuários. O prefixo desta tabela será CFG e a sua estrutura completa é apresentada na tabela 5.4.

CONFIGURACAO			
Atributo	**Tipo**	**Tamanho**	**Obrigatório**
CFG_SIST_OPER	CARACTERE	10	SIM
CFG_PORTA_SERIAL	CARACTERE	50	SIM
CFG_TEMPO_SESSAO	INTEIRO	-	SIM

tabela 5.4 – Tabela CONFIGURACAO

5.1.5 – Outros objetos do banco de dados

Além das tabelas, há ainda mais alguns detalhes acerca do banco de dados:

- Deverá ser criado um índice único sobre o atributo USU_LOGIN da tabela usuário de modo a impedir que usuários diferentes possuam o mesmo *login*;
- Deverá ser também criado um *database* e um usuário de banco de dados. Para o *database* será dado o nome ELEPOT e para o usuário (que será usado na aplicação *web*) o nome elepot_web com senha dfr457g (senha esta que será padrão neste projeto todo, porém nada impede que uma outra seja definida);
- O usuário elepot_web deverá receber todos os privilégios de acesso ao *database* ELEPOT.

5.2 – Java e o servidor de aplicação

Uma aplicação *web* típica não possui a mesma estrutura de uma aplicação *desktop* com uma rotina principal (em Java e linguagem C por exemplo, esta rotina é o método *main*) que é chamada após um arquivo executável ser iniciado. No caso da *web*, existe um software, chamado de servidor de aplicação, que fica em execução o tempo todo e é responsável por receber requisições dos clientes (que utilizam os navegadores *web*), passá-las à correspondente aplicação que está sendo executada dentro dele (esta aplicação deverá ser compatível com o servidor de aplicação), e após o processamento das informações feitas, receber uma "resposta" e repassá-la ao requisitante, ou seja, ao navegador *web* do usuário. Esta compatibilidade entre a aplicação e o servidor de aplicação é necessária para que o servidor saiba como se comunicar com sua aplicação. No caso do Java, para existir esta compatibilidade é necessário que o servidor de aplicação seja um *Servlet Container*, ou seja, suporte as tecnologias *Java Servlets* e *JavaServer Pages* (JSP).

Como já citado anteriormente, o servidor de aplicação escolhido para este projeto é o Apache Tomcat, que além de ser um *Java Servlet Container*, é também um software livre, está largamente difundido no mundo com extensa documentação e suporte e, por ser um dos servidores de aplicação mais utilizados em todo o mundo, possui grande credibilidade.

Durante a etapa de desenvolvimento, o servidor de aplicação embutido na IDE já estará automaticamente configurado para trabalhar com a aplicação. Isso significa que ao apertar o botão "*run*" a aplicação irá abrir dentro de um navegador *web*, porém quando a mesma estiver pronta para o ambiente de execução da bancada didática, deverá ser feita uma "instalação" da aplicação dentro do servidor de aplicação. Esta operação é chamada de *deploy* da aplicação. Ainda durante a etapa de desenvolvimento todas as bibliotecas (APIs, *drivers* etc.) necessárias à aplicação deverão ser incluídas. No caso deste projeto são três: o *framework* Struts, o *driver* JDBC de conexão com o banco de dados e a API de comunicação RXTX. No capítulo 6, há mais detalhes sobre a instalação destes componentes, porém para informações mais detalhadas, consultar a documentação específica ([18], [24], [25], [26]).

5.3 – Aplicação *web*

A aplicação *web* se divide da seguinte forma: páginas *web* estáticas (arquivos .html) e dinâmicas (arquivos .jsp); pacotes com classes Java (pacotes da aplicação e do *framework* Struts) e arquivos de configuração (arquivos .properties, struts-config.xml e web.xml etc.).

Em toda esta aplicação, que utiliza o *framework* Struts, a arquitetura MVC, já comentada na seção 3.1.4, é implementada. Na camada de apresentação estão as páginas *web* (estáticas e dinâmicas); na camada de modelo, as classes Java que representam o modelo de dados e regras (pacotes br.ufrj.dee.elepot.model e br.ufrj.dee.elepot.model.*, br.ufrj.dee.elepot.dao, br.ufrj.dee.elepot.bo e br.ufrj.dee.elepot.struts.form) e na camada de controle, as classes Java do Struts que determinam o fluxo de dados (pacote br.ufrj.dee.elepot.struts.action) e o arquivo struts-config.xml, que centraliza e organiza todo o controle da aplicação.

5.3.1 – Interface com o usuário: páginas *web* da aplicação

As páginas *web* estáticas servem para definir os elementos fixos na interface com o usuário (cabeçalho e rodapé da aplicação por exemplo) e normalmente ficam dentro do diretório "Web Pages".

As páginas *web* dinâmicas por sua vez misturam código de visualização (em HTML) com código JSP. No momento em que essas páginas são requisitadas todas as *tags* de JSP[1] são processadas (ou "renderizadas") e seu conteúdo substituído pelo resultado do processamento, sendo entregue ao usuário sempre um conteúdo de HTML puro. A vantagem de utilizar as *tags* JSP ao invés de código Java dentro das páginas é impedir a mistura de código de visualização com código da lógica de negócios. A tecnologia JSP permite a implementação da arquitetura MVC. Porém, essa característica nem sempre pode ser seguida à risca e por isso a JSTL (e também as *tags* do Struts) incluem *tags* com comandos condicionais e de repetição, além de diversos outros comandos relacionados à lógica de programação.

([1]) A tecnologia JSP permite a utilização de *tags* especiais cujo conteúdo é processado dinamicamente de acordo com cada requisição que é feita. Esse conjunto de *tags* é chamado de JSTL porém outras *tags* podem ser definidas pelo desenvolvedor. Alguns *frameworks* como o Struts por exemplo possuem seu próprio conjunto de *tags* JSP.

Como não é interessante que o usuário da aplicação tenha como acessar as páginas dinâmicas (pois elas somente farão sentido após terem sido processadas), elas normalmente ficam localizadas dentro do diretório "Web Pages / WEB-INF", que é um diretório especial onde o conteúdo não pode ser acessado diretamente pelo servidor *web*, mas somente pelo servidor de aplicação.

A listagem completa de todos os arquivos .html e .jsp que compõem o conjunto de páginas *web* da aplicação encontram-se no Apêndice C.

Existem outros dois diretórios importantes na aplicação: "img" e "javascript", ambos dentro de "Web Pages". O primeiro contém todas as imagens gráficas usadas na aplicação (arquivos .jpg, .gif, .png etc.), inclusive as figuras utilizadas nos roteiros das experiências, e o segundo contém arquivos .js com funções em Javascript, utilizadas nas validações das telas. Em uma aplicação *web* é comum que certas validações sejam feitas antes dos dados serem enviados ao servidor, evitando processamento desnecessário, por exemplo, caso um campo obrigatório da tela tenha ficado acidentalmente em branco. As funções em Javascript permitem testar e validar diversas condições de preenchimento dos formulários da aplicação, interagindo com o usuário e impedindo que requisições incompletas sejam feitas.

A estrutura de organização das páginas *web* é bem simples e pode ser facilmente expandida em projetos futuros. Cada cadastro é composto por basicamente três arquivos jsp: «cadastro».jsp, «cadastro»_form.jsp e «cadastro»_ok.jsp, onde «cadastro» é o nome do cadastro em questão (por exemplo, alunos). O primeiro lista os registros existentes e permite a seleção de um deles, o segundo exibe os campos para inclusão ou alteração do registro selecionado e o terceiro exibe uma mensagem com o resultado de alguma ação do usuário. Há ainda os arquivos referentes às páginas principais do sistema (index.jsp, login_erro.jsp, laboratorio.jsp e manutencao.jsp) e dos 5 experimentos (exp_n.jsp onde n é o número do experimento) e outras páginas de apoio, como por exemplo: senha.jsp, que é a tela de troca de senha; popup_*.jsp que exibe informações em uma outra janela do navegador *web*; valida_sessao.jsp, que trata das sessões HTTP, assunto que será visto na seção 5.3.3; bloqueia_cache.jsp, que impede que as páginas da aplicação fiquem guardadas no *cache* de internet (como as páginas são quase todas dinâmicas, não é interessante que informações desatualizadas armazenadas no *cache* apareçam para o usuário); entre outras páginas.

5.3.2 – *Framework* Struts

O *framework* Struts está se tornando um padrão em projetos de desenvolvimento de aplicações Java. Sua principal característica é servir de base para a implementação da arquitetura MVC, porém outras notáveis vantagens podem ser obtidas com ele, como por exemplo a abstração de código referente ao tratamento dos pacotes HTTP e uso simplificado dos *servlets* Java. Além disso, o Struts permite que o código fique muito organizado e bem estruturado. Usando técnicas como a reflexão[1] o Struts permite ao desenvolvedor criar um código muito mais legível e robusto em suas páginas *web*.

O Struts controla ainda todo o fluxo de dados da aplicação, decidindo o que chamar e em quais momentos e permite, com bastante organização, tornar a aplicação internacionalizável, conceito importante do Java que permite que uma mesma aplicação, mantendo o mesmo código-fonte, possua versões em qualquer língua suportada pelo sistema Unicode[2].

O Struts divide-se basicamente da seguinte forma:

1) O arquivo struts-config.xml é centro do controle da aplicação e lá são feitos os mapeamentos de qual *action* será executada de acordo com a URL solicitada, além de outros detalhes, como configurações referentes à internacionalização e *form-beans*, e uso do *validator*, que no caso deste projeto não será utilizado pois a validação necessária em diversas telas é muito particular e será feita somente com Javascript;

2) As *actions* são classes herdadas pela aplicação da classe Action do Struts que permitem acesso ao conteúdo do pacote HTTP e aos campos dos formulários HTML, além de também permitirem o redirecionamento da aplicação para outras páginas *web*;

3) Os *form-beans* são classes Java que representam todos os formulários usados na aplicação. Todos os *form-beans* herdam da classe ActionForm do Struts e é por meio deles que os dados são lidos dentro das *actions*.

([1]) A reflexão em Java é a característica que permite que uma aplicação instancie e utilize objetos Java que somente existem em tempo de execução. Com a reflexão é possível por exemplo que uma classe acesse uma segunda classe que em tempo de compilação que ainda não foi instanciada. No Struts a reflexão é usada para determinar e executar métodos *get* e *set* dos objetos (*beans*) sem que o programador precise escrevê-los por completo (somente é necessário o nome do *bean*).

([2]) Unicode é um padrão que permite aos computadores representar e manipular, de forma consistente, texto de qualquer sistema de escrita existente (fonte: Wikipédia).

O diagrama da figura a seguir ilustra resumidamente como é essa divisão do Struts e permite ter uma idéia, também resumida, do seu funcionamento. Para mais detalhes sobre o *framework* Struts consultar [16] e [26].

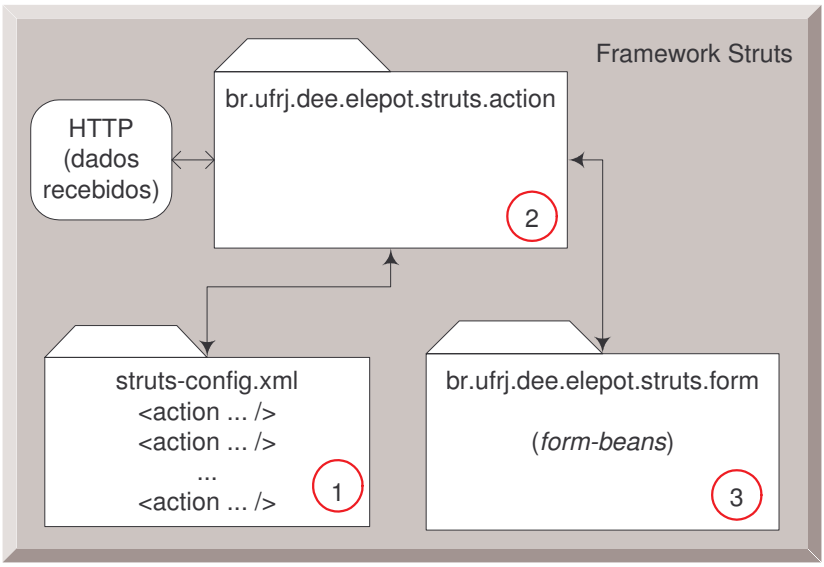


figura 5.3 – Funcionamento resumido do *framework* Struts

O funcionamento do Struts, basicamente se resume ao seguinte "roteiro": sua função é receber uma requisição do usuário (as URLs no Struts, que por convenção, sempre possuem o sufixo ".do"), "entrar" no arquivo struts-config.xml e localizar a requisição em questão por meio do "*path*" da URL, executar o código da *action* associada em "*type*" e, feito o processamento (dentro da *action* correspondente), redirecionar a aplicação para a URL descrita em "*path*" de acordo com o valor passado no método *findFoward* da *action*. Existem ainda alguns outros casos onde o redirecionamento é feito diretamente, sem a utilização de *actions* (caso por exemplo de *hyperlinks* dentro da aplicação) e outros em que são usados coringas ("*" e "{1}" por exemplo) que funcionam como parâmetros. Para utilizar o Struts, o desenvolvedor deve definir no arquivo struts-config.xml uma série de mapeamentos para as *actions* que deverão estar definidas como classes dentro do pacote de *actions* do Struts. Todas essas *actions* irão ser descendentes da classe Action e, dentre os objetos de Action, existirá o *form*, da classe ActionForm do Struts e que representa um formulário HTML qualquer, que, usado em conjunto com os *form-beans*, permite acessar todos os campos das páginas JSP. Uma outra característica nos mapeamentos do Struts é o escopo. O escopo de uma *action*, definida no arquivo struts-config.xml pode ser entendido como o espaço de objetos do sistema que a *action* em questão irá "enxergar". O escopo pode ser definido como *request*, *session* ou *application*. Na maior parte dos casos (inclusive

nesse projeto) o escopo será sempre o *request*, ou seja, cada action irá "enxergar" apenas os objetos daquela requisição. Se um outro escopo fosse definido, outros objetos, mais abrangentes e relacionados à sessão ou à aplicação como um todo poderiam ser acessados, como, por exemplo, um possível contador de usuários conectados.

O exemplo a seguir mostra um mapemento usado na aplicação elepot, definido no arquivo struts-config.xml e na classe br.ufrj.dee.elepot.struts.action:

```
<action path="/logout" type="br.ufrj.dee.elepot.struts.action.Logout" scope="request">
  <forward name="ok" path="/index.jsp"/>
</action>
```

```
package br.ufrj.dee.elepot.struts.action;

import javax.servlet.http.*;
import org.apache.struts.action.*;

public final class Logout extends Action {

    public ActionForward execute (ActionMapping map, ActionForm form, HttpServletRequest
request, HttpServletResponse response) throws Exception {
        HttpSession sessao = request.getSession(false);
        sessao.invalidate();
        return map.findForward("ok");
    }
}
```

Neste exemplo é definido um mapeamento para a *action* Logout (uma classe cujo código define os procedimentos de saída do usuário no sistema) que é executada sempre que a URL http://url-da-aplicação/logout.do é chamada. No final do código da *action* é feito um redirecionamento para o *alias* "ok", que na definição do mapeamento, também faz um redirecionamento para a página index.jsp. Ou seja, quando o usuário deseja sair do sistema, alguns procedimentos de finalização são executados e no final é devolvido a ele a página inicial da aplicação.

Um outro aspecto importante do Struts é seu suporte a internacionalização. Existe um arquivo, normalmente chamado de ApplicationResources.properties, que contém todas as mensagens que serão usadas na interface com o usuário. A idéia é que nas páginas JSP não exista texto mas apenas *tags* que apontam para um determinado registro dentro do arquivo de internacionalização. O navegador *web*, ao transmitir uma requisição informa também qual o local (*locale*) do usuário para o servidor de aplicação, que por sua vez, utiliza esta informação para determinar qual a linguagem será usada dentre as disponíveis no arquivo de internacionalização. Esse arquivo normalmente fica dentro do pacote struts e é ligado à aplicação por meio do struts-config.xml.

5.3.3 – *JavaBeans* e persistência

Além das classes relacionadas ao Struts a aplicação possui diversas outras em seus vários pacotes, cada uma delas com funções específicas relacionadas ao negócio (os conversores de eletrônica de potência) ou funções que auxiliam a aplicação.

O conceito de *bean*[1] no Java, também conhecido como JavaBean em quase nada se assemelha ao *beans* apresentados no capítulo 4. Os JavaBeans servem para representar, no ambiente orientado a objetos, as entidades do sistema. Para cada uma destas entidades, existe um *bean* correspondente, com todos os seus atributos privados e com acesso único por meio de métodos *get* e *set*, e sempre sem construtor (ou então, conforme especificação da Sun [18], com um construtor definido porém sem código). Tal estrutura auxilia a manter três características extremamente desejáveis em sistemas orientados a objeto: encapsulamento, baixo (ou nenhum) acoplamento e alta coesão. Os *beans* representam então unicamente os objetos e como os mesmos armazenam e fornecem acesso às suas propriedades. O comportamento (regras de negócio) desses objetos ficará em outras classes, chamadas de *bussiness object classes* (classes de negócio do objeto), cuja nomenclatura costuma acrescentar o sufixo "BO" ao nome das classes. Alguns destes objetos precisam ser persistidos[2] no banco de dados e para tal existe mais um conjunto específico de classes, cujo sufixo é "DAO" por usarem um padrão com mesma sigla. Dentro dessas classes existem métodos para inserir, atualizar, excluir e pesquisar registros no banco de dados e carregá-los nos *beans*, entre outros métodos mais específicos de cada classe. Essa estrutura facilita em muito o desenvolvimento, mantendo o código organizado e enxuto, além de atender aos já consagrados conceitos benéficos da orientação a objetos. A aplicação ELEPOT possui os seguintes pacotes:

- Pacote br.ufrj.dee.elepot.model: contém as classes que representam as entidades do sistema, como "Usuario", "Pratica" e "Experimento".[3] Dentro deste pacote há o sub-pacote br.ufrj.dee.elepot.model.report que contém classes que representam os relatórios (que também são entidades) do sistema.

([1]) O *bean* apresentado no capítulo 4 é um componente de software, com propriedades (que são diretamente acessadas), métodos e eventos, enquanto que o *bean* do Java Bean é uma classe com determinadas características que representa uma entidade qualquer em um sistema.

([2]) Persistência, no jargão técnico de TI é ato de armazenar alguma informação na memória não-volátil do sistema, ou ainda, em outras palavras, gravar informações em disco.

([3]) Não se utiliza caracteres acentuados ou específicos como o c-cedilha em nomes de classes, propriedades ou métodos.

- Pacote br.ufrj.dee.elepot.dao: contém as classes que representam a persistência dos objetos. As classes deste pacote possuem uma série de métodos para ler e pesquisar, gravar, alterar e excluir, entre outras operações mais específicas, as informações contidas nos objetos dentro do banco de dados. Para cada entidade do sistema existe uma classe DAO correspondente (cujo nome é, por exemplo, "ExperimentoDAO") que possui métodos com comandos de SQL, a linguagem usada para manipular o banco de dados. O uso do DAO objetiva desacoplar a camada de negócio da camada de persistência – uma grande vantagem desse desacoplamento é permitir que, por exemplo, o SGBD seja trocado sem que a lógica de programação e todo o restante do código-fonte necessitem sofrer mudanças.

- Pacote br.ufrj.dee.elepot.bo: como visto anteriormente, as classes do pacote *model* representam as entidades e as classes do pacote *dao* a persistência destas entidades. Para representar o comportamento dos objetos do sistema (entidades) e seus relacionamentos com os demais objetos, utiliza-se um conjunto de classes denominadas *BO* (*bussiness object*) com diversos métodos que representam a lógica de negócio do sistema.

- Pacote br.ufrj.dee.elepot.jdbc: possui apenas uma classe, responsável por fornecer à aplicação uma conexão com o SGBD sempre que qualquer informação precisar ser recuperada ou persistida no banco de dados. É nesta classe que ficam os parâmetros de conexão (usuário, senha etc.).

5.3.4 – Classes utilitárias e comunicação serial

As classes utilitárias são normalmente usadas dentro do código dos métodos das classes principais do sistema (relacionadas aos *JavaBeans*) e seus métodos servem para realizar conversões de tipo, unidades, manipulação de números e texto, contagens, entre diversos outros usos comuns a todo o sistema. Os métodos das classes utilitárias são quase sempre estáticos, ou seja, para utilizá-los não é necessário instanciar a classe, basta chamar o método com seus argumentos apropriados.

As classes utilitárias do sistema estão dentro do pacote br.ufrj.dee.elepot.util e seus subpacotes. As classes mais importantes são "Conversao" e "DispositivoSerial".

Dentro da classe "Conversao" há um método chamado *IntToByte* que transforma um número inteiro Java (32 bits, ou seja 4 bytes) em um vetor com 4 bytes separados.

Essa sequência de bytes é que será transmitida ao microcontrolador por meio da interface serial RS-232C. A sua sequência de transmissão é de fundamental importância para que o software embarcado seja capaz de transformar de volta os bytes em um único número inteiro e igual ao número original. Essa sequência de formação do número, chamada de *endianness*, e apresentada no capítulo 3, deverá então ser a mesma usada no microcontrolador, que é a *big-endian*, ou seja, o número é formado do byte mais significativo para o menos significativo. O vetor deverá então começar com o byte mais significativo. Outros dois métodos nesta classe são *ByteToBit* e *BitToString*, que servem apenas para depuração, convertendo os bytes em "0s e "1s".

A classe "DispositivoSerial" existe para facilitar o uso do RXTX. Ela provê métodos para configurar, iniciar e usar a porta serial do microcomputador. Algumas das configurações da porta serial como, por exemplo, a sua velocidade (*baud rate*) já são ajustadas dentro da própria classe.

Outras duas classes utilitárias que justificam um resumido comentário são "Zebrado", que serve para a aplicação exibir listas de registros com 2 cores de fundo diferentes e intercaladas, e "TestaDispositivoSerial", que é única classe que apesar de empacotada com a aplicação não faz parte dela. Esta classe serve apenas para realizar testes de transmissão de dados pela interface serial e é executada como um programa independente (*stand alone*) por meio de seu único método *main*.

O subpacote br.ufrj.dee.elepot.util.listener serve para implementar um *listener* Java, que é uma classe especial com comportamento de evento. Um *listener* fica em "alerta" e sempre que um determinado evento pré-definido ocorre seu(s) método(s) correspondentes são executados. Esses métodos também são pré-definidos e geralmente sobrescritos para que o desenvolvedor possa codificar o comportamento desejado. Neste caso da aplicação, o único *listener* usado é o *HttpSessionListener*, referente às sessões *web*, a ser explicado adiante.

As demais classes utilitárias e os demais métodos não citados servem apenas para o módulo administrativo do sistema e outros detalhes típicos de aplicações *web* que não envolvem o tema principal deste trabalho, sendo por isso omitidos.

5.3.5 – Acesso, segurança e sessões

Uma aplicação distribuída pode ser acessada por diversos usuários em diversos micros ou mesmo em diversas janelas no mesmo micro pelo mesmo usuário. Na maior parte dos sistemas computacionais (sistemas comerciais por exemplo) essa característica não limita o seu funcionamento porém no caso da aplicação ELEPOT, a mesma controla um hardware mono-tarefa e é necessário impedir que usuários diferentes (ou o mesmo usuário com diversas instâncias da aplicação) acessem simultaneamente as páginas que comandam o módulo de controle da bancada didática, o que causaria comportamento desconexo entre o conteúdo apresentado na tela da aplicação e o que de fato estaria acontecendo com o hardware da bancada didática.

A aplicação ELEPOT possui o módulo administrativo (que deve ser acessado somente por professores) e a parte do comando do módulo de controle microprocessado da bancada didática. Para auxiliar os professores no acompanhamento das atividades realizadas no equipamento pelos alunos, o módulo administrativo possui um relatório com as práticas realizadas e os alunos devem ser identificados ao acessarem o sistema. Essa identificação será o *login* do aluno e cada um possuirá uma senha. Ou seja, a aplicação deverá possuir áreas de acesso restrito e somente usuários identificados por meio de seus *logins* e senhas poderão acessá-las.

Por conta destas características apresentadas a aplicação ELEPOT precisará possuir um sistema de segurança que impedirá acessos não autorizados, não identificados e simultâneos (a certas partes da aplicação somente). Em aplicações *web* essas restrições podem ser implementadas de 3 modos:

- *Cookies*;
- Sessões com *cookies*;
- Sessões com reescrita de URL (*URL rewriting*).

Os *cookies* são arquivos temporários que o navegador *web* armazena em seu *cache* de internet e que mantém no micro cliente algumas informações que não podem ser perdidas quando se passa de uma página para outra (ou mesmo de uma instância qualquer do navegador *web* para outra) pois o protocolo HTTP usado nas aplicações *web* não mantém quaisquer informações de estado. As sessões são um mecanismo disponível nos *servlets* Java que permitem gravar e recuperar as informações de estado entre as páginas de uma aplicação *web* de forma muito prática, sem que o programador necessite implementar todo um código para essa finalidade. As sessões são instanciadas

e podem a partir daí armazenar e recuperar objetos Java, sendo ideais para aplicações com controle de acesso de usuários, entre outros. O gerenciador de sessões dos *servlets* Java somente conseguem distinguir as sessões corretas para cada usuário e instância da aplicação por meio do identificador único de sessão (*jsessionid*). Este identificador precisa ser mantido por meio de armazenamento local (sessões com *cookies*) ou então ser repassado dentro de cada nova URL gerada pela aplicação. O primeiro caso é mais simples pois o próprio gerenciador de sessões cuida de tudo, porém a utilização de *cookies* onde sua função está ligada ao controle de acesso do sistema deve ser evitada já que todo navegador *web* possui opção para que o usuário desative os *cookies*. Além disso alguns navegadores *web* não trabalham com *cookies*, principalmente no ambiente Linux, que possui dezenas deles.

A aplicação ELEPOT utilizará então um complexo mecanismo de controle, restrição e simultaneidade de acesso que envolverá o uso do banco de dados e de sessões com implementação por meio de reescrita de URL. A reescrita de URL consiste em acrescentar, para toda URL gerada pela aplicação, um parâmetro com o identificador da sessão. Para facilitar o trabalho, o JSP possui em seu objeto *response* o método *encodeURL* que recebe uma URL qualquer e devolve a mesma acrescida do *jsessionid* correto. Para as URLs que são formadas dentro do código das classes Java, o método que retorna o identificador da sessão é o *getId* da classe *HttpSession*. Quando o usuário efetua seu *login* com sucesso, o objeto que representa este usuário é armazenado em uma sessão e no acesso a cada página é feita uma verificação para localizar este objeto, ou seja, caso um usuário tente acessar diretamente uma URL da aplicação sem antes passar pela tela de *login*, o mesmo não conseguirá prosseguir. Para impedir que usuários diferentes acessem simultaneamente as páginas de controle da bancada didática, a aplicação utiliza o banco de dados (atributo "PRA_SN_USO" da tabela "PRATICA") para manter o controle de qual usuário está utilizando o equipamento e somente permite o acesso dos demais quando o mesmo estiver disponível. O ponto complicador deste mecanismo é quando o mesmo usuário acessa 2 ou mais instâncias da aplicação. No banco de dados já constará que é ele quem está trabalhando na bancada e seu n-ésimo acesso não teria como ser impedido. Para contornar esse comportamento, a aplicação terá de impedir que o mesmo usuário faça *logins* simultâneos e usará as informações da sessão para tal. Como o objeto que representa o usuário somente sairá da sessão quando a mesma for encerrada, por meio do *hyperlink* "voltar" da aplicação ou quando o tempo de expiração da sessão (*timeout*) for atingido, será extremamente importante que os

usuários da bancada didática não fechem seus navegadores *web* sem antes sair completamente da aplicação utilizando os *hyperlinks* "voltar" até a tela incial do sistema, sob pena de ficar com a sua respectiva sessão "travada" até que a mesma expire por tempo ou o servidor de aplicação seja reiniciado.

Para impedir que páginas da aplicação já visitadas fiquem guardadas no *cache* de internet e sejam novamente exibidas, o que pode trazer resultados desatualizados e incoerentes, há um arquivo chamado "bloqueia_cache.jsp" que é responsável por limpar as informações do *cache* referentes à aplicação ELEPOT, independentemente de como o usuário tenha configurado seu navegador *web*.

5.3.6 – Arquivo de *log*

É comum que sistemas computacionais possuam um registro de eventos e alertas importantes a respeito de seu funcionamento. Este mecanismo é comumente conhecido como *log*. O arquivo de *log* deve conter sempre informações importantes que auxiliem o administrador do sistema (ou mesmo o desenvolvedor) a localizar, mapear e corrigir comportamentos fora do esperado.

Aplicações *web* normalmente utilizam o próprio mecanismo de *log* do servidor de aplicação para registrar seus eventos. No caso do J2EE e do Tomcat, tudo o que é enviado para o console[1] ficará automaticamente gravado no arquivo "catalina.out", que localiza-se no diretório "logs", dentro do diretório do Tomcat. Esse arquivo deverá ser sempre lido pelo administrador do sistema para pesquisar possíveis anormalidades ou problemas na aplicação (por exemplo, falha na comunicação de dados).

Fora os eventos do próprio servidor de aplicação, a aplicação ELEPOT irá armazenar no arquivo de *log* dois tipos de eventos:

- Criação e destruição de sessões com seu respectivo *jsessionid*;
- Transmissão de bytes pela porta serial com registro no *log* de toda a sequência, bit a bit, a ser transmitida e também o *status* da transmissão (sucesso ou erro de transmissão).

Sempre então que surgirem dúvidas quanto às sessões ou aos dados transmitidos este arquivo deverá ser consultado (o comando será apresentado no item 6.5).

([1]) O console é uma saída de texto comum em diversas linguagens de programação. No Java, o método *System.out.println* envia um objeto *String* qualquer, passado como parâmetro no método, para o console. Ao contrário de uma aplicação *desktop*, onde o console geralmente é a tela do monitor, em uma aplicação *web* o conteúdo do console é enviado para arquivos de *log* (caso do Tomcat).

5.3.7 – Conversores de eletrônica de potência

Conforme apresentado no capítulo anterior, a bancada didática de eletrônica de potência irá trabalhar com 3 tipos de conversor: CC-CC (*chopper*), CC-CA (inversor) e CA-CC (retificador). Assim como no software embarcado, na aplicação *web* cada um destes conversores deverá ter seu comportamento representado em módulos, que neste caso serão classes Java.

O pacote br.ufrj.dee.elepot, último desta análise, conterá as classes referentes aos principais objetos do sistema. Algumas delas representando os três conversores de forma genérica (uma classe específica para cada tipo de conversor, e todas elas agrupadas em uma única classe comum a todos eles, conforme figura 5.4) e outras como classes utilitárias dentro do pacote representando o protocolo de comunicação ESCP e o microcontrolador MC56F8013 na aplicação *web*.

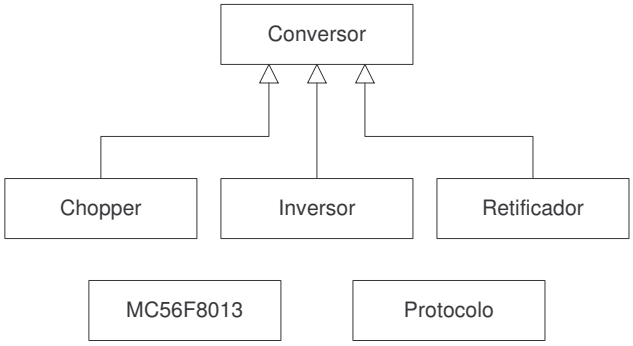


figura 5.4 – Diagrama de classes simplificado do pacote br.ufrj.dee.elepot

A classe abstrata Conversor, herdada por todos os conversores possui um único método, *set*, que efetua a transmissão dos dados, empacotando os comandos e parâmetros do conversor por meio da classe Protocolo – responsável pela transformação dos valores em uma cadeia (*array*) de bytes, e utilizando a interface serial por meio da classe DispositivoSerial, configurada com os parâmetros previamente armazenados no banco de dados. Todo conversor irá então possuir um método *set* (no caso do *chopper* por exemplo, *setChopper*) que irá organizar os comandos e parâmetros do conversor em questão e transmití-los à sua classe pai, com o método *set*.

Dependendo do conversor a ser utilizado os parâmetros a serem transmitidos podem variar em quantidade e conteúdo, assim como também as grandezas envolvidas. Por este motivo, cada conversor terá uma classe específica para representar o seu comportamento. Ainda neste escopo, essas classes devem ser o mais genérico possível

de modo a permitir que em implementações futuras do sistema o microcontrolador possa ser substituído por outro hardware ou ter seu software modificado, ou seja, elas não devem possuir características particulares do MC56F8013, como por exemplo faixas específicas de valores. As classes que representam os conversores deverão ainda possuir métodos que calculem diversas grandezas do conversor (por exemplo, período, frequência e amplitude dos sinais) para serem informadas na interface com o usuários. Tais grandezas poderiam ser medidas diretamente no hardware do conversor, digitalizadas e transmitidas à aplicação, porém este procedimento não foi adotado pois tornaria este projeto demasiadamente complexo fugindo ao escopo deste trabalho, ficando então apenas como idéia para implementações futuras. Estes valores que são calculados nas classes *Chopper*, Inversor e Retificador servem para auxiliar o usuário a acompanhar o funcionamento do conversor e como seu objetivo é ser exibido na tela, seus métodos apresentam saída sempre em texto (*String*) e formatada.

Outra importante função das classes dos conversores é a limitação dos valores de entrada. Os métodos *set* dos parâmetros do conversor sempre limitam os valores recebidos aos valores máximo e mínimo definidos como constantes em cada classe, impedindo assim que o usuário informe valores incoerentes ou irrealizáveis.

Por fim, para tratar das particularidades do MC56F8013 existirá uma classe de mesmo nome e com métodos que farão a adaptação, quando necessária, nos valores dos parâmetros do conversor em grandezas compatíveis com o microcontrolador. Um exemplo típico é o ciclo de trabalho (*duty cycle*) do *chopper*, que é uma grandeza real de valor entre zero e um, porém no MC56F8013 deve ser representado por um inteiro entre 0 e 65.535.

5.3.8 – Código-fonte

O conjunto de pacotes de classes e demais componentes apresentados neste capítulo compõem a aplicação *web*. Por se tratar de um assunto mais específico e pertencente à area de engenharia de software (e não à engenharia elétrica), e visando não tornar este trabalho muito heterogêneo, o desenvolvimento da aplicação *web* ficou resumido a uma visão geral e conceitual apenas, sem detalhes acerca de todos os métodos, atributos e classes envolvidos no processo.

O código-fonte de todos os arquivos desenvolvidos encontra-se no Apêndice C, que ficará servindo de referência completa sobre a aplicação *web*.

6 – Implementação

Para implementar o sistema projetado para a bancada didática será necessário integrar, instalar e configurar todo o material utilizado e desenvolvido nos capítulos anteriores. Este capítulo explica como todo esse sistema foi montado, servindo não apenas de documentação para este trabalho mas também permitindo que no futuro o mesmo seja alterado se necessário.

Essa montagem pode ser divida em:

a) Montagem do hardware da bancada didática – o equipamento a ser usado na bancada didática deverá funcionar em conjunto com o sistema desenvolvido. Poderá ser utilizado tanto o hardware atual, que consiste dos módulos de fontes, cargas, chaves e seus circuitos auxiliares (sincronizador analógico por exemplo), como o novo hardware que está sendo desenvolvido atualmente pelo Departamento de Engenharia Elétrica da UFRJ. Somente os módulos de interfaces e controle é que serão completamente substituídos (seção 6.1);

b) Montagem do servidor ELEPOT – será utilizado um novo microcomputador para substituir o atual. Este equipamento irá abrigar os novos módulos de controle e administração da bancada didática (seção 6.2);

c) Desenvolvimento dos softwares (embarcado e aplicação *web*) – nesta etapa os softwares projetados serão codificados e compilados em unidades prontas para serem instaladas no microcontrolador e no servidor ELEPOT (seções 6.3 e 6.4);

d) Configuração do servidor ELEPOT – esta etapa compreende a instalação dos softwares no servidor ELEPOT (seção 6.5).

Algumas tarefas de configuração e manutenção do sistema, como por exemplo criação de usuários, atribuição de privilégios e senhas, gerenciamento do desempenho do micro etc., por fugirem do escopo deste trabalho não serão aqui citadas mas cabe lembrar que algumas delas são de grande importância para o uso do sistema no dia-a-dia acadêmico e podem ser realizadas pelo técnico responsável ou administrador de rede ou sistemas do laboratório.

6.1 – Montagem do hardware

A montagem do hardware que será feita servirá para a realização dos testes propostos no próximo capítulo (homologação). Esta montagem permitirá pleno acesso aos sinais de controle dos conversores. Sua conexão com a nova base de montagem do laboratório de eletrônica de potência será feita futuramente quando este equipamento estiver completamente pronto. Para o presente momento, há ainda a possibilidade de se conectar a atual base de montagem no sistema, porém, para tanto é preciso lembrar que o microcontrolador MC56F8013 não trabalha com níveis lógicos TTL. Apesar desta incompatibilidade, não será necessário o uso de um conversor de sinais para a utilização do equipamento considerando-se que o hardware atual do laboratório é 100% compatível com o padrão TTL. O microcontrolador MC56F8013 trabalha com o nível lógico alto em 3,3V e o nível lógico baixo em 0V. Medições feitas no laboratório comprovaram que os sinais de tensão gerados no microcontrolador apresentam um erro menor que 5%, ou seja, dentro da faixa do padrão TTL, descrita na tabela a seguir:

Nível lógico	Faixa de valores aceitáveis
Baixo	0 a 0,8V
Alto	2 a 5V
Indefinido	Todos os demais valores

tabela 6.1 – Faixa de valores do padrão TTL

Neste caso, então, fica sendo possível utilizar os sinais de controle do sistema diretamente em dispositivos TTL. No caso deste projeto, o que interessa medir são os sinais de controle apenas e, por esta razão, não será utilizada a atual base de montagem. Em seu lugar será conectado um osciloscópio digital para a aferição do sinais.

O microcomputador PC deverá possuir sua própria alimentação, diretamente da rede ou com uso de estabilizador ou *no-break*. O microcontrolador também deverá ser alimentado com $9V_{CC}$ por meio de seu conversor próprio. A conexão do PC com o microcontrolador será feita por meio de um cabo serial padrão RS-232C, com um conector DB9 macho de um lado e um conector DB9 fêmea do outro. É importante ainda lembrar que o cabo serial não deverá fazer cruzamento dos fios, ou seja, deve manter a posição da pinagem. Um exemplo deste cabo pode ser visto na figura 6.1.

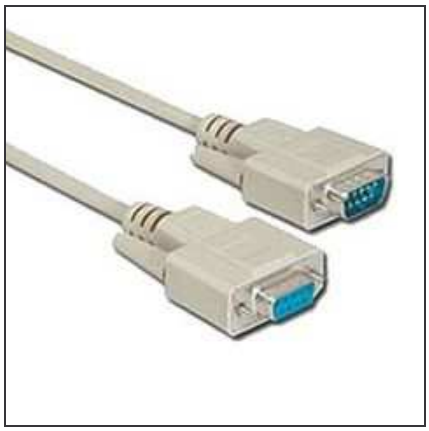


figura 6.1 – Cabo serial padrão RS-232C com conector DB9
(fonte: http://www.solbet.com/)

Já a conexão do microcontrolador com o osciloscópio deverá ser feita de acordo com a figura 6.2. É extremamente recomendado que o osciloscópio possua pelo menos dois canais para a visualização de todos os sinais simultaneamente (se necessário), ajuste de *trigger*, possibilidade de congelar a tela e opções de medição (inclusive com cursores), já que alguns dos sinais são impulsivos e possuem elevada frequência.

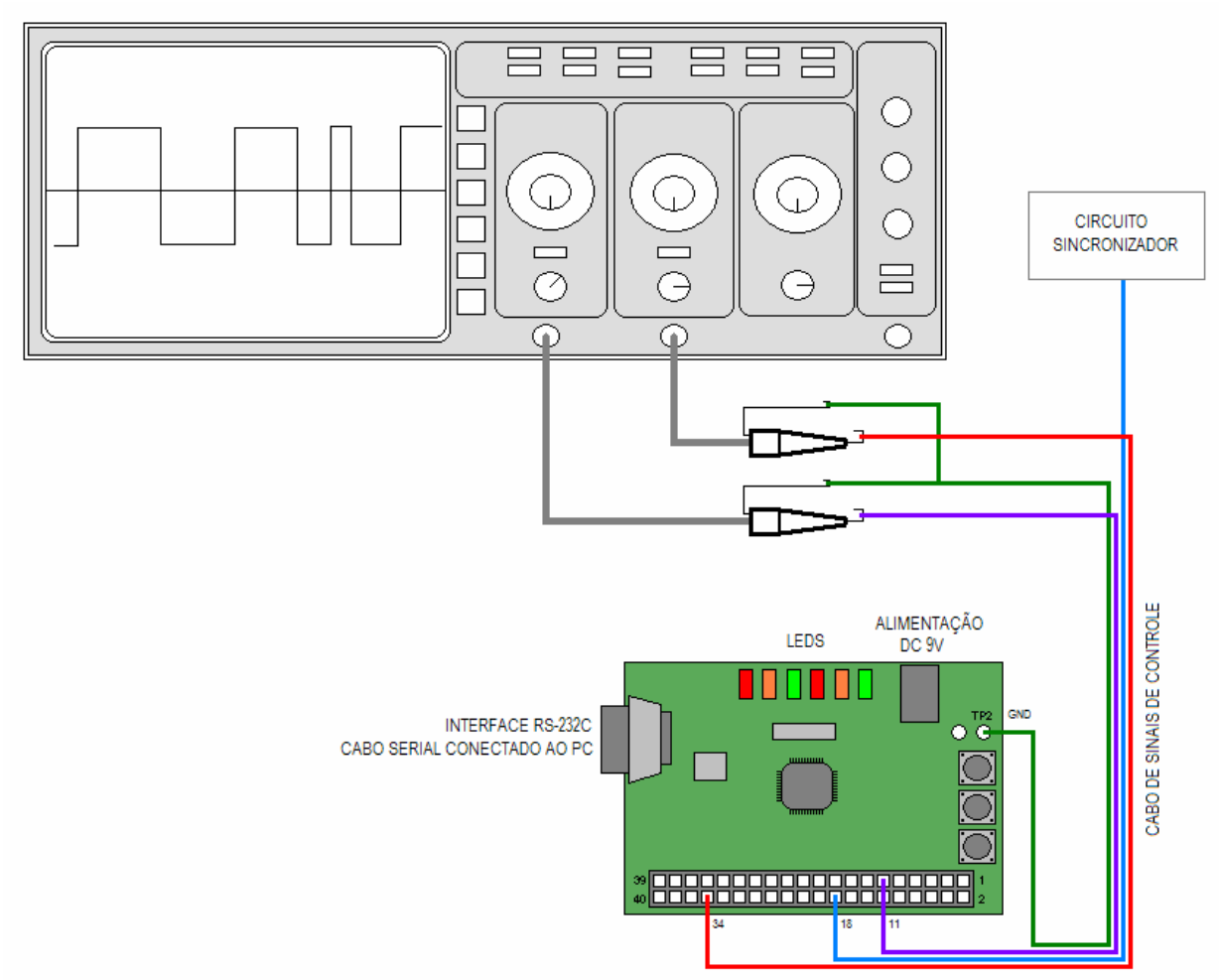


figura 6.2 – Conexões do microcontrolador com o osciloscópio

6.2 – Montagem do servidor ELEPOT

Para a montar o servidor ELEPOT o primeiro passo é a instalação do sistema operacional Kurumin Linux. Para garantir compatibilidade com o sistema que será homologado neste trabalho, o ideal é que seja utilizada a versão 4.x ou superior deste sistema operacional. Os procedimentos são listados a seguir:

- Com o CD do Kurumin inserido no *drive*, iniciar o microcomputador para que o sistema operacional seja automaticamente carregado[1]. Será então necessário que o mesmo seja instalado no disco rígido. A opção de instalar o Kurumin está na tela inicial do "painel de controle" (ver figura 6.3) ou então pode ser executada em terminal com o comando:

```
sudo kurumin-install
```

figura 6.3 – Instalação do *Kurumin Linux*
(fonte: http://www.guiadohardware.net/kurumin/)

- Durante a instalação serão feitas diversas perguntas sobre os procedimentos a serem adotados (particionamento do disco rígido, configuração de rede e segurança, usuários, instalação de utilitários e periféricos, entre outros). Uma instalação padrão pode ser feita escolhendo sempre as opções recomendadas para usuários iniciantes, conforme o próprio instalador indica. Maiores esclarecimentos sobre esta etapa podem ser obtidos na documentação específica [5, 16].

([1]) Para que isso aconteça é necessário que o micro esteja configurada para dar *boot* pela unidade de CDROM.

- Com o Kurumin corretamente instalado será necessário proceder com a instalação e configuração dos seguintes softwares:

Nome do software	Versão utilizada	Arquivo de instalação
MySQL	5.0.19	mysql-standard-5.0.19-linux-i686.tar.gz
JRE	1.5.0.06	jre-1_5_0_06-linux-i586.bin
Apache Tomcat	6.0.10	apache-tomcat-6.0.10.tar.gz

tabela 6.2 – Softwares necessários para a montagem do servidor ELEPOT

Os procedimentos para instalação do servidor de banco de dados MySQL são descritos a seguir. Para executar os comandos listados deverá ser utilizado o usuário *root* do Kurumin.

- Copiar o arquivo de instalação para o diretório /tmp e executar os comandos:

```
groupadd mysql
useradd -g mysql mysql
cd /usr/local
gunzip < /tmp/mysql-standard-5.0.19-linux-i686.tar.gz | tar xvf -
cd mysql-standard-5.0.19-linux-i686
scripts/mysql_install_db --user=mysql
chown -R root .
chown -R mysql data
chgrp -R mysql .
bin/mysqld_safe --user=mysql &
```

- Mudar para o diretório bin do MySQL e entrar no console com o comando:

```
./mysql -h localhost -u root -p
```

(quando a senha for solicitada, deixar em branco)

- Substituir a senha do usuário root do MySQL com os comandos:

```
use mysql;
update user set password=password('dfr457g') where user='root';
flush privileges;
exit
```

- Criar um *link* simbólico com o nome "mysql":

```
ln -2 /usr/local/mysql-standard-5.0.19-linux-i686 mysql
```

- Remover os arquivos que não serão mais utilizados:

```
rm /tmp/mysql-standard-5.0.19-linux-i686.tar.gz
```

Para instalar o Java Runtime Environment (JRE) e o Apache Tomcat, são necessários os seguintes procedimentos (os comandos listados deverão ser executados com o usuário *root* do Kurumin).

- Copiar os arquivos do JRE e do Tomcat para o diretório /opt e executar os comandos:

```
./jre-1_5_0_06-linux-i586.bin
rm jre-1_5_0_06-linux-i586.bin
tar -xzvf apache-tomcat-6.0.10.tar.gz
rm apache-tomcat-6.0.10.tar.gz
```

- Mudar para o diretório /etc e editar o arquivo profile incluíndo as seguintes linhas após a definição do *path*:

```
JAVA_HOME=/opt/jre1.5.0_06
CATALINA_HOME=/opt/apache-tomcat-6.0.10
export JAVA_HOME CATALINA_HOME
```

- Ir para o diretório /opt/apache-tomcat-6.0.10/conf e editar o arquivo tomcat-users.xml incluíndo a linha:

```
<user username="admin" password="dfr457g" roles="admin,manager" />
```

- Proteger o arquivo tomcat-users.xml mudando a permissão de todos os usuários de 'pode ler' para 'negado'.

figura 6.4 – Servidor ELEPOT em funcionamento

6.3 – Desenvolvimento do software embarcado

A aplicação embarcada, projetada no capítulo 4, funcionará no microcontrolador MC56F8013 porém seu desenvolvimento, compilação e transferência (*download*) é feito no "CodeWarrior Development Studio 56800/E Hybrid Controllers" (daqui por diante denominado de CodeWarrior), que atualmente possui apenas versão Windows. Uma vez codificado, compilado e transferido o programa, o mesmo fica residente na memória *flash* do microcontrolador não necessitando do microcomputador com o CodeWarrior instalado e do cabo paralelo com o adaptador JTAG.

Os procedimentos para instalação do CodeWarrior são os mesmos de qualquer software para Windows. Durante o desenvolvimento deste trabalho foi utilizada a versão 7.3 do pacote (IDE versão 5.6.1), instalada em 2 etapas: primeiramente a versão 7.0, completa, disponível no CDROM que acompanha o microcontrolador e depois a atualização (obrigatória) para a versão 7.3, obtida na internet.

O CodeWarrior não apenas permite codificar e compilar programas feitos em linguagem C compatíveis com o hardware do microcontrolador mas também depurá-lo em tempo real usando os recursos mais comuns de programação como *breakpoints*, *watches*, mapas de memória, componentes de software (chamados de *beans*) etc. Estes recursos auxiliam em muito o desenvolvimento do programa embarcado, permitindo ao desenvolvedor ganhar muito tempo.

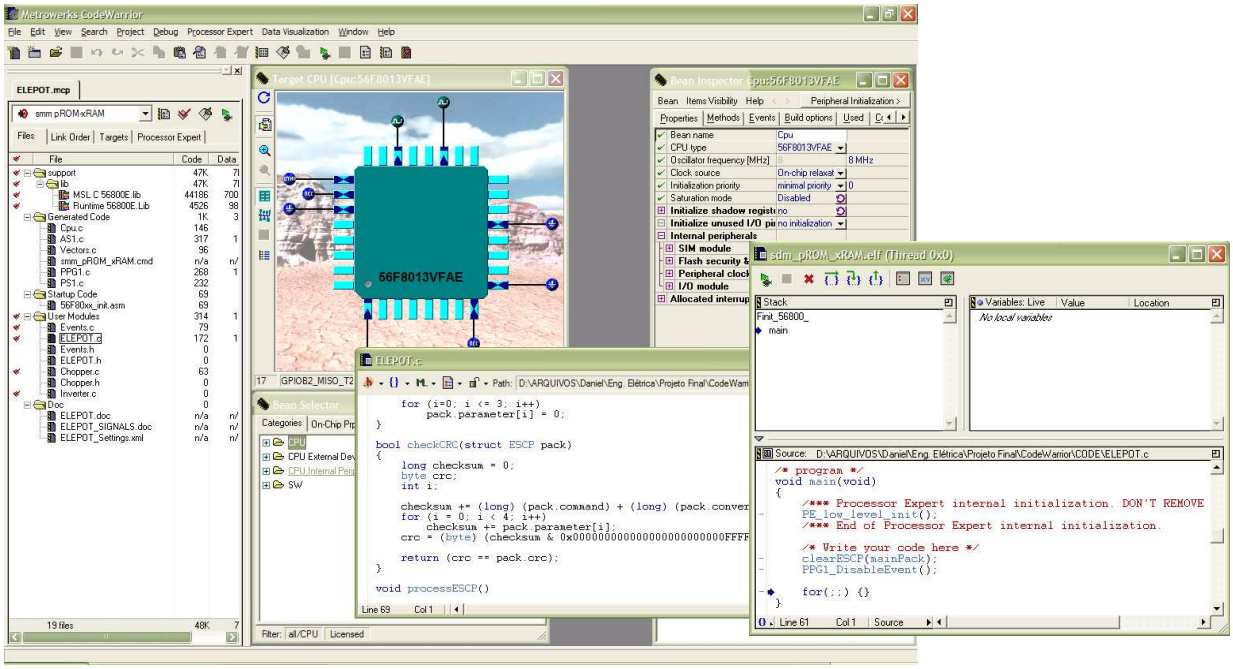


figura 6.5 – CodeWarrior IDE (em destaque um programa executando em modo de depuração)

A transferência do programa embarcado, codificado no microcomputador PC, para a memória do microcontrolador é feita por meio um cabo paralelo conectado ao adaptador JTAG.

A ligação que deve ser feita durante a etapa de desenvolvimento do programa embarcado é apresentada na figura 6.6, retirada do manual do equipamento. O cabo conecta a porta paralela do microcomputador com o conector J1 do MC56F8013.

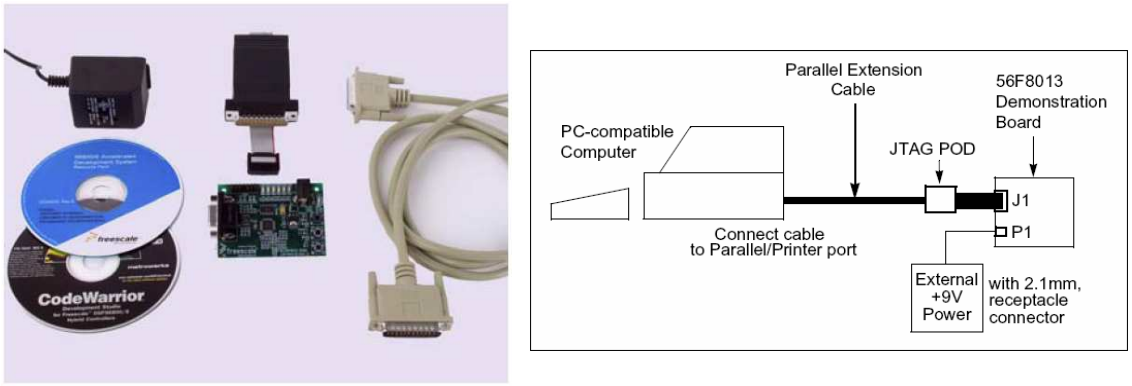


figura 6.6 – Acessórios e conexões que devem ser feitas para programação do MC56F8013
(fonte: 56F8013 Demo Board Kit Installation Guide e 56F8013 Demonstration Board User Guide)

Caso se deseje utilizar o CodeWarrior em ambiente Linux uma boa alternativa é a instalação do VMware Player, software gratuito[1] que emula um microcomputador PC completo, permitindo que um sistema operacional seja instalado dentro de outro[2]. Neste caso uma observação importante a ser feita é que para utilizar-se a porta paralela dentro da máquina virtual é necessário antes desabilitá-la no Linux, usando o comando:

```
modprobe -r lp
```

E para habilitá-la novamente o comando

```
modprobe lp
```

Deve-se usar o usuário *root* para habilitar ou desabilitar a porta paralela. Em caso de problemas com permissões de acesso, usar o comando:

```
chmod +666 /dev/parport
```

Para maiores informações sobre o VMware consultar [23].

([1]) O VMware Player apenas permite executar máquinas virtuais pré-configuradas. Para utilizá-lo é necesário obter uma máquina virtual criada no VMware Workstation (a versão comercial do VMware).

([2]) Os sistemas operacionais funcionarão de forma independente, apesar de compartilhando os mesmos recursos de hardware.

6.4 – Desenvolvimento do software da aplicação

Para desenvolver a aplicação *web*, projetada no capítulo anterior, é necessário utilizar o ambiente descrito no item 3.1, que pode ser montado, por exemplo, em Windows. Para manter a metodologia deste trabalho, o ambiente de desenvolvimento utilizado será também o Linux, porém nada impede que este roteiro seja facilmente adaptável a outros ambientes. Para os procedimentos abaixo usar o usuário *root*.

- Além do servidor de banco de dados MySQL, será necessário instalar e configurar os seguintes softwares:

Nome do software	Versão utilizada	Arquivo de instalação
JDK	1.5.0.09	jdk-1_5_0_09-nb-5_0-linux-ml.bin
NetBeans	5.0	
MySQL JDBC Driver	3.1.12	mysql-connector-java-3.1.12-bin.jar
RXTX	2.1.7	rxtx-2.1-7-bins-r2.zip

tabela 6.3 – Softwares necessários para a montagem do ambiente de desenvolvimento

- Copiar o pacote JDK + NetBeans para o diretório /tmp e executar os comandos:

```
chmod 755 /tmp/jdk-1_5_0_09-nb-5_0-linux-ml.bin
./tmp/jdk-1_5_0_09-nb-5_0-linux-ml.bin
rm /tmp/jdk-1_5_0_09-nb-5_0-linux-ml.bin
```

(será aberto um assistente de instalação: seguir todos os passos até concluir a instalação)

- Instalar o *driver* JDBC para MySQL no NetBeans:

```
cp mysql-connector-java-3.1.12-bin.jar /opt/jdk1.5.0_09/jre/lib/ext
```

(abrir o NetBeans e adicionar o *driver* usando a opção "Libraries" → "Add JAR/folder")

- Informar os seguintes parâmetros de conexão:

```
URL="jdbc:mysql://localhost/elepot"
DRIVER="com.mysql.jdbc.Driver"
```

- Descompactar o arquivo do RXTX para o diretório /tmp e instalar a API[1]:

```
cp /tmp/RXTXcomm.jar /opt/jdk1.5.0_09/jre/lib/ext
cp /tmp/librxtxSerial.so /opt/jdk1.5.0_09/jre/lib/i386
```

(**a**brir o NetBeans e adicionar a API usando a opção "Libraries" → "Add JAR/folder")

([1]) Em ambiente Windows usar a pasta ..\jre\bin e copiar nela o arquivo rxtxSerial.dll no lugar do arquivo librxtxSerial.so

6.5 – Configuração do sistema

Com o servidor ELEPOT devidamente montado e com as aplicações embarcada e *web* desenvolvidas é preciso então configurá-lo para trabalhar com estas aplicações. Para a execução dos comandos a seguir, utilizar o usuário *root* do Kurumin.

O primeiro passo é a instalação do banco de dados 'elepot' no servidor de banco de dados MySQL, conforme procedimento descrito a seguir. O arquivo elepot-install.sql contém todos os comandos necessários para criar a estrutura do banco de dados definida no capítulo 5. A listagem completa deste arquivo encontra-se no apêndice A.

- Mudar para o diretório bin do MySQL e entrar no console com o comando:

```
./mysql -h localhost -u root -p
```

(quando a senha for solicitada, usar a senha 'dfr457g' ou outra que tenha sido definida)

- Executar o script de criação do banco de dados 'elepot' e ao final sair do console:

```
\. /mnt/cdrom/elepot-install.sql
```

(substituir o caminho /mnt/cdrom se necessário)

O passo seguinte é instalar a aplicação *web* dentro do servidor de aplicação Tomcat. No jargão técnico este procedimento é chamado de *deploy* da aplicação.

- Iniciar o Tomcat com o comando:

```
./opt/apache-tomcat-6.0.10/bin/startup.sh
```

- Acessar o endereço http://localhost:8080/ em qualquer navegador *web* disponível e acessar a opção "*Tomcat manager*". Na tela de *login*, usar o usuário e senha definidos no final do item 6.2, ou seja, usuário 'admin' e senha 'dfr457g'.

- Acessar a opção "D*eploy WAR file*", selecionar o arquivo elepot.war (no CD) e clicar em "D*eploy*". Fechar o navegador *web*.

- Copiar todos os arquivos com a extensão '.jpg' (figuras dos roteiros dos experimentos) para dentro do diretório /opt/apache-tomcat-6.0.10/webapps/elepot/img

Com todos estes procedimentos realizados o servidor ELEPOT estará pronto para uso, porém o ideal é que este possua uma inicialização automática, de modo que os equipamentos da bancada didática ao serem desligados e ligados novamente estejam prontos para operar sem a necessidade de intervenção humana.

No caso do microcontrolador da bancada didática isso já acontece devido à sua arquitetura. O software embarcado é gravado em memória do tipo *flash* e por isso não necessita ser reprogramado ao ligar. Além disso, o microcontrolador MC56F8013 já possui um sistema interno de inicialização (*bootloader*) que executa automaticamente o programa que reside em sua memória.

Torna-se então necessário que o software que reside no microcomputador PC seja também iniciado automaticamente, deixando assim o sistema como um todo completamente inicializável. Os seguintes procedimentos devem então ser executados:

- Para iniciar automaticamente o servidor de banco de dados MySQL é preciso primeiro copiar o arquivo mysql.server para dentro de /etc/init.d

```
cp /usr/local/mysql/support-files/mysql.server /etc/init.d/mysql
cd /etc/init.d
```

- Depois configurar o utilitário update-rc do Kurumin para iniciá-lo automaticamente:

```
update-rc.d mysql defaults
```

- Para iniciar automaticamente o servidor de aplicação Tomcat é preciso primeiro copiar o arquivo catalina.sh para dentro de /etc/init.d

```
cp /opt/apache-tomcat-6.0.10/bin/catalina.sh /etc/init.d/tomcat
cd /etc/init.d
```

- Abrir o arquivo /etc/profile do Kurumin e copiar o trecho de definição do '*locale*' (que começa na linha "# Set LOCALE" e termina na linha "# END LOCALE") para dentro do arquivo tomcat, no diretório /etc/init.d, acrescentado-o logo no ínicio do arquivo (após a linha "# Environment Variable Prequisites").

- Acrescentar também as seguintes linhas no ínicio deste arquivo (logo após as linhas do item anterior):

```
CATALINA_HOME=/opt/apache-tomcat-6.0.10
JAVA_HOME=/opt/jre1.5.0_06
```

- Fechar o arquivo profile do Kurumin e salvar e fechar o arquivo tomcat com as mudanças feitas.

- Depois configurar o utilitário update-rc do Kurumin para iniciá-lo automaticamente:

```
update-rc.d tomcat defaults
```

- Para criar um atalho de acesso ao sistema, utilizar a opção "atalho p/ aplicativo" do *menu* "novo" do Kurumin e usar o comando:

```
firefox http://localhost:8080/elepot/
```

(ou então substituir 'firefox' por outro navegador *web* como por exemplo 'konqueror')

Se por algum motivo for desejado desativar a inicialização automática do banco de dados e/ou do servidor de aplicação, usar o comando:

```
update-rc.d nome_do_servico remove
```

(substituindo nome_do_servico por 'mysql' ou 'tomcat')

A seguir é listado um resumo dos principais comandos para utilizar e gerenciar o banco de dados MySQL e o servidor de aplicação Apache Tomcat:

- Comandos de parar, iniciar e reiniciar o MySQL (respectivamente):

```
./diretorio_do_mysql/support-files/mysql.server stop
./diretorio_do_mysql/support-files/mysql.server start
./diretorio_do_mysql/support-files/mysql.server restart
```

(onde 'diretorio_do_mysql' é o caminho onde o MySQL está instalado)

- Gerenciamento do MySQL (acesso ao banco de dados, consultas, modificações etc.):

```
./mysql -h localhost -u root -p
```

(quando a senha for solicitada, usar a senha 'dfr457g' ou outra que tenha sido definida)

- Comandos de parar, iniciar, reiniciar e visualizar o *log* do Tomcat (respectivamente):

```
./opt/apache-tomcat-6.0.10/bin/shutdown.sh
./opt/apache-tomcat-6.0.10/bin/startup.sh
./opt/apache-tomcat-6.0.10/bin/catalina.sh restart

cat /opt/apache-tomcat-6.0.10/logs/catalina.out
```

- Gerenciamento do Tomcat (inclusive disponibilização de aplicações - *deploy*):

```
http://localhost:8080/
```

7 – Homologação

Com o sistema completamente desenvolvido e implementado, a última etapa deste trabalho será a sua homologação. Os testes feitos no laboratório têm por finalidade validar o sistema e demonstrar sua viabilidade de utilização. Os testes aqui apresentados não conseguirão esgotar todas as possibilidades existentes, mas com certeza darão ao projeto a robustez necessária para ser bem aproveitado e, conforme citado na introdução deste trabalho, auxiliar na evolução necessária do laboratório de eletrônica de potência da UFRJ.

A etapa de homologação será dividida em 6 partes, sendo a primeira referente ao novo módulo administrativo e as seguintes referentes aos experimentos. Sobre essas etapas, é importante ressaltar que como o projeto do novo módulo de chaves da bancada didática e sua utilização não estão no escopo deste trabalho, a homologação dos experimentos será feita por meio da homologação dos sinais de controle gerados pelo microcontrolador, já que este é o ponto principal do sistema.

A homologação do módulo administrativo será feita utilizando-se todas as funcionalidades disponíveis na aplicação *web* (cadastros, relatórios, *login* e troca de senha, páginas de comando dos conversores, de instruções para os alunos etc.) e fazendo-se alguns testes de segurança e integridade da aplicação.

A homologação dos experimentos será feita por meio da análise em osciloscópio conforme a montagem de hardware do item 6.1 dos sinais de controle gerados pelo sistema. Para auxiliar nesta análise serão utilizadas as simulações realizadas em PSCAD durante o curso do laboratório de eletrônica de potência. Apesar de serem cinco os experimentos do laboratório, há somente 3 conversores envolvidos e por conta disso os experimentos 1 e 2 (*chopper*), e os experimentos 4 e 5 (retificador monofásico) possuirão suas homologações praticamente iguais, já que as diferenças entre esses experimentos estão no circuito externo, permanecendo o sistema de controle o mesmo.

7.1 – Homologação do módulo administrativo

A homologação do módulo administrativo será feita por meio de exemplos de uso das funcionalidades do sistema e em tentativas de acesso não autorizado e acesso simultâneo às páginas de comando da bancada didática. Será utilizado o ambiente de produção (final) do sistema com o navegador *web* Konqueror (e também Firefox no caso do item 7.1.8).

7.1.1 – *Login* do usuário

Ao acessar a aplicação, o usuário encontra a tela inicial do sistema, composta por dois formulários: um para *login* no laboratório e outro, somente para professores (e por isso há somente o campo "senha"), que dará acesso ao módulo de manutenção.

figura 7.1 – Tela inicial do sistema

Como não há alunos previamente cadastrados, a primeira etapa será acessar o módulo de manutenção do sistema. Ao tentar usar uma senha qualquer, o sistema faz a verificação e fornece ao usuário uma mensagem de alerta, impedindo seu *login*. Com a senha correta (a senha inicial é "professor") o *login* é registrado e o *menu* principal do módulo de manutenção é exibido.

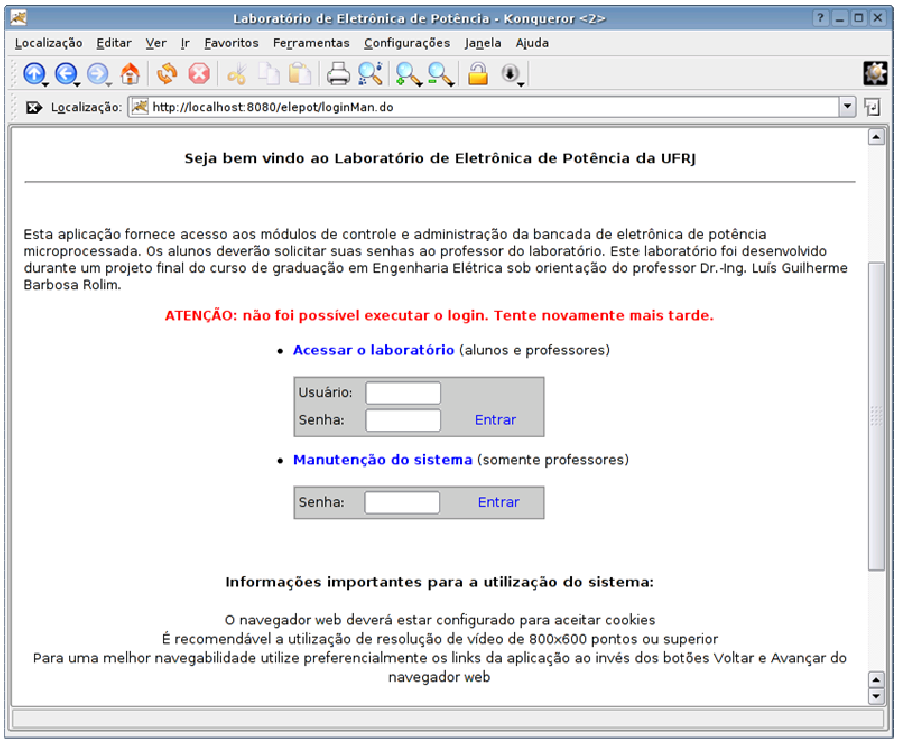


figura 7.2 – Tentativa de *login* inválida

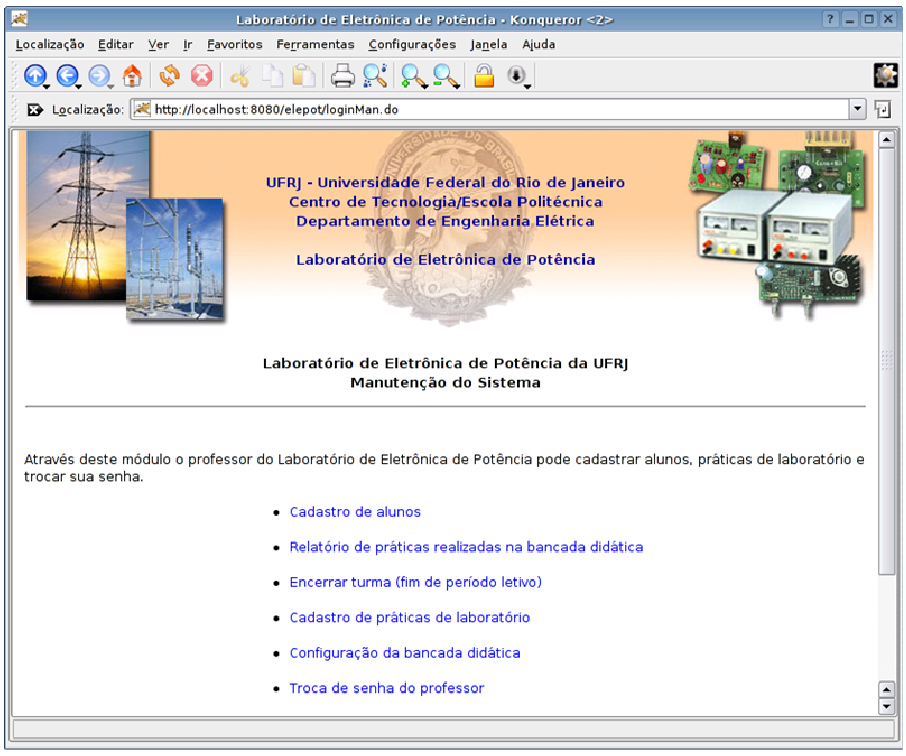


figura 7.3 – Módulo de manutenção do sistema

Foi ainda realizada uma outra tentativa de acesso não autorizado à aplicação, digitando-se a URL http://localhost:8080/elepot/loginMan.do diretamente no navegador *web* e sem efetuar o *login*, porém a mesma encerrou-se sem sucesso.

7.1.2 – Cadastro de alunos

O cadastro de alunos permite que os alunos (usuários) sejam cadastrados, passando não apenas a constar no banco de dados mas também recebendo um *login* e senha para acesso ao sistema. O cadastro de alunos permite incluir, alterar, excluir e pesquisar pelo nome e/ou identificação (matrícula) todos os alunos. A pesquisa é feita somente com os alunos ativos porém no formulário há uma opção para incluir também os alunos inativos.

A homologação deste cadastro foi feita por meio da execução de todas as funcionalidades aqui descritas, obtendo-se êxito em todas elas. A figura a seguir exibe algumas das telas.

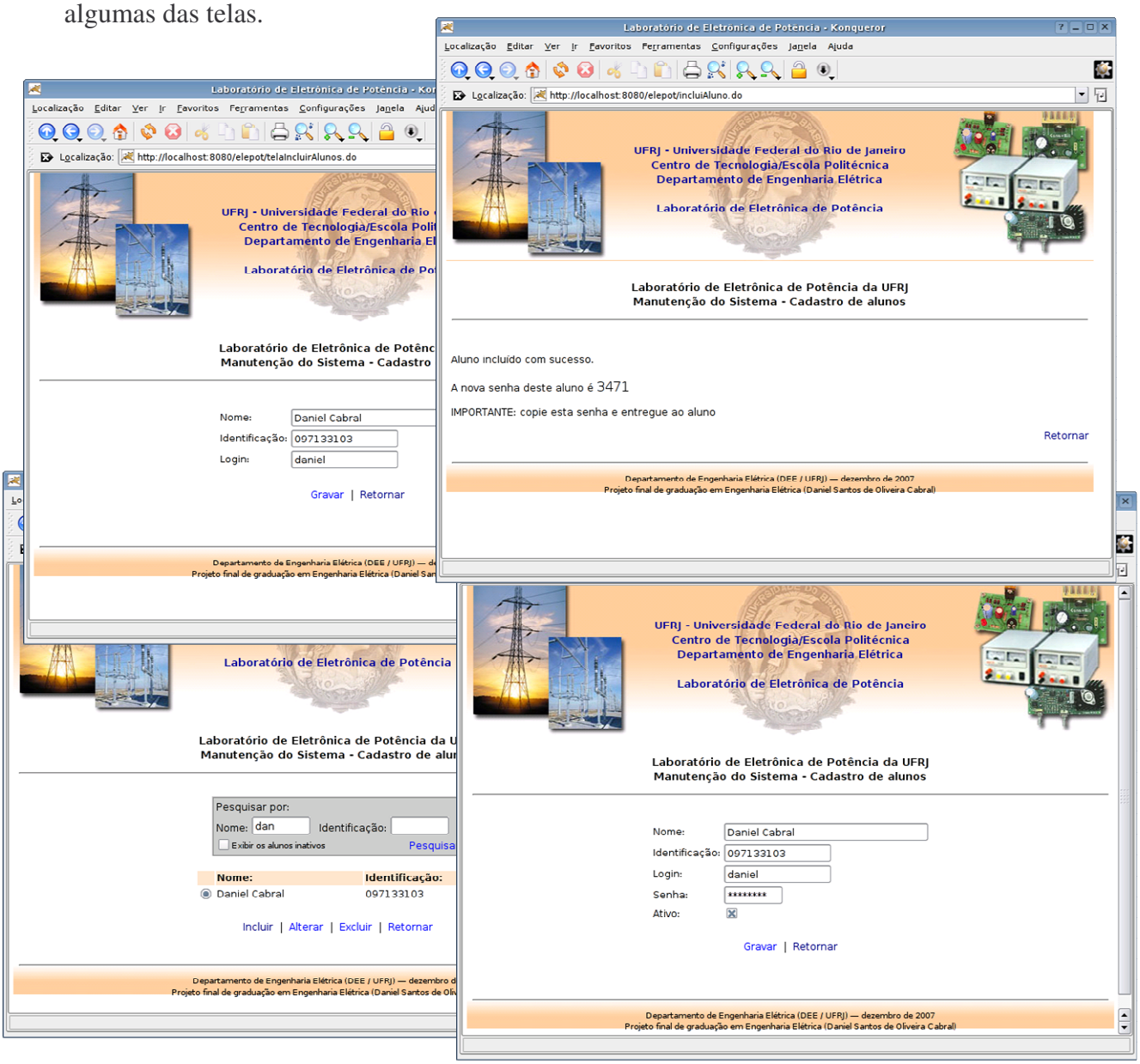


figura 7.4 – Resultados da homologação do cadastro de alunos

7.1.3 – Cadastro de práticas de laboratório

O cadastro de práticas de laboratório serve para que os professores modifiquem detalhes acerca dos experimentos como as instruções, a descrição ou a ordem destes. Este cadastro já é preenchido no *script* de criação do banco de dados com o conteúdo do atual roteiro de atividades do laboratório [3] e, apesar de possuir as opções de inclusão de novos ou exclusão dos atuais experimentos, ele praticamente se manterá o mesmo. O cadastro de práticas de laboratório permite incluir, alterar, excluir e pesquisar pelo número e/ou nome dos experimentos. Um detalhe que deve ser citado é que os campos "descrição" e "instruções" permitem códigos de formatação HTML, dando ao professor toda a liberdade para incluir um visual elegante e didaticamente eficiente nas suas instruções – inclusive com conteúdo multimídia (figuras e animações, sons, vídeos etc.).

A homologação deste cadastro foi feita por meio da execução de todas as funcionalidades aqui descritas, obtendo-se êxito em todas elas. A figura a seguir exibe algumas das telas.

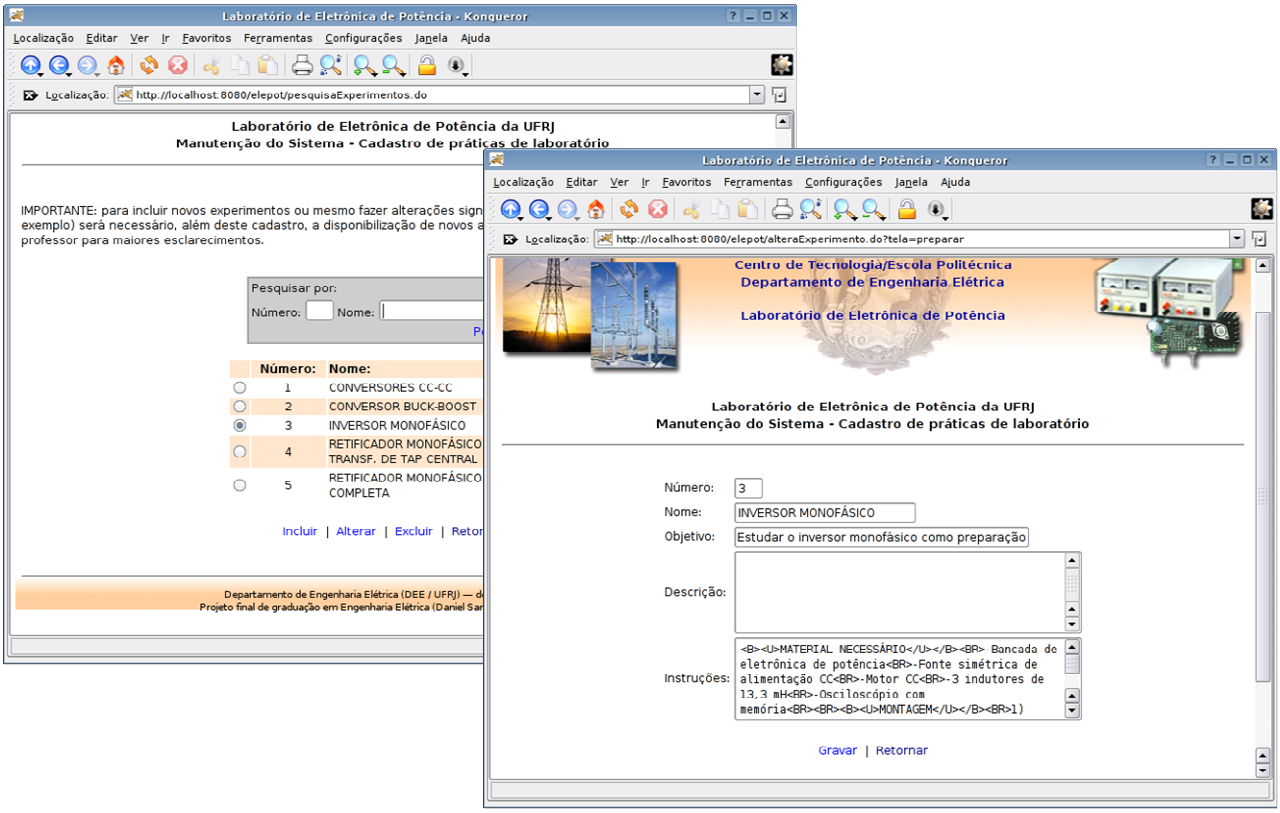


figura 7.5 – Resultados da homologação do cadastro de práticas de laboratório

7.1.4 – Configuração da bancada didática

Com intuito de tornar o sistema portável e flexível, algumas das principais configurações precisam ser parametrizadas pelo administrador. Esta funcionalidade permite ao professor selecionar qual sistema operacional e porta serial será utilizada e qual será o tempo de sessão do aluno dentro do módulo de comando dos conversores. A tela de configuração da bancada didática permite então alterar esses parâmetros.

A homologação desta funcionalidade foi feita por meio da modificação dos parâmetros do sistema, obtendo-se êxito nesta operação. Um detalhe importante sobre o tempo de sessão é que este parâmetro apenas atua na sessão definida dentro do módulo do laboratório (páginas de comando do conversor). O tempo limite de sessão para o restante do sistema é definido dentro da aplicação Java e permanece fixo já que nos outros módulos não há impedimentos quanto ao acesso simultâneo. Observou-se que passado esse tempo limite (fixo) a sessão expira e o usuário é redirecionado para a tela de *login* do sistema.

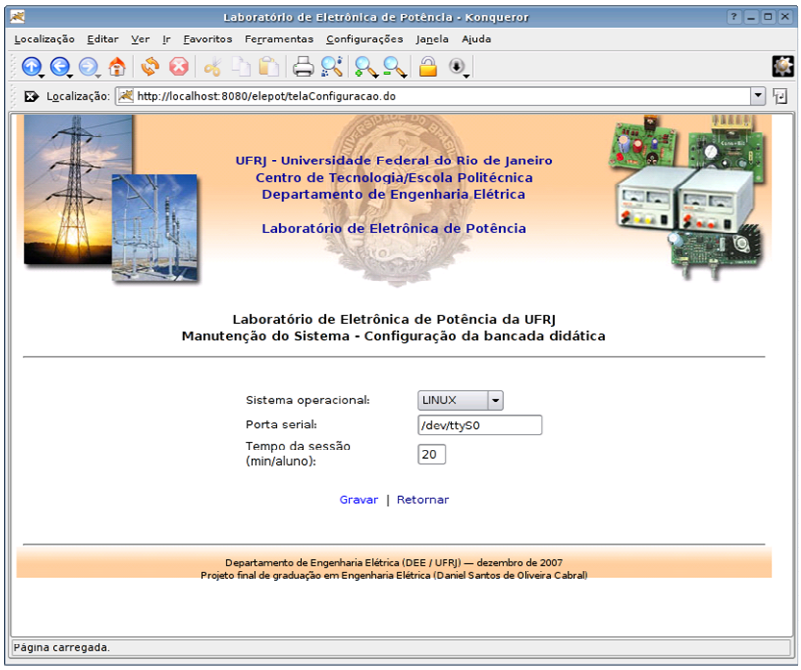


figura 7.6 – Resultado da homologação da configuração da bancada didática

7.1.5 – Encerrar turma (fim de período letivo)

Ao fim do período letivo o professor deverá tornar seus alunos inativos para que estes não se misturem com os novos alunos a serem cadastrados. Por se tratar de uma tarefa rotineira (acontecerá sempre ao término do curso) esta funcionalidade foi desenvolvida para que o professor não precise alterar aluno por aluno. Ao clicar no link "encerrar turma" todos os alunos atualmente ativos são transformados em inativos.

A homologação desta funcionalidade foi feita usando-se o correspondente *hyperlink* e observando-se que após a mensagem de sucesso, de fato, os alunos cadastrados e ativos passaram a ser inativos, obtendo-se êxito nesta operação.

7.1.6 – Relatório de práticas realizadas na bancada didática

Toda vez que os alunos acessam as páginas de comando dos conversores é criado um registro na tabela "PRATICA". O conteúdo desta tabela pode então ser acessado pelo professor por meio do relatório de práticas realizadas, facilitando seu acompanhamento das atividades dos alunos no laboratório. Esse relatório possui um filtro de data de modo a somente exibir os registros referentes ao período desejado.

Para homologar este relatório, as páginas de comando foram duas vezes acessadas pelo aluno de teste e depois o mesmo foi gerado para a data atual, obtendo-se êxito nesta operação.

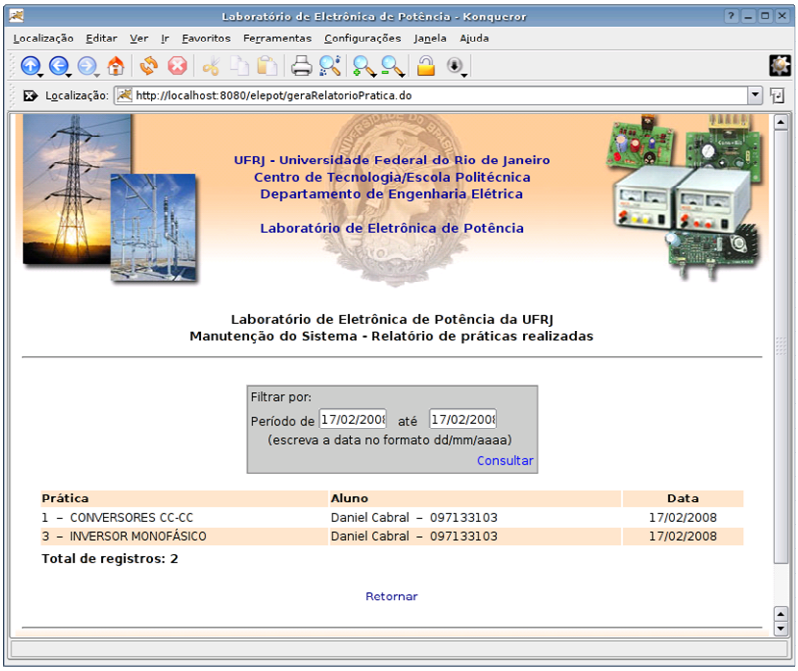


figura 7.7 – Resultado da homologação do relatório de práticas realizadas

7.1.7 – Troca de senha do professor

A última funcionalidade do módulo de manutenção do sistema é a troca de senha do professor. Ao acessar a tela correspondente o professor digita e confere a nova senha desejada, o sistema efetua sua troca e exibe uma mensagem. A homologação desta funcionalidade foi feita trocando-se a senha do professor e, recebida a mensagem de sucesso, efetuando-se uma saída do sistema (*logout*) e voltando a acessá-lo (*login*), sendo confirmada a mudança da senha.

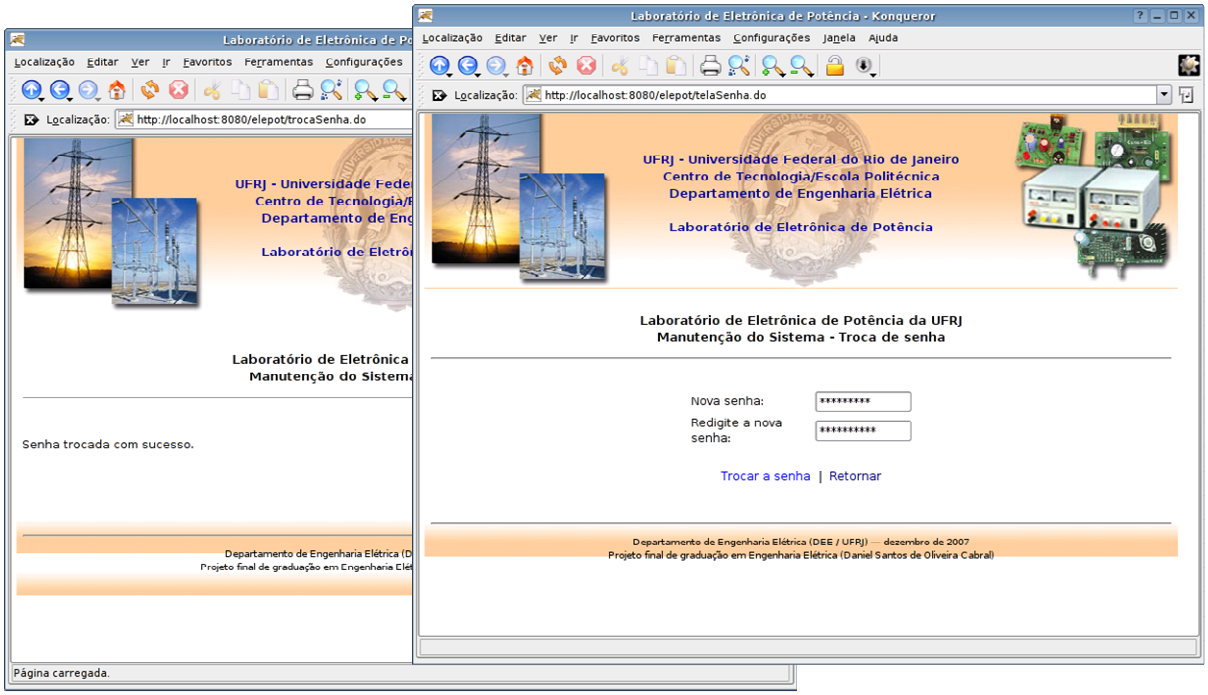


figura 7.8 – Resultado da homologação da troca de senha do professor

Um detalhe interessante de observar é que, conforme já explicado nos capítulos referentes ao projeto do sistema, a senha é armazenada no banco de dados apenas como número, resultado de uma operação matemática irreversível, conforme figura 7.9.

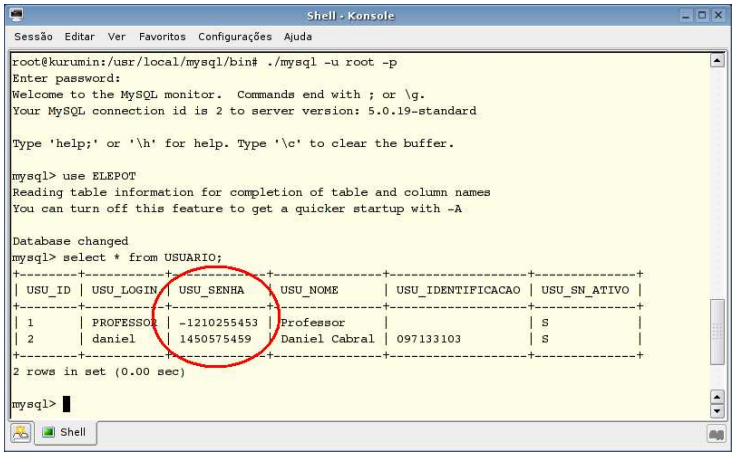


figura 7.9 – Verificação da senha armazenada no banco de dados

7.1.8 – Laboratório

O módulo do laboratório contém as páginas de comando dos conversores. Na tela inicial deste módulo existe um *menu* de opções com os experimentos disponíveis e a funcionalidade de troca de senha do aluno, similar a troca de senha do professor.

Nesta tela principal do laboratório aparece o título, o objetivo e *hyperlinks* para a descrição e as instruções do experimento. O *hyperlink* principal, com o número do experimento dá acesso à página de comando do conversor correspondente.

A homologação deste módulo foi feita então acessando-se os *hyperlinks* e verificando o conteúdo apresentado na tela. Em todas as verificações obteve-se êxito. Na figura 7.10 é possível observar a tela de instruções usando recursos HTML. A tela de descrição do experimento é idêntica, porém seu conteúdo inicial será vazio, ficando a cargo do professor preencher com o texto que achar conveniente.

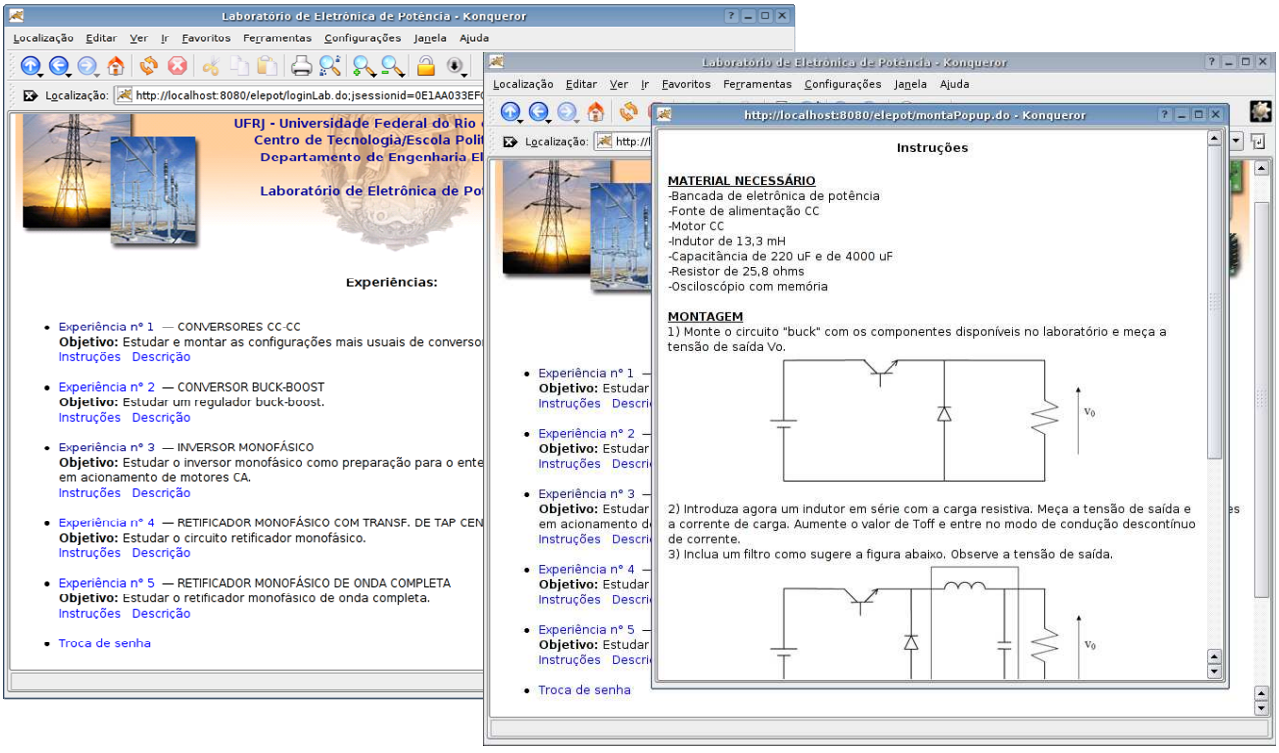


figura 7.10 – Módulo laboratório

A troca de senha do aluno funciona exatamente da mesma forma da troca de senha do professor. A homologação foi realizada efetuando-se uma troca na senha do aluno e acessando-se novamente o sistema (*logout* e *login*), tendo o procedimento sido realizado com êxito. Um detalhe a observar é que o professor, usando o cadastro de alunos pode, a qualquer instante e sem conhecer a senha do aluno trocá-la – essa opção pode ser útil por exemplo quando o aluno esquecer sua própria senha.

Uma importante estapa da homologação do laboratório é a realização do teste de simultaneidade de acesso às páginas de comando do conversor. Conforme já explicado, o hardware por ser unitário e mono-usuário permite que somente um único usuário por vez possa controlá-lo. Então a aplicação deverá apresentar o seguinte comportamento, verificado com êxito durante os testes realizados:

- Caso um usuário entre na página de comando de qualquer um dos experimentos, a mesma deverá ser "bloqueada" para outros acessos – isso é feito no banco de dados;
- Um outro usuário qualquer pode entrar no sistema porém ao tentar acessar qualquer uma das páginas de comando o mesmo será impedido e será exibido um aviso na tela, ou seja, ele ficará limitado a apenas acessar as informações textuais dos experimentos ou trocar sua senha até que a bancada didática seja liberada – o que acontecerá quando o usuário que a bloqueou sair da página usando a opção "voltar" ou seu tempo de sessão expirar (neste caso é o tempo de sessão definido na configuração da bancada didática);
- Se o mesmo usuário que já está mantendo o bloqueio na página de comando tentar efetuar novo *login* no sistema (seja de uma outra máquina ou mesmo de uma outra janela – independentemente de ser ou não o mesmo navegador *web*) este será impedido até que seu *login* anterior seja encerrado pelo próprio usuário ou expiração da sessão.

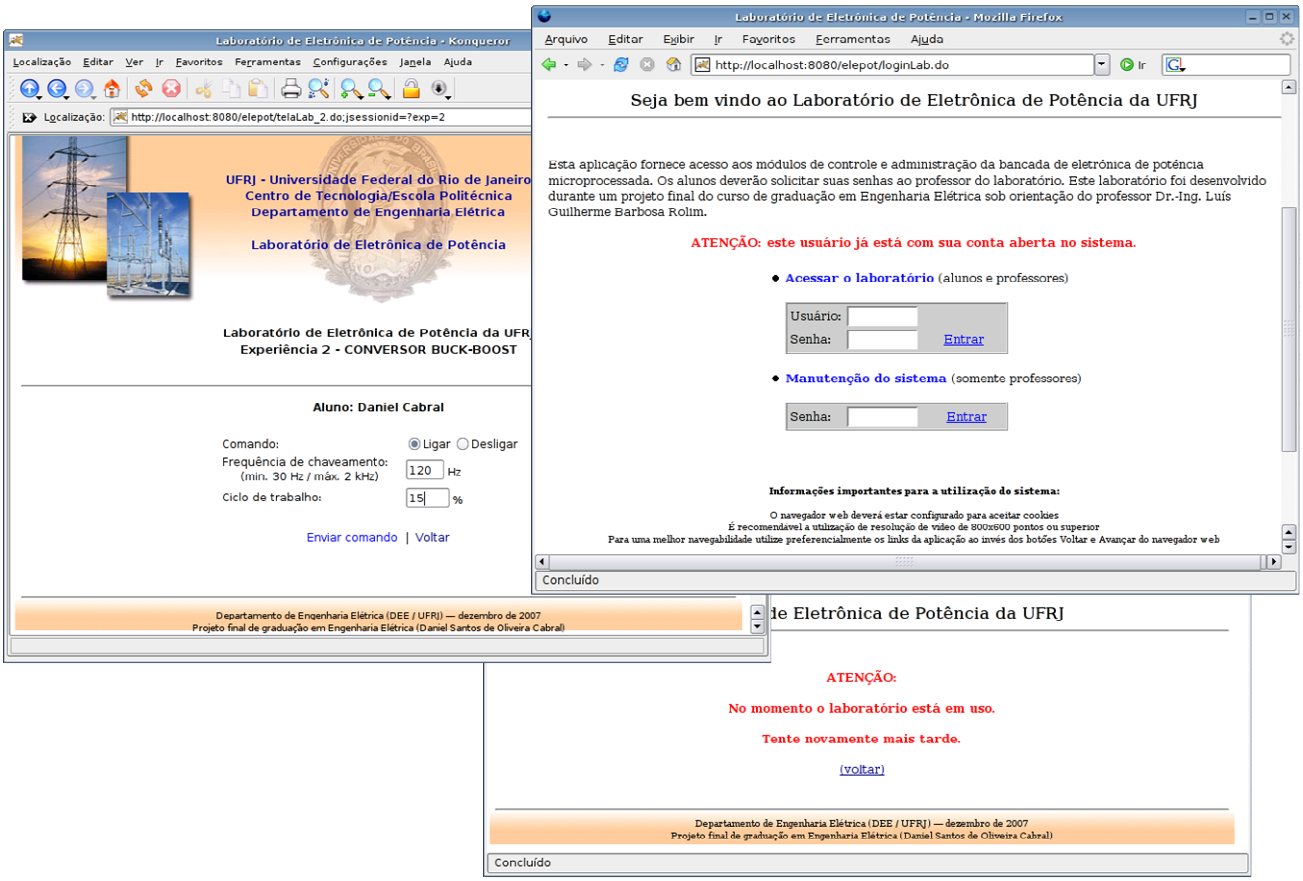


figura 7.11 – Testes com tentativas de acesso simultâneo

7.2 – Homologação do experimento 1

O experimento de número 1 do laboratório de eletrônica de potência utiliza um conversor *chopper* (CC-CC) monofásico de um quadrante em duas de suas típicas configurações: *buck* (abaixador de tensão) e *boost* (elevador de tensão). O controle de ambos os conversores é o mesmo. O que determina se ele será um abaixador ou elevador de tensão é o circuito externo (circuito de potência). Como o foco deste trabalho está no controle do *chopper*, ficará como sugestão a leitura de Mohan [14] para um pleno entendimento de como funciona este circuito.

Esta homologação consiste então de analisar os sinais de controle do conversor *buck* e do conversor *boost* e compará-los com o obtido experimentalmente usando todo o equipamento desenvolvido.

Os itens 7.2.1 e 7.2.2 ilustram um exemplo de cada um destes conversores. Esses exemplos foram utilizados nas simulações em PSCAD realizadas no laboratório de eletrônica de potência. A partir deles será possível estabelecer alguns parâmetros para esta homologação.

7.2.1 – O conversor *buck*

O exemplo a seguir mostra um conversor *buck* usado no item 3 do roteiro do experimento 1. Os parâmetros deste conversor são a tensão de saída desejada (V_{out}) e a frequência de chaveamento (f_{chav}) a ser utilizada. No exemplo em questão deseja-se uma saída de aproximadamente 25V com frequência de chaveamento 140Hz. Observar que a tensão média obtida (V_o) ficou muito próxima do desejado. O circuito de controle atuará então produzindo pulsos de tensão PWM na chave que levará o circuito de potência a produzir a saída desejada (figura 7.13).

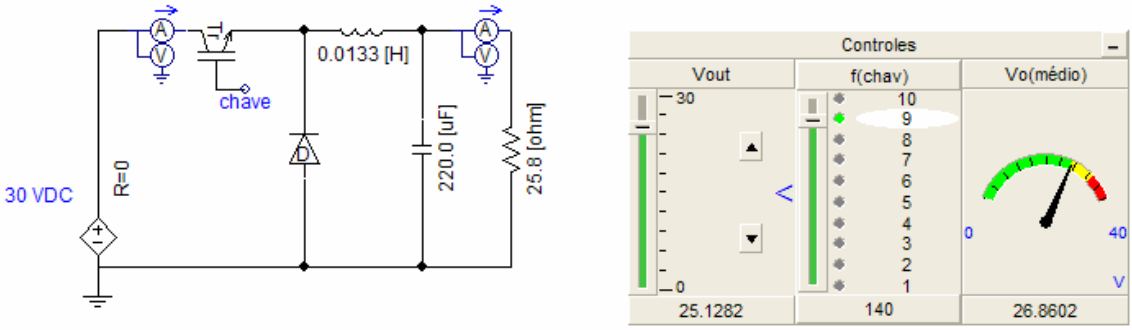


figura 7.12 – Conversor *buck* de exemplo

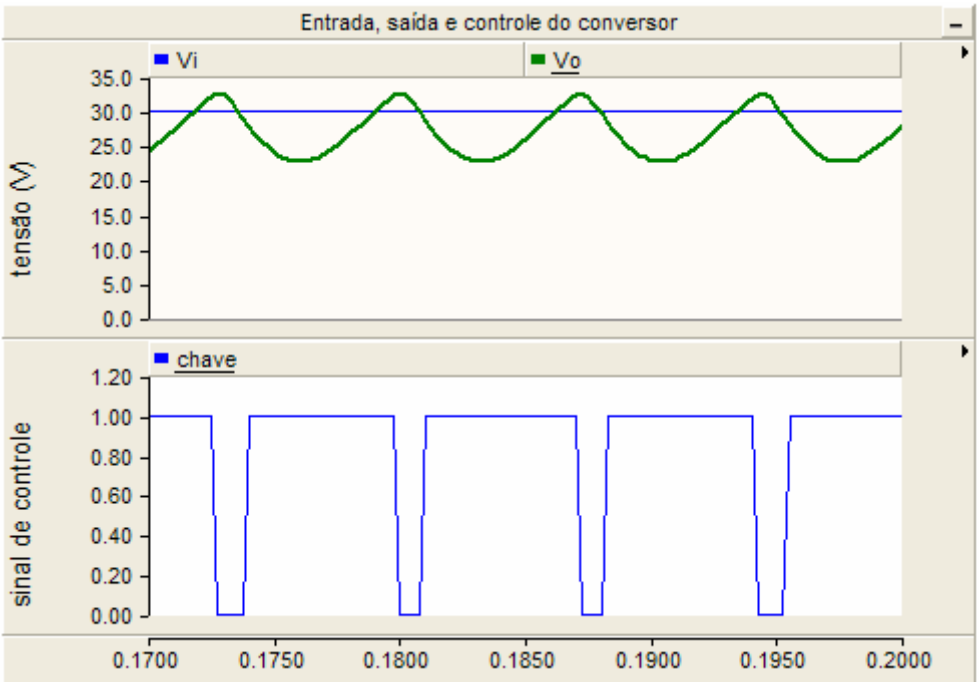


figura 7.13 – Tensão de entrada, saída e controle do conversor *buck* de exemplo

O ciclo de trabalho do conversor *buck* é dado pela expressão:

$$DC(\%) = \frac{V_o}{V_i} \times 100 \cong \frac{25}{30} \times 100 = 83{,}33\%$$

7.2.2 – Conversor *Boost*

O exemplo a seguir mostra agora um conversor *boost*, usado nos itens 5 e 6 do roteiro do experimento 1 (este item sugere a utilização de um motor CC como carga). Os parâmetros deste conversor são também a tensão de saída desejada (V_{out}) e a frequência de chaveamento (f_{chav}). Neste exemplo deseja-se uma saída de cerca de 87V com frequência de chaveamento 90Hz. A tensão média obtida (V_o) ficou também muito próxima do desejado. O circuito de controle atuará do mesmo modo, produzindo pulsos de tensão PWM na chave e levando o circuito de potência a produzir a saída desejada.

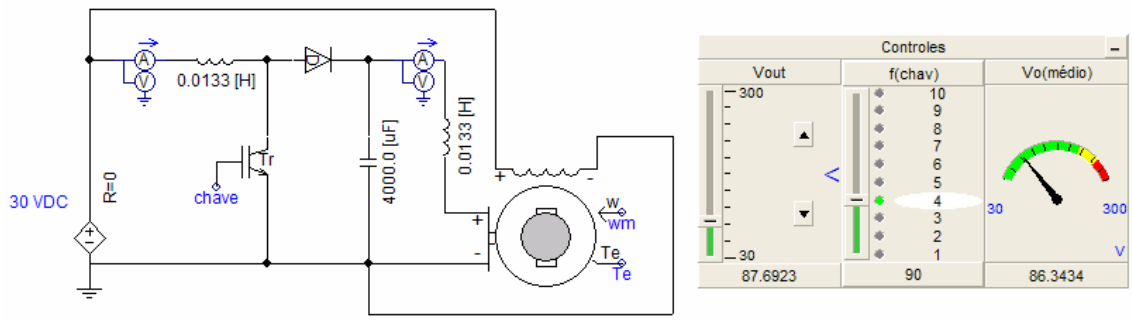


figura 7.14 – Conversor *boost* de exemplo

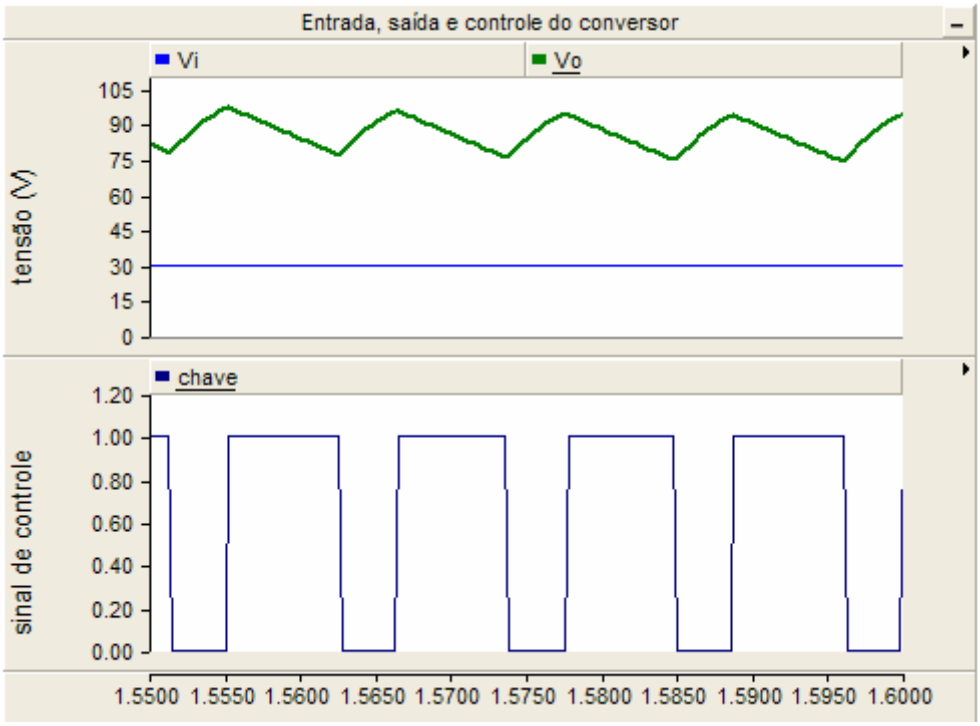


figura 7.15 – Tensão de entrada, saída e controle do conversor *boost* de exemplo

O ciclo de trabalho do conversor *boost* é dado pela expressão:

$$DC(\%) = \left(1 - \frac{V_i}{V_o}\right) \times 100 \cong \left(1 - \frac{30}{87}\right) \times 100 = 65{,}52\%$$

7.2.3 – Resultados da homologação

O comportamento do controle das chaves do *chopper* deverá então seguir o apresentado nos itens 7.2.1 e 7.2.2. A página de comando do experimento 1 apresenta quatro opções para o usuário:

- Comando do conversor: ligar ou desligar. Ao ligar, o microcontrolador acende o LED verde e começa a produzir o sinal de controle de acordo com os demais parâmetros e ao desligar o LED vermelho é acesso e o sinal de controle fixado em zero. É importante lembrar que o MC56F8013 não trabalha com os níveis lógicos TTL. Seus níveis lógicos baixo e alto são, respectivamente, 3,3V e 0V.
- Tipo de conversor: *buck* ou *boost*. Este parâmetro serve apenas para indicar à aplicação de qual conversor se trata para os cálculos dos resultados esperados que são exibidos na tela pois tanto o controle do *buck* como o controle do *boost* são exatamente o mesmo. Conforme citado anteriormente, o que determinará seu funcionamento como abaixador ou elevador de tensão será apenas o circuito de potência.

- Frequência de chaveamento: qualquer valor real, de 30 a 2.000Hz. Determina qual será a frequência de chaveamento desejada para o conversor. Caso o usuário digite um valor fora do limite aceito, a frequência usada será o valor mais próximo neste limite.
- Ciclo de trabalho: qualquer valor real entre 0 e 99,999%. Determina qual será o ciclo de trabalho desejado para o conversor. Caso o usuário digite um valor fora do limite aceito, o ciclo de trabalho usado será o valor mais próximo neste limite. Um detalhe neste parâmetro é o valor máximo não ser 100% pois no *boost* um ciclo de trabalho de 100% levaria a tensão de saída a infinito, um valor nada desejável.

Foram executados testes com este conversor e os seguintes resultados obtidos:

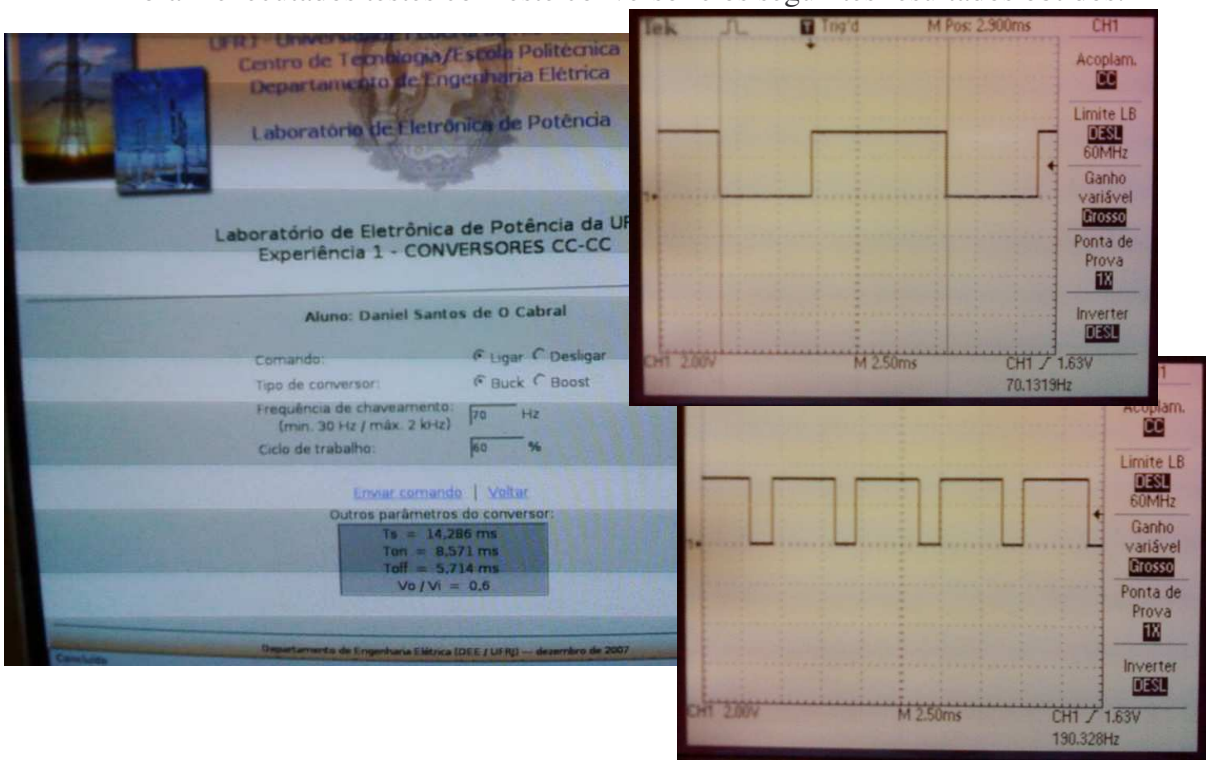


figura 7.16 – Resultados da homologação do experimento 1

Teste	Período (ms)		T_{on} (ms)		T_{off} (ms)	
	Teórico	Real	Teórico	Real	Teórico	Real
1) *Buck*	14,286	14,260	8,571	8,600	5,714	5,700
2) *Buck*	1,471	1,468	0,221	0,220	1,250	1,250
3) *Boost*	5,263	5,256	3,947	3,920	1,316	1,320
4) *Boost*	0,714	0,713	0,179	0,176	0,536	0,536

tabela 7.1 – Resultados da homologação do experimento 1

Todos os resultados foram satisfatórios ficando o controle do experimento 1 adequado à sua futura implementação prática no laboratório de eletrônica de potência.

7.3 – Homologação do experimento 2

O experimento de número 2 do laboratório de eletrônica de potência utiliza novamente um conversor *chopper* (CC-CC) monofásico de um quadrante, porém, desta vez na sua configuração *buck-boost* (abaixador-elevador de tensão). O controle deste conversor será o mesmo do experimento 1 pois novamente o que determinará seu comportamento como abaixador e elevador de tensão será apenas seu circuito de potência.

Esta homologação consiste então de analisar os sinais de controle do conversor *buck-boost* e compará-los com o obtido experimentalmente. Apesar do controle ser igual ao apresentado no item anterior, uma breve análise será feita com um circuito de exemplo usado na simulação em PSCAD. Para mais detalhes acerca do conversor *buck-boost* consultar Mohan [14].

7.3.1 – O conversor *buck-boost*

O exemplo a seguir mostra um conversor *buck-boost* usado no experimento 2. Os parâmetros deste conversor são os mesmos do experimento 1, ou seja, a tensão de saída desejada (V_{out}) e a frequência de chaveamento (f_{chav}) a ser utilizada, sendo que neste caso do exemplo a frequência é pré-fixada em 3.000Hz. No exemplo em questão deseja-se uma saída de aproximadamente 15V e é possível observar que a tensão média obtida (V_o) ficou muito próxima do desejado. Assim como nos conversores *buck* e *boost*, o circuito de controle atuará produzindo pulsos de tensão PWM na chave que levará o circuito de potência a produzir a saída desejada (figura 7.17).

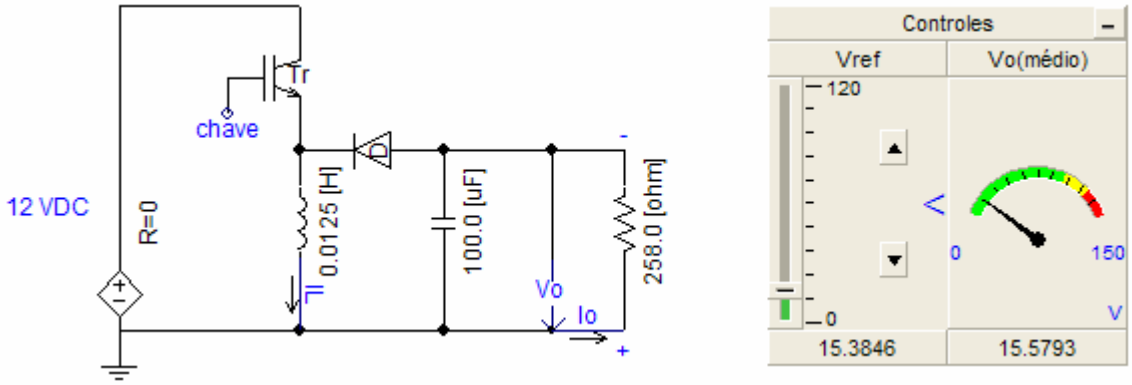


figura 7.17 – Conversor *buck-boost* de exemplo

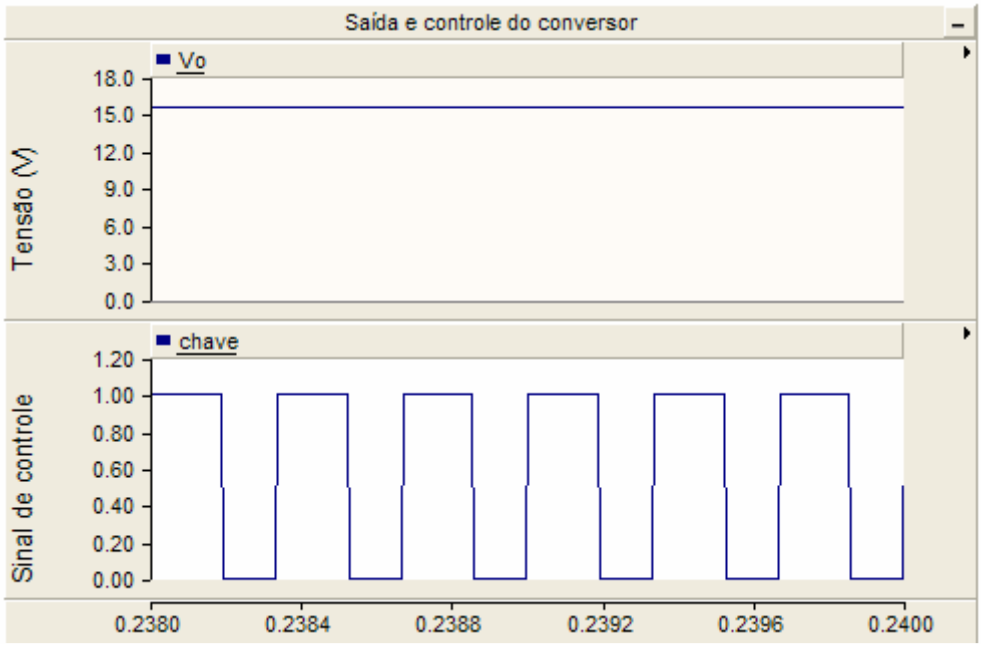


figura 7.18 – Tensão de saída e controle do conversor *buck-boost* de exemplo

O ciclo de trabalho do conversor *buck-boost* é dado pela expressão:

$$DC(\%) = \left(\frac{V_o}{V_o + V_i}\right) \times 100 \cong \left(\frac{15}{15+12}\right) \times 100 = 55{,}55\%$$

7.3.2 – Resultados da homologação

O comportamento do controle das chaves do *chopper* deverá então seguir o apresentado no item anterior, onde nota-se ser o mesmo comportamento do item 7.2. A página de comando do experimento 2 apresenta três opções para o usuário:

- Comando do conversor: ligar ou desligar. Idêntico ao caso anteior, ou seja, ao ligar, o LED verde acende e o microcontrolador começa a produzir o sinal de controle desejado e ao desligar o LED vermelho é acesso e o sinal de controle é fixado em zero, sendo os níveis lógicos 3,3V e 0V.
- Frequência de chaveamento: qualquer valor real, de 30 a 2.000Hz. Determina qual será a frequência de chaveamento desejada, idem ao item 7.2.3.
- Ciclo de trabalho: qualquer valor real entre 0 e 99,999%. Determina qual será o ciclo de trabalho desejado para o conversor, idem ao item 7.2.3. Assim como no experimento 1, o valor máximo não pode ser 100% porque o conversor *buck-boost* também produzirá uma tensão infinita para um ciclo de trabalho de exatamente 100%.

Foram executados testes com este conversor e os seguintes resultados obtidos:

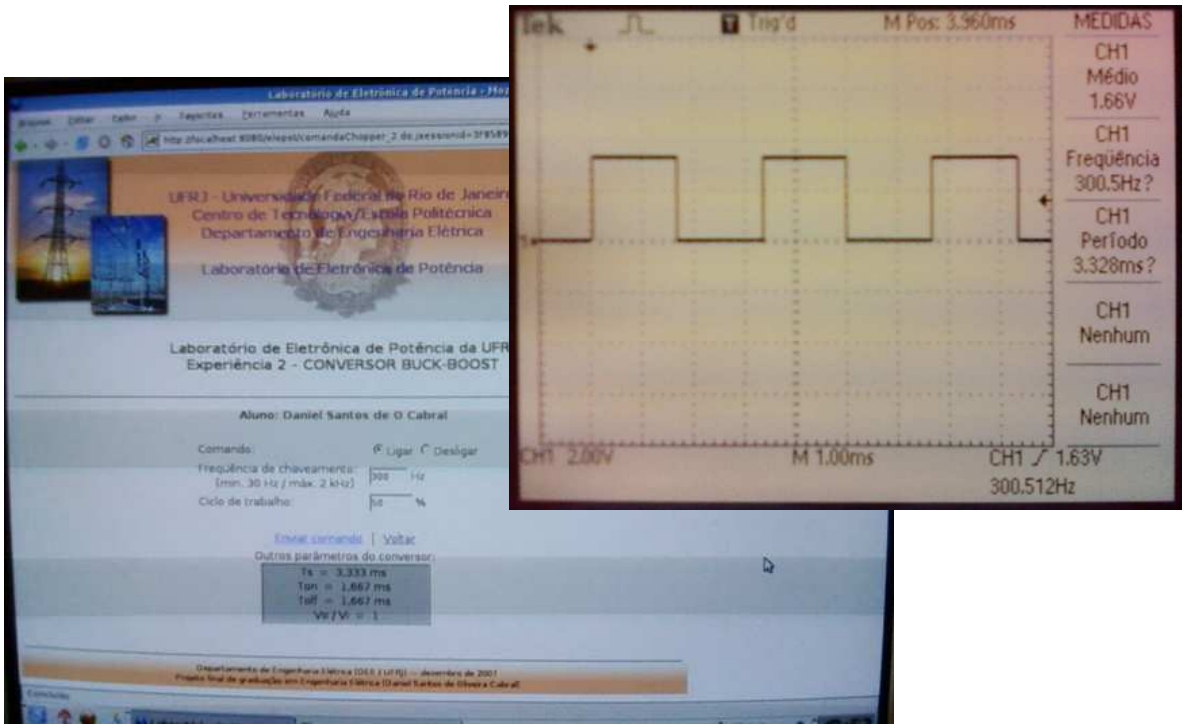


figura 7.19 – Resultados da homologação do experimento 2

Teste	Período (ms)		T_{on} (ms)		T_{off} (ms)	
	Teórico	Real	Teórico	Real	Teórico	Real
1) *Buck-boost*	3,333	3,328	1,667	1,660	1,667	1,660
2) *Buck-boost*	0,556	0,554	0,233	0,232	0,322	0,324

tabela 7.2 – Resultados da homologação do experimento 2

Todos os resultados foram satisfatórios ficando o controle do experimento 2 adequado à sua futura implementação prática no laboratório de eletrônica de potência.

7.4 – Homologação do experimento 3

O experimento de número 3 do laboratório de eletrônica de potência utiliza um conversor inversor (CC-CA) monofásico, na sua configuração de 2 pulsos. O controle deste conversor será feito por meio da comparação constante de um sinal periódico fixo com um sinal senoidal de referência. O resultado é um sinal de controle PWM com frequência variável que aplicado nas 2 chaves (complementares entre si) do circuito de potência produz uma saída também PWM com frequência fundamental igual à frequência de referência, ou seja, a frequência desejada. Para mais detalhes acerca do funcionamento do inversor consultar Mohan [14].

Esta homologação consiste então em analisar os sinais de controle do inversor e compará-los com o obtido experimentalmente. No item 7.4.1 será feita uma breve análise com um circuito de exemplo usado na simulação em PSCAD realizada no laboratório de eletrônica de potência, tornando possível estabelecer alguns parâmetros para esta homologação.

7.4.1 – O inversor

O exemplo a seguir mostra um inversor monofásico usado no experimento 3. Os parâmetros deste inversor são a tensão (V_{out}) e a frequência (f_{out}) de saída desejadas. No exemplo em questão deseja-se uma saída alternada de aproximadamente 9V e com frequência de 40Hz. O circuito de controle atuará então produzindo pulsos de tensão PWM complementares nas chaves que levará o circuito de potência a produzir a saída desejada. Na figura 7.21 somente é mostrado um dos sinais de controle com o intuito de facilitar a visualização. O outro sinal é exatamente o seu complemento.

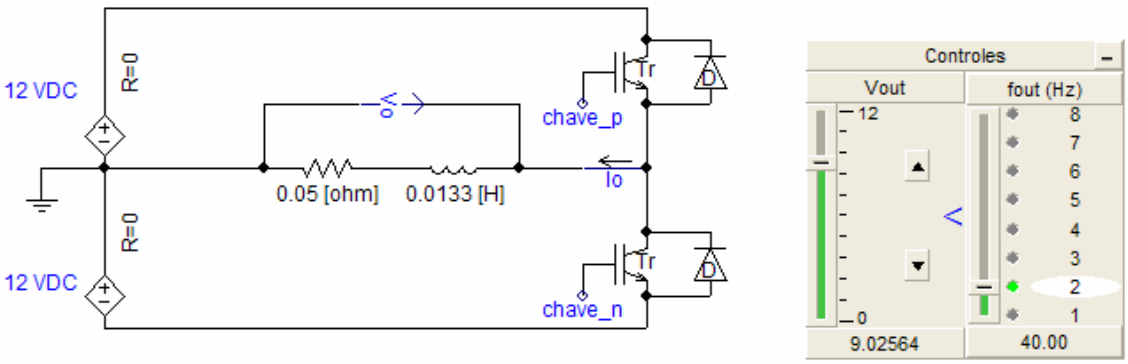


figura 7.20 – Inversor de exemplo

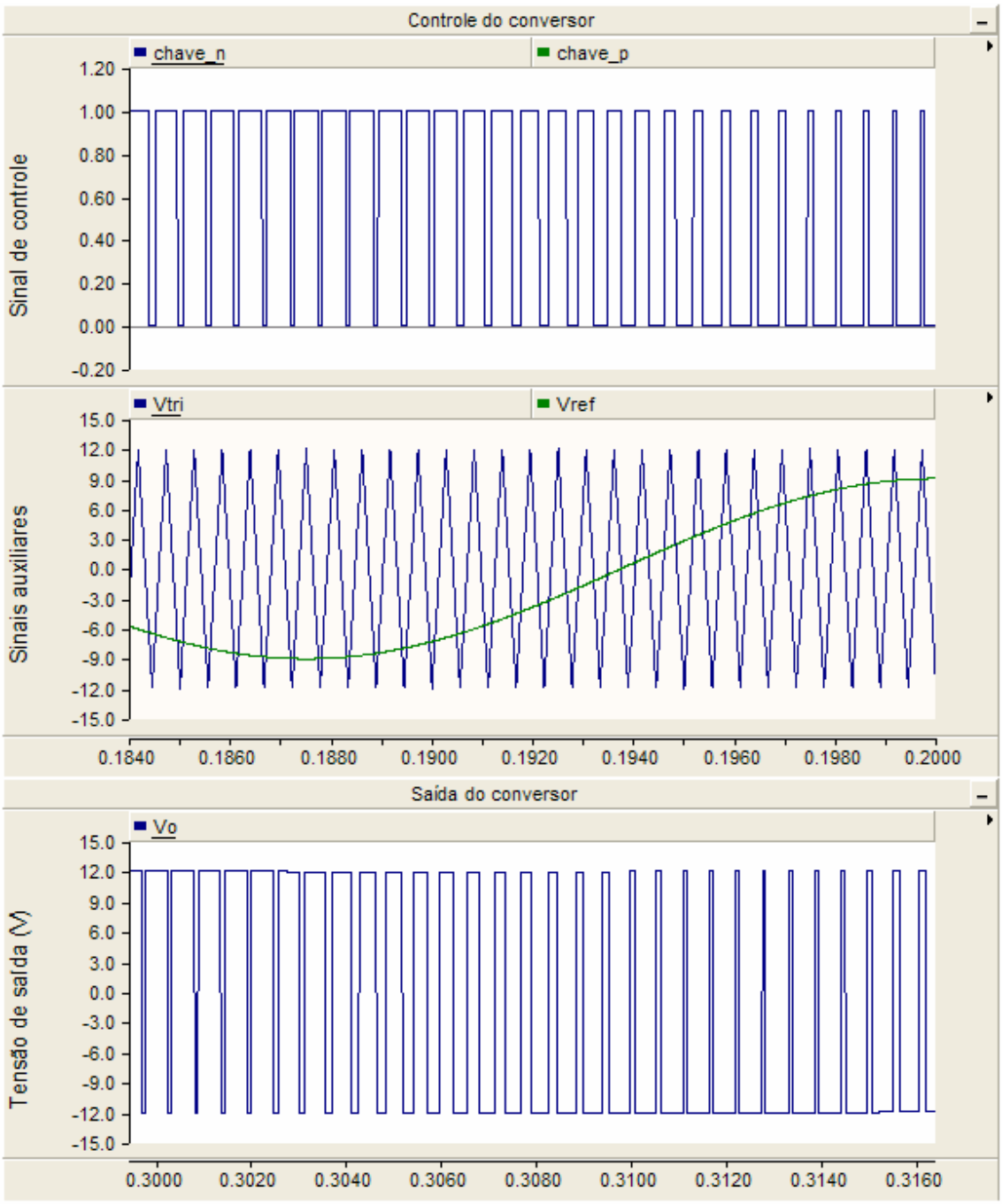


figura 7.21 – Sinais de controle e saída do inversor de exemplo

O sinal periódico triangular deste exemplo possui frequência de 1.800Hz. Para analisar a saída do inversor será necessário um filtro (ou analisador de harmônicos) capaz de extrair a componente fundamental do sinal PWM gerado. No caso do PSCAD isso pode ser feito usando-se o componente FFT, que permite extração de diversas componentes harmônicas do sinal. Para este exemplo, a frequência de base escolhida foi de 40Hz. A figura 7.22 mostra então o sinal de saída do inversor filtrado, senoidal e com frequência aproximadamente igual à frequência desejada.

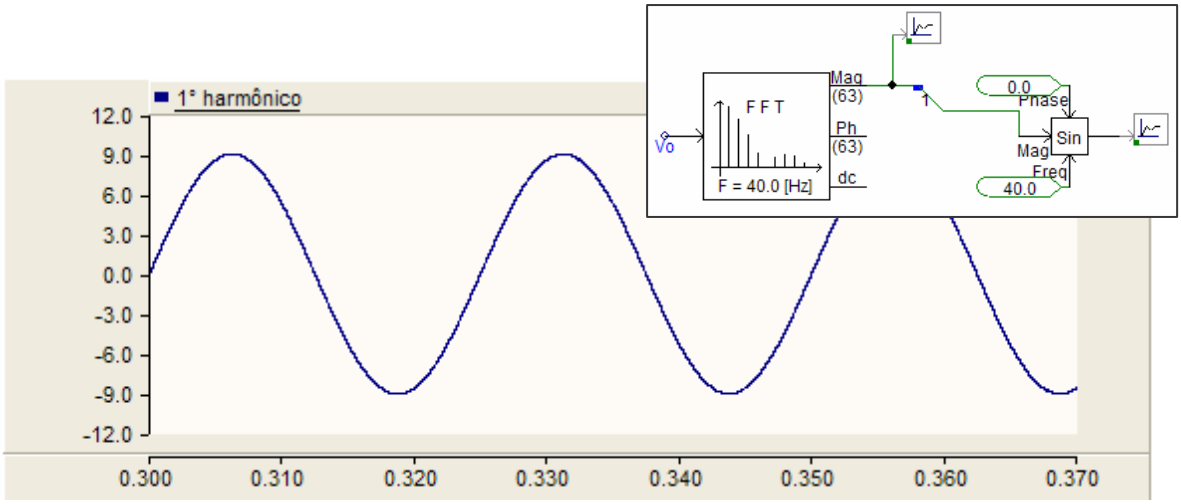


figura 7.22 – Sinal de saída filtrado do inversor de exemplo

Para homologar este conversor será utilizado um filtro RC de 1ª ordem, passa-baixa e com frequência de corte de 75Hz. O produto RC deste filtro deverá ser de:

$$RC = \frac{1}{2 \cdot \pi \cdot 75Hz} = \frac{1}{471{,}24} = 0{,}00212$$

Poderá então ser usado um resistor de 22kΩ e um capacitor de 0,1µF (ou valores próximos). Por se tratar de um filtro simples e composto de componentes discretos é possível que o sinal de saída seja atenuado, porém o propósito desta homologação, que é a análise da forma de onda e a frequência do sinal de saída, será mantido.

7.4.2 – Resultados da homologação

O comportamento do controle das chaves do inversor deverá então seguir o apresentado no item 7.4.1. A página de comando do experimento 3 apresenta quatro opções para o usuário:

- Comando do conversor: ligar ou desligar. Idêntico ao item anterior (7.3).
- Frequência de chaveamento: qualquer valor real, de 360 a 4.000Hz. Determina qual será a frequência de chaveamento desejada. Neste caso, é importante observar os limites impostos. A frequência de chaveamento não pode ser muito baixa pois um bom projeto de inversor exige que $f_{chav} >> f_{ref}$. Sendo assim, no pior caso é possível ter $f_{chav} = 360Hz$ para $f_{ref} = 180Hz$ (o valor máximo para a frequência de referência), uma diferença pequena entre os dois. E o limite superior não pode ser muito elevado porque o algoritmo de controle é executado iterativamente em uma frequência de 158kHz, ou seja, na máxima frequência de chaveamento haverá apenas 39 iterações completas para cada ciclo de chaveamento, um valor bem reduzido.

- Frequência de referência: qualquer valor real entre 0 e 180Hz. Determina qual será o valor da frequência desejada para a tensão de saída do inversor. Normalmente em aplicações práticas se utilizam apenas valores de baixa frequência, próximos por exemplo de 50 ou 60Hz.
- Tensão de referência: qualquer valor real entre 0 e 2pu. Determina qual será o valor da amplitude desejada para a tensão de saída do inversor. É importante lembrar que tensões acima de 1pu levarão o sinal de saída a mudar da forma de onda senoidal para a forma de onda quadrada.

Foram executados testes com este conversor e os seguintes resultados obtidos:

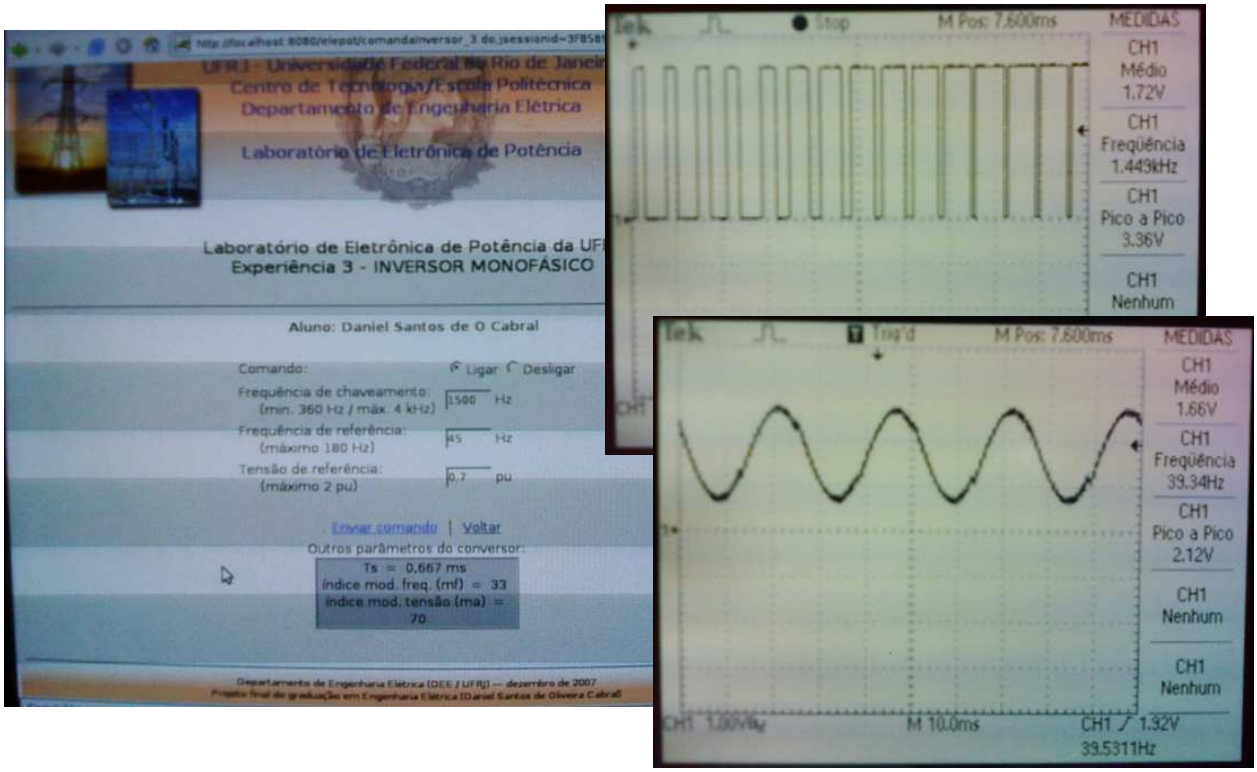


figura 7.23 – Resultados da homologação do experimento 3

Teste	Tensão (pu)		f_{ref} (Hz)	
	Teórico	Real	Teórico	Real
1) Inversor	0,80	0,78	20,0	16,0
2) Inversor	0,50	0,48	55,0	44,5
2) Inversor	1,00	0,95	30,0	24,1

tabela 7.3 – Resultados da homologação do experimento 3

Todos os resultados foram satisfatórios ficando o controle do experimento 3 adequado à sua futura implementação prática no laboratório de eletrônica de potência.

7.5 – Homologação do experimento 4

O experimento de número 4 do laboratório de eletrônica de potência utiliza um conversor retificador (CA-CC) monofásico, controlado, em sua configuração de onda completa com transformador de *tap* central. Esta configuração permite obter o sinal retificado de onda completa utilizando apenas duas chaves de potência. O controle deste conversor é um pouco mais complexo que os demais, pois é o único que depende da interação entre os sinais de potência e controle, que devem ser sincronizados entre si. O controle do retificador irá iniciar uma contagem temporal ou angular, dependendo do algoritmo usado, sempre que o sinal de potência mudar de polaridade disparando um trem de pulsos ao término da contagem. Essa detecção da mudança de polaridade é normalmente feita com uma amostra do sinal de potência e um circuito sincronizador que "avisa" ao controlador que a contagem deve iniciar-se. Essa contagem permite que o disparo das chaves (normalmente são tiristores) ocorra no ângulo de disparo desejado. Teoricamente um único pulso de controle poderia ser aplicado ao tiristor para ele iniciar a condução de corrente, mas em implementações práticas são utilizados alguns outros artifícios, como o trem de pulsos e o *snubber* (para proteção contra disparos acidentais da chave). Para mais detalhes acerca do retificador consultar Mohan [14].

Esta homologação consiste então de analisar os sinais de controle do retificador e compará-los com o obtido experimentalmente, sendo que, neste caso, o sinal de controle analisado será visto sem a sincronização com o circuito de potência. O sistema desenvolvido possui um mecanismo simples de sincronização conforme explicado no item 4.3.5, porém ficará como sugestão para um estudo futuro sua implementação já que a idéia deste trabalho é apenas desenvolver a base de software necessária, e homologar esta rotina necessitaria de uma complexidade muito maior que os demais experimentos.

No item 7.5.1 será feita uma breve análise com um circuito de exemplo usado na simulação em PSCAD realizada no laboratório de eletrônica de potência, tornando possível estabelecer alguns parâmetros para esta homologação.

7.5.1 – O retificador com transformador de *tap* central

O exemplo a seguir mostra um retificador monofásico com transformador de *tap* central, usado no experimento 4. O único parâmetro deste retificador é o seu ângulo de

disparo (alpha). No exemplo em questão a tensão de entrada é de $120V_{ca}$ (valor RMS) e a tensão no secundário do transformador (sem o *tap*) é de $60V_{ca}$ (valor RMS), não exibida na figura 7.24, sendo ainda o ângulo de disparo desejado de aproximadamente 15,4º. O circuito de controle atuará então disparando um pulso no *gate* de cada tiristor (na simulação não é necessário o disparo de diversos pulsos) a cada intervalo de 15,4º contados a partir da mudança de polaridade do sinal de potência do conversor.

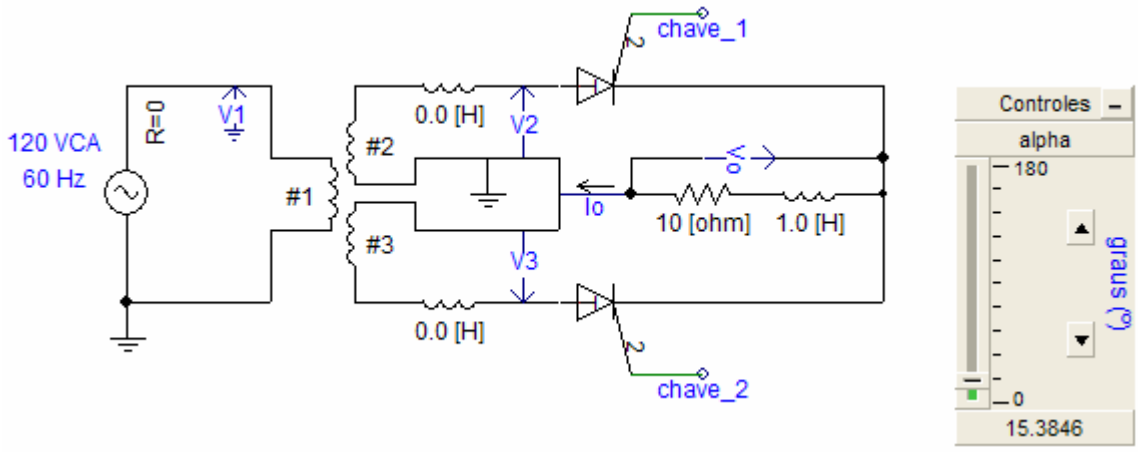


figura 7.24 – Retificador com transformador de *tap* central de exemplo

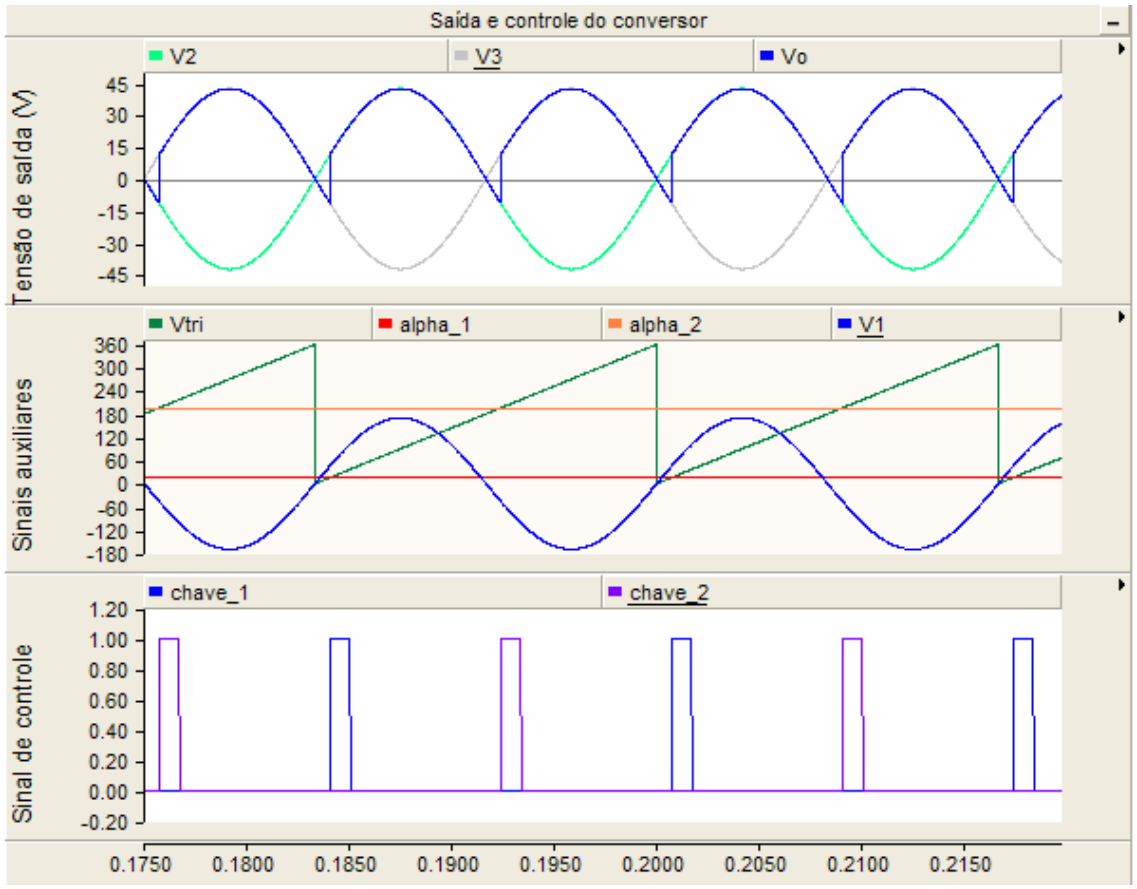


figura 7.25 – Sinais de controle e saída do retificador com transformador de *tap* central de exemplo

7.5.2 – Resultados da homologação

O comportamento do controle das chaves do retificador deverá então seguir o apresentado no item 7.5.1. A página de comando do experimento 4 apresenta três opções para o usuário:

- Comando do conversor: ligar ou desligar. Idêntico ao item anterior (7.4).
- Ângulo de disparo: qualquer valor real, de 0 a 180 graus. Determina qual será o ângulo de disparo desejado.
- Frequência da rede: 50 ou 60Hz. Indica qual a frequência do sinal alternado de potência na entrada do conversor. Por meio deste valor o algoritmo de controle calcula o período correto para a contagem do tempo.

Foram executados testes com este conversor e os seguintes resultados obtidos:

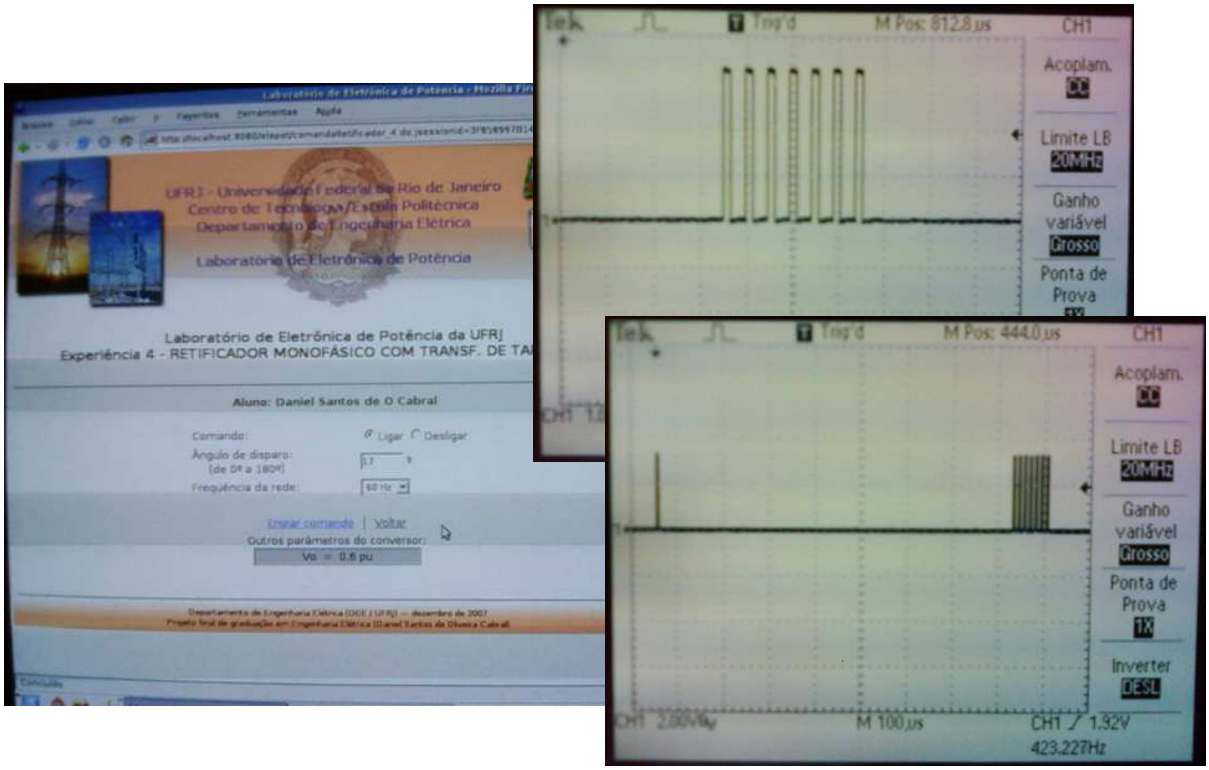


figura 7.26 – Resultados da homologação do experimento 4

Teste	Ângulo de disparo		Quantidade de pulsos
	Teórico	Real	
1) Retificador	15º (694µs)	16,6º (768µs)	7
2) Retificador	58º (2685µs)	63,1º (2920µs)	7

tabela 7.4 – Resultados da homologação do experimento 4

É possível observar nos testes que o controle envia um sinal lógico ON sempre que uma contagem é iniciada (um pulso isolado antes do trem de pulsos). Com este sinal foi possível então medir o ângulo de disparo porém é importante lembrar que em uma implementação prática do sistema o mesmo deverá ser removido pois poderia causar um fechamento indesejado na chave.

Apesar dos resultados mais limitados, esta homologação pode ser considerada satisfatória para os propósitos deste trabalho, ficando o controle do experimento 4, com alguns ajustes, adequado à sua futura implementação prática no laboratório de eletrônica de potência.

7.6 – Homologação do experimento 5

O experimento de número 5 do laboratório de eletrônica de potência utiliza mais uma vez o conversor retificador (CA-CC) monofásico, controlado, porém, desta vez em sua configuração de onda completa com 4 pulsos. Esta configuração permite obter o mesmo sinal do retificador com transformador de *tap* central, porém ele utiliza um simples transformador monofásico e quatro chaves de potência. Uma vantagem desta configuração é que o transformador poderia até mesmo ser dispensado (no laboratório de eletrônica de potência da UFRJ isso não ocorre por motivos de segurança pois deve-se trabalhar somente com tensões extremamente baixas).

O controle deste conversor é idêntico ao controle do experimento anterior, inclusive no que diz respeito à sincronização dos sinais de potência e controle. O único detalhe a destacar é que agora duas chaves são sempre acionadas simultaneamente – o que não afeta o controle do conversor já que o mesmo sinal de acionamento de uma das chaves será compartilhado entre as duas. Para mais detalhes acerca do funcionamento deste outro tipo de retificador consultar Mohan [14].

Para a homologação do experimento 5, seriam executados exatamente os mesmos procedimentos da anterior, sendo assim desnecessária sua realização. A única diferença entre este experimento e o anterior está na configuração do circuito de potência, que não faz parte desta homologação. Assim como os demais experimentos, no item 7.6.1 será feita uma breve análise com um circuito de exemplo do retificador monofásico de onda completa de 4 pulsos usado na simulação em PSCAD realizada no laboratório de eletrônica de potência.

7.6.1 – O retificador de onda completa de 4 pulsos

O exemplo a seguir mostra um retificador monofásico de onda completa de 4 pulsos, usado no experimento 5. Seu único parâmetro é o ângulo de disparo (alpha), similar ao experimento 4. Neste exemplo a tensão de entrada é de $120V_{ca}$ (RMS) e a tensão no secundário do transformador é de $30V_{ca}$ (RMS), não exibida na figura 7.27. O ângulo de disparo desejado é de aproximadamente 15,4º.

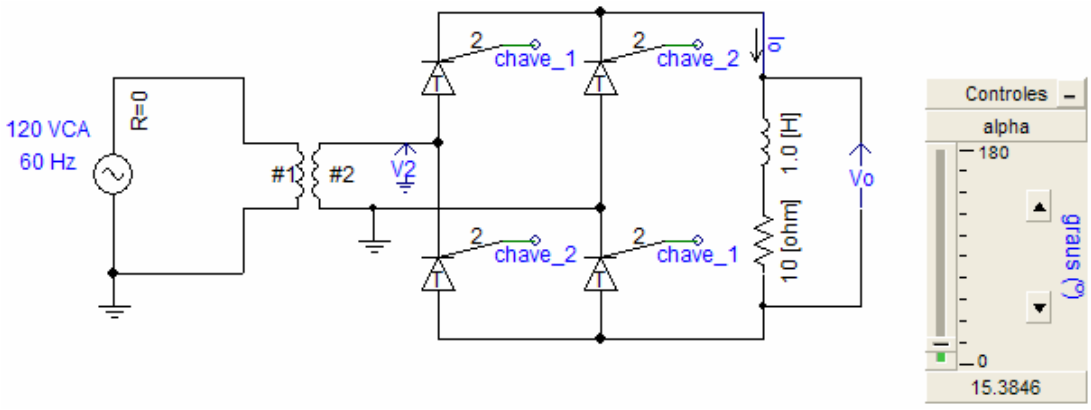


figura 7.27 – Retificador de onda completa de 4 pulsos de exemplo

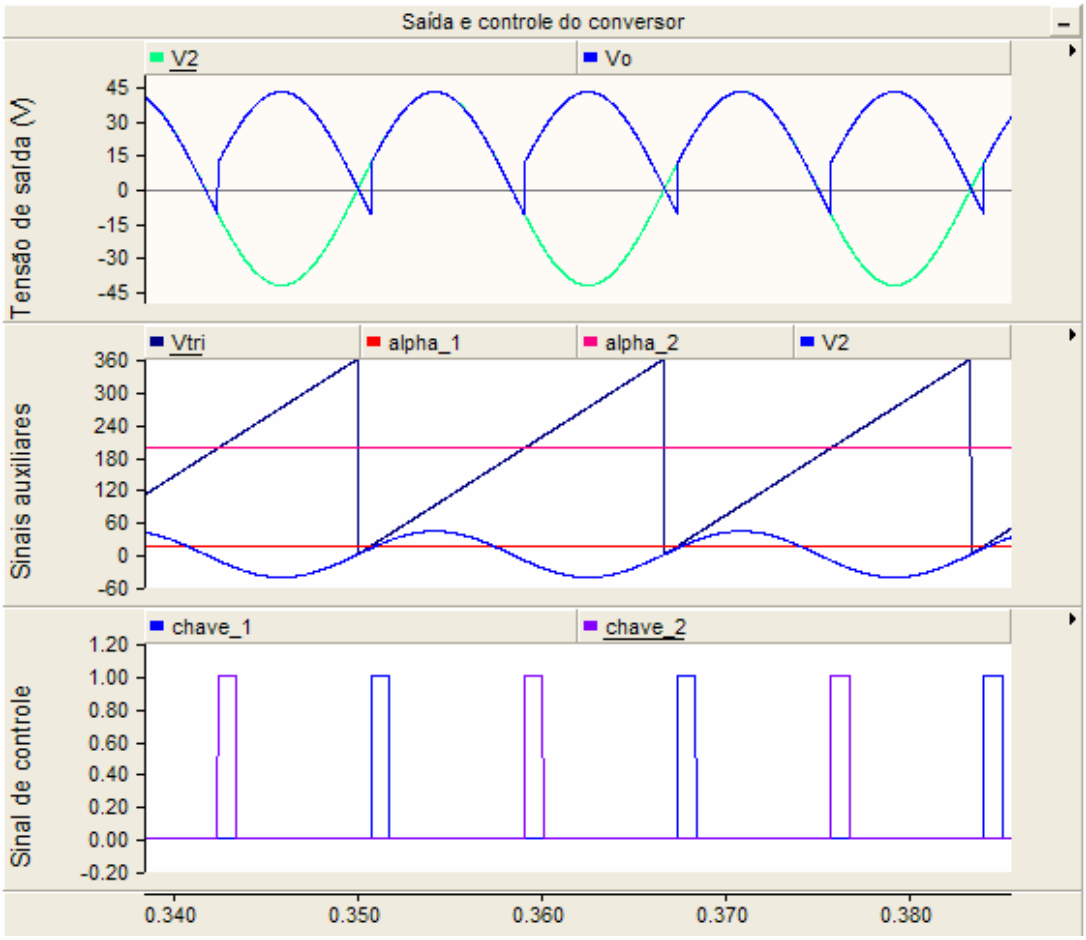


figura 7.28 – Sinais de controle e saída do retificador de onda completa de 4 pulsos de exemplo

8 – Conclusão

O maior desafio deste trabalho foi sem dúvida o uso de um microprocessador dedicado para funcionar como principal elemento de controle do sistema. Atualmente quase todos os dispositivos eletro-eletrônicos são também microprocessados, porém, ainda há uma extrema deficiência de documentação sobre esses dispositivos e poucos dominam este assunto. Os microprocessadores e microcontroladores estarão cada dia mais presentes em nossas vidas e conhecer seu funcionamento e características é fundamental. Uma notável evolução destes dispositivos, que vem permitindo aplicações cada vez mais complexas, foi o seu exponencial aumento da velocidade e capacidade de processamento e memória, além do surgimento de muitos compiladores para linguagens de alto nível como ADA, C ou C++, e de ambientes de programação integrados (IDE) extremamente poderosos, ricos em recursos e com grande interatividade com os microcomputadores e suas interfaces (serial, USB etc.).

Além de servir como vasto material de estudo, este projeto teve como objetivo auxiliar o corpo docente da Universidade Federal do Rio de Janeiro na reestruturação do seu laboratório de eletrônica de potência, cuja principal ferramenta, a bancada didática de eletrônica de potência, encontra-se em necessidade de modernização. Este objetivo foi integralmente cumprido, sendo entregue à UFRJ uma excelente e completa base de software para a modernização deste fundamental equipamento do laboratório.

Um aspecto importante e bem sucedido a considerar neste projeto foi a intensa utilização de software livre e de código aberto, que é uma tendência mundial a ser seguida. Utilizar sistemas de software livres e de código aberto já é um padrão no meio acadêmico e em breve estará também atingindo a escala industrial. Desde o sistema operacional Linux até todas as bibliotecas e APIs Java utilizadas no projeto, optou-se por alternativas abertas.

Uma das características mais marcantes deste projeto foi o grande intercâmbio de informações entre ciências diferentes. Com o avanço cada vez maior dos sistemas de informação e telecomunicação, os sistemas elétricos precisam cada vez mais estar preparados para trabalhar em conjunto com sistemas automação, comunicação de dados, controle e monitoramento remotos, sendo necessário então que o engenheiro eletricista possua sólidos conhecimentos de TI (principalmente análise e programação, bancos de dados, arquitetura e redes de computadores), telecomunicações e sistemas de controle.

Esta miscelânea de assuntos exemplifica como atualmente soluções complexas de sistemas (elétricos ou não) dependem cada vez mais de um conhecimento abrangente em diversas ciências. Este trabalho foi um imenso desafio, uma vez que envolveu temas novos e complexos e muito diversificados entre si. Isso permitiu fornecer uma visão muito interessante de como um único projeto pode se tornar uma grande fonte de conhecimento.

Com os resultados obtidos após a etapa de homologação, o sistema desenvolvido fica então apto a ser implementado no laboratório de eletrônica de potência. Outros usos poderão ainda ser dados ao material produzido, desde seu aproveitamento didático como fonte de consulta até implementações em outros sistemas também microprocessados ou similares, em outros laboratórios ou até mesmo em aplicações industriais. Conforme citado constantemente ao longo do texto, há ainda a possibilidade de diversas melhorias futuras para este sistema, das quais se destacam:

i) Acréscimo de recursos no módulo administrativo como, por exemplo, novos cadastros e relatórios;

ii) Aumento dos limites de escolha dos usuários na interface de controle dos conversores, de acordo com a capacidade do microcontrolador;

iii) Diminuição do período de iteração dos algoritmos do inversor e retificador por meio da aferição mais detalhada dos tempos de processamento dos respectivos algoritmos, tornando assim o controle mais preciso;

iv) Implementação, por software, de um PLL capaz de sincronizar o sinal de potência do retificador com seu sinal de controle. Esta implementação permitirá dispensar o uso do circuito analógico de sincronização, necessário no retificador;

v) Alteração no protocolo ESCP a fim de torná-lo bidirecional e permitir que a aplicação *web* também receba dados da bancada didática, como, por exemplo, a saída gráfica do osciloscópio. Para este item deve-se lembrar que será necessária a utilização de velocidade superior aos atuais 4.800baud na comunicação serial entre o microcontrolador e o microcomputador PC.

Referências Bibliográficas

[1] Rolim, L.G.B.; ***Laboratório Modular de Eletrônica de Potência***. Tese, COPPE/UFRJ, 1993.

[2] Fernandes, R.M.; ***Laboratório Remoto de Eletrônica de Potência***. Projeto de graduação, DEE/UFRJ, 2002.

[3] Rolim, L.G.B., Stephan, R.M., Suemitsu, W., Aredes, M.; ***Laboratório de Eletrônica de Potência***. DEE/UFRJ, 2004. http://www.dee.ufrj.br/elepot/labepot3.pdf

[4] Rolim, L.G.B., Stephan, R.M., Suemitsu, W., Aredes, M.; ***Bancada Didática de Eletrônica de Potência – Manual de utilização***. DEE/UFRJ, 2004. http://www.dee.ufrj.br/elepot/usmanual.pdf

[5] Morimoto, C. E.; ***Kurumin Linux – Desvendando seus segredos***. Book Express, 2003.

[6] Morimoto, C.E.; ***Entendendo e Dominando o Linux***. *7ª ed.* GDH Press, 2004.

[7] Zelenovsky, R., Mendonça, A.; ***PC: Um Guia Prático de Hardware e Interfaceamento***. *2ª ed.* MZ Editora, 1999.

[8] Rolim, L.G.B.; ***Aplicações de Microprocessadores em Eletrônica de Potência***. Notas de aula do curso de microprocessadores. DEE/UFRJ, 2006.

[9] Freescale Semiconductor; ***56F8013 Technical Data Sheet***. *Rev.3*. Set/2005.

[10] Freescale Semiconductor; ***56F8000 Peripheral Reference Manual***. *Ver.1*. Abr/2005.

[11] Freescale Semiconductor; ***56F8013 Demonstration Board User Guide***. *Rev.0*. Mar/2005.

[12] Grossmann, C.A.K.; ***O Caminho das Pedras para o Linux***. Book Express, 1998.

[13] Montezano, B., Cabral, D., Mendes, L. ***Desenvolvimento de um protótipo simplificado de injeção eletrônica EFI (monoponto) para motores a gasolina usando o microcontrolador 56F8013 da Freescale***. UFRJ, 2006. http://cabral.ifastnet.com/graduacao/microprocessadores.pdf

[14] Mohan, N., Undeland, T.M., Robbins, W.P.; ***Power Eletronics: Converters, Applications and Design***. *3rd ed*. John Wiley & Sons, 2003.

[15] Deitel, H.M, Deitel, P.J.; ***Java – Como Programar***. *6ª ed.* Pearson Education do Brasil, 2005.

[16] Husted, T. et al.; ***Struts in Action: building web applications with the leading Java framework***. Manning Publications Co., 2003.

[17] Curso de microprocessadores (DEE/UFRJ). http://www.dee.ufrj.br/microproc/

[18] Sun Microsystems. http://java.sun.com/

[19] Kurumin Linux. http://www.guiadohardware.net/kurumin/

[20] Wikipédia. http://pt.wikipedia.org/

[21] Freescale Semiconductor. http://www.freescale.com/

[22] Tutorial do Tomcat. http://www.mhavila.com.br/topicos/java/tomcat.html

[23] VMware. http://www.vmware.com/

[24] MySQL. http://www.mysql.org/

[25] RXTX. http://www.rxtx.org/

[26] Struts. http://struts.apache.org/

[27] Javafree. http://www.javafree.org/

Apêndice A

Listagem do *script* do banco de dados

elepot-install.sql

```
create database ELEPOT;

grant all privileges on ELEPOT.* to 'elepot_web'@'localhost' identified by 'dfr457g'
with grant option;

use ELEPOT;

create table CONFIGURACAO
(
    CFG_SIST_OPER       VARCHAR(10)           not null,
    CFG_PORTA_SERIAL    VARCHAR(50)           not null,
    CFG_TEMPO_SESSAO    INTEGER               not null
) TYPE=InnoDB;

create table EXPERIMENTO
(
    EXP_ID              INTEGER               not null AUTO_INCREMENT,
    EXP_NUMERO          INTEGER               not null,
    EXP_NOME            VARCHAR(50)           not null,
    EXP_OBJETIVO        VARCHAR(150)          not null,
    EXP_DESCRICAO       VARCHAR(1000)                 ,
    EXP_INSTRUCOES      VARCHAR(2000)                 ,
    primary key (EXP_ID)
) TYPE=InnoDB;

create table USUARIO
(
    USU_ID              INTEGER               not null AUTO_INCREMENT,
    USU_LOGIN           VARCHAR(10)           not null,
    USU_SENHA           INTEGER               not null,
    USU_NOME            VARCHAR(50)           not null,
    USU_IDENTIFICACAO   VARCHAR(20)                   ,
    USU_SN_ATIVO        CHAR(1)               not null,
    primary key (USU_ID)
) TYPE=InnoDB;

create unique index USU_LOGIN_AK on USUARIO (USU_LOGIN asc);

create table PRATICA
(
    PRA_ID              INTEGER               not null AUTO_INCREMENT,
    PRA_EXP_ID          INTEGER               not null,
    PRA_USU_ID          INTEGER               not null,
    PRA_SN_USO          CHAR(1)               not null,
    PRA_DATA            DATE                  not null,
    primary key (PRA_ID),
    foreign key  (PRA_EXP_ID)
       references EXPERIMENTO (EXP_ID),
    foreign key  (PRA_USU_ID)
       references USUARIO (USU_ID)
) TYPE=InnoDB;

create index RF_PRA_EXP_FK on EXPERIMENTO (EXP_ID asc);

create index RF_PRA_USU_FK on USUARIO (USU_ID asc);

insert into CONFIGURACAO (CFG_SIST_OPER, CFG_PORTA_SERIAL, CFG_TEMPO_SESSAO)
values ('WINDOWS', 'COM1', '20');

insert into USUARIO (USU_LOGIN, USU_SENHA, USU_NOME, USU_IDENTIFICACAO, USU_SN_ATIVO)
values ('PROFESSOR', -1008871825, 'Professor', NULL, 'S');
```

```
insert into EXPERIMENTO (EXP_NUMERO, EXP_NOME, EXP_OBJETIVO, EXP_DESCRICAO,
EXP_INSTRUCOES)
values (1, 'CONVERSORES CC-CC', 'Estudar e montar as configurações mais usuais de
conversores CC-CC.', '', '<B><U>MATERIAL NECESSÁRIO</U></B><BR>-Bancada de eletrônica de
potência<BR>-Fonte de alimentação CC<BR>-Motor CC<BR>-Indutor de 13,3 mH<BR>-
Capacitância de 220 uF e de 4000 uF<BR>-Resistor de 25,8 ohms<BR>-Osciloscópio com
memória<BR><BR><B><U>MONTAGEM</U></B><BR>1) Monte o circuito "buck" com os componentes
disponíveis no laboratório e meça a tensão de saída Vo.<BR><CENTER><IMG
SRC="http://localhost:8080/elepot/img/exp1-fig1.jpg" ALT="fig.1"></CENTER><BR>2)
Introduza agora um indutor em série com a carga resistiva. Meça a tensão de saída e a
corrente de carga. Aumente o valor de Toff e entre no modo de condução descontínuo de
corrente.<BR>3) Inclua um filtro como sugere a figura abaixo. Observe a tensão de
saída.<BR><CENTER><IMG SRC="http://localhost:8080/elepot/img/exp1-fig2.jpg"
ALT="fig.2"></CENTER><BR>4) Comente e explique seus resultados.<BR>5) Monte um conversor
"boost".<BR><CENTER><IMG SRC="http://localhost:8080/elepot/img/exp1-fig3.jpg"
ALT="fig.3"></CENTER><BR>6) Alimente o motor CC do laboratório. Coloque em série com a
armadura um indutor de 13,3 mH. Altere Ton e Toff para observar os regimes de condução
contínua e descontínua de corrente.');

insert into EXPERIMENTO (EXP_NUMERO, EXP_NOME, EXP_OBJETIVO, EXP_DESCRICAO,
EXP_INSTRUCOES)
values (2, 'CONVERSOR BUCK-BOOST', 'Estudar um regulador buck-boost.', '',
'<B><U>MATERIAL NECESSÁRIO</U></B><BR>-Bancada de eletrônica de potência<BR>-Fonte de
alimentação CC<BR>-Indutor de 12,5 mH<BR>-Capacitor de 100 uF<BR>-Resistor de 258
ohms<BR>-Diodo<BR>-Osciloscópio<BR><BR><B><U>MONTAGEM</U></B><BR>1) Monte o circuito da
figura abaixo para uma carga puramente resistiva. Varie a frequência de chaveamento e o
ciclo de trabalho.<BR><CENTER><IMG SRC="http://localhost:8080/elepot/img/exp2-fig1.jpg"
ALT="fig.1"></CENTER><BR>2) Obtenha a forma de onda da corrente no indutor, e da tensão
e corrente na carga para as frequências e ciclos de trabalhos do item anterior.<BR>3)
Observe o "ripple" na tensão de saída.<BR>4) Observe para quais ciclos de trabalho o
conversor funciona no modo descontínuo e contínuo, observando o limite entre eles.<BR>5)
Observe em que condições o conversor funciona como buck ou boost.');

insert into EXPERIMENTO (EXP_NUMERO, EXP_NOME, EXP_OBJETIVO, EXP_DESCRICAO,
EXP_INSTRUCOES)
values (3, 'INVERSOR MONOFÁSICO', 'Estudar o inversor monofásico como preparação para o
entendimento de inversores trifásicos e aplicações em acionamento de motores CA.', '',
'<B><U>MATERIAL NECESSÁRIO</U></B><BR>-Bancada de eletrônica de potência<BR>-Fonte
simétrica de alimentação CC<BR>-Motor CC<BR>-3 indutores de 13,3 mH<BR>-Osciloscópio com
memória<BR><BR><B><U>MONTAGEM</U></B><BR>1) Monte o circuito inversor sugerido na figura
abaixo.<BR><CENTER><IMG SRC="http://localhost:8080/elepot/img/exp3-fig1.jpg"
ALT="fig.1"></CENTER><BR>2) Faça medidas da tensão de saída Vo para cargas com
diferentes parcelas indutivas. Determine os períodos de condução do diodo e do
transistor.<BR> Dica: coloque 3 indutores em série e vá curto-circuitando aos
poucos.<BR>3) Comente e explique seus resultados. Faça uma análise harmônica dos sinais
de tensão e corrente na carga.<BR>4) Seria possível alimentar um motor CA monofásico,
com capacitor auxiliar de partida e chave centrífuga?<BR>5) Justifique a quebra que
ocorre na forma de onda da corrente.<BR> Dica: lembre-se que o indutor usado no
laboratório tem núcleo de ferro.');

insert into EXPERIMENTO (EXP_NUMERO, EXP_NOME, EXP_OBJETIVO, EXP_DESCRICAO,
EXP_INSTRUCOES)
values (4, 'RETIFICADOR MONOFÁSICO COM TRANSF. DE TAP CENTRAL', 'Estudar o circuito
retificador monofásico.', '', '<B><U>MATERIAL NECESSÁRIO</U></B><BR>-Bancada de
eletrônica de potência<BR>-Transformador de tap central (120:30 V)<BR>-Motor CC<BR>-2
indutores de 13,3 mH (linha CA)<BR>-1 indutor de 1 H (carga)<BR>-Resistor de 25,8 ohms
ou 10 ohms (carga)<BR>-Osciloscópio com memória<BR><BR><B><U>MONTAGEM</U></B><BR>1)
Monte, com o auxílio da matriz de chaveamento disponível no laboratório, um retificador
monofásico de onda completa a tiristores, com transformador de tap
central.<BR><CENTER><IMG SRC="http://localhost:8080/elepot/img/exp4-fig1.jpg"
ALT="fig.1"></CENTER><BR>2) Verifique a necessidade do circuito de sincronização,
invertendo, por exemplo, os bornes de alimentação ou tomando como referência outra fase
da rede elétrica.<BR>3) Para uma carga puramente resistiva, varie o ângulo de disparo
dos tiristores. Meça a tensão e a corrente na carga.<BR>4) Repita o item anterior para
cargas RL, com diferentes valores de indutâncias. Constate a existncia dos modos de
condução de corrente: contínuo e descontínuo.<BR>5) Coloque um indutor em série com a
linha de alimentação CA. Constate o aparecimento de "notchs" de tensão tanto do lado CA
quant do lado CC. Verifique que o "notch" só ocorre em condução contínua de
corrente.<BR>6) Através de medições adequadas, verifique se as fórmulas aprendidas no
curso teórico representam razoavelmente a situação criada no laboratório.');
```

```
insert into EXPERIMENTO (EXP_NUMERO, EXP_NOME, EXP_OBJETIVO, EXP_DESCRICAO,
EXP_INSTRUCOES)
values (5, 'RETIFICADOR MONOFÁSICO DE ONDA COMPLETA', 'Estudar o retificador monofásico
de onda completa.', '', '<B><U>MATERIAL
NECESSÁRIO</U></B><BR><BR><BR><B><U>MONTAGEM</U></B><BR>1) Monte o circuito da figura
abaixo.<BR><CENTER><IMG SRC="http://localhost:8080/elepot/img/exp5-fig1.jpg"
ALT="fig.1"></CENTER><BR>2) Com uma carga puramente resistiva, obtenha a forma de onda
da tensão na carga para vários valores de ângulo de disparo.<BR>3) Com uma carga RL,
obtenha as formas de onda de tensão e de corrente na carga para vários valores de ângulo
de disparo.<BR>4) Observe o "notch".<BR>5) Compare os resultados desta experiência com
os da experiência 4.');
```

*** IMPORTANTE ***

Para que as figuras do roteiro de experimentos do laboratório possam ser visualizadas de qualquer estação de trabalho que tenha acesso ao sistema será necessário substituir o termo *localhost* pelo nome do servidor ELEPOT na rede ou seu endereço IP.

Apêndice B

Listagem do código-fonte do software embarcado

ELEPOT.h

```
/** ####################################################################
**     ELEPOT SERIAL COMUNICATION PROTOCOL (ESCP)
**  ####################################################################*/

struct ESCP {
       long parameter[4];
       byte converter;
       byte command;
       byte crc;
       long reserved;
};

/** ####################################################################
**     LEDS AND I/O PORTS CONTROL
**  ####################################################################*/

/* port A base address (GPIOA) */
#define GPIOA_BASE  0xF100
/* DATA, DDIR and PEREN registers address */
#define GPIOA_DATA   (*(volatile unsigned int *)(GPIOA_BASE + 0x0001))
#define GPIOA_DDIR   (*(volatile unsigned int *)(GPIOA_BASE + 0x0002))
#define GPIOA_PEREN  (*(volatile unsigned int *)(GPIOA_BASE + 0x0003))
/* other registers */
#define GPIOA_PUPEN  (*(volatile unsigned int *)(GPIOA_BASE + 0x0000))
#define GPIOA_IASSRT (*(volatile unsigned int *)(GPIOA_BASE + 0x0004))
#define GPIOA_IEN    (*(volatile unsigned int *)(GPIOA_BASE + 0x0005))
#define GPIOA_IENPOL (*(volatile unsigned int *)(GPIOA_BASE + 0x0006))
#define GPIOA_IPEND  (*(volatile unsigned int *)(GPIOA_BASE + 0x0007))
#define GPIOA_IEDGE  (*(volatile unsigned int *)(GPIOA_BASE + 0x0008))
#define GPIOA_PPOUTM (*(volatile unsigned int *)(GPIOA_BASE + 0x0009))
#define GPIOA_RDATA  (*(volatile unsigned int *)(GPIOA_BASE + 0x000A))
#define GPIOA_DRIVE  (*(volatile unsigned int *)(GPIOA_BASE + 0x000B))

/* masks */
#define SET_BITS_0_TO_5   0x003F
#define CLR_BITS_0_TO_5  ~0x003F

#define RESET_BITS_DATA   0x00C1
#define RESET_BITS_DDIR   0x003F
#define RESET_BITS_PEREN  0x0010

#define SET_BIT_0  0x0001
#define SET_BIT_1  0x0002
#define SET_BIT_2  0x0004
#define SET_BIT_3  0x0008
#define SET_BIT_4  0x0010
#define SET_BIT_5  0x0020
#define SET_BIT_6  0x0040
#define SET_BIT_7  0x0080

#define CLR_BIT_0 ~0x0001
#define CLR_BIT_1 ~0x0002
#define CLR_BIT_2 ~0x0004
#define CLR_BIT_3 ~0x0008
#define CLR_BIT_4 ~0x0010
#define CLR_BIT_5 ~0x0020
#define CLR_BIT_6 ~0x0040
#define CLR_BIT_7 ~0x0080
```

ELEPOT.c

```
/** ####################################################################
**     Project   : ELEPOT - UFRJ
**     Version   : 1.0
**     Author    : Daniel Santos de O Cabral
**  ####################################################################*/
```

```
/* MODULE ELEPOT */
/* Including used modules for compiling procedure */
#include "Cpu.h"
#include "Events.h"
#include "Chopper.h"
#include "Inverter.h"
#include "Rectifier.h"
#include "AS.h"
#include "Btn.h"
#include "Inhr1.h"
#include "PPG_PWM.h"
#include "I_Timer.h"
#include "R_Timer.h"
#include "ADConv.h"
#include "ELEPOT.h"

/* Include shared modules, which are used for whole project */
#include "PE_Types.h"
#include "PE_Error.h"
#include "PE_Const.h"
#include "IO_Map.h"

/* global variables */
struct ESCP mainPack;

/* methods */
void initProgram();
void configureGPIOA(bool default_configuration);
void setESCP(struct ESCP pack);
void clearESCP(struct ESCP pack);
bool checkCRC(struct ESCP pack);
void processESCP();
void setChopper(byte command, long fsw, long dcNormalized);
void setInverter(byte command, long fsw, long frf, long vrf);
void setRectifier(byte command, long alpha, long fi);

/* program */
void main(void)
{
       /*** Processor Expert internal initialization. DON'T REMOVE THIS CODE!!! ***/
       PE_low_level_init();
       /*** End of Processor Expert internal initialization.                    ***/

       initProgram();

       for(;;) {}
}

void initProgram()
{
       clearESCP(mainPack);
       GPIOA_DATA &= CLR_BITS_0_TO_5; /* turn OFF all the leds */
       GPIOA_DATA |= SET_BIT_0;       /* turn ON the red led (all converters OFF) */

       configureGPIOA(FALSE);
       setInverter(0, 0L, 0L, 0L); /* turn OFF all the leds */
       setRectifier(0, 0L, 0L);    /* turn OFF all the leds */

       configureGPIOA(TRUE);
       setChopper(0, 0L, 0L);      /* turn OFF all the leds */
}

void configureGPIOA(bool default_configuration)
{
       if (!default_configuration)
       {
              GPIOA_PEREN &= CLR_BITS_0_TO_5; /* active the peripheral register */
              GPIOA_DDIR |= SET_BITS_0_TO_5;  /* define the communication direction */
       }
       else
       {
              GPIOA_PEREN = RESET_BITS_PEREN; /* reset the peripheral register */
              GPIOA_DDIR = RESET_BITS_DDIR;   /* reset the communication direction */
       }
}
```

```
void setESCP(struct ESCP pack)
{
       mainPack = pack;
       processESCP();
}

void clearESCP(struct ESCP pack)
{
       int i;

       pack.converter = 0;
       pack.command = 0;
       pack.crc = 0;
       for (i=0; i <= 3; i++)
              pack.parameter[i] = 0;
}

bool checkCRC(struct ESCP pack)
{
       long checksum = 0;
       byte crc;
       int i;

       checksum += (long) (pack.command) + (long) (pack.converter);
       for (i = 0; i < 4; i++)
              checksum += pack.parameter[i];
       crc = (byte) (checksum & 0x000000000000000000000000FFFFFFFF);

       return (crc == pack.crc);
}

void processESCP()
{
       switch (mainPack.converter)
       {
              case 1 :
              {
                     configureGPIOA(TRUE);
                     setChopper(mainPack.command, mainPack.parameter[0],
mainPack.parameter[1]);
                     break;
              }
              case 2 :
              {
                     configureGPIOA(FALSE);
                     setInverter(mainPack.command, mainPack.parameter[0],
mainPack.parameter[1], mainPack.parameter[2]);
                     break;
              }
              case 4 :
              {
                     configureGPIOA(FALSE);
                     setRectifier(mainPack.command, mainPack.parameter[0],
mainPack.parameter[1]);
                     break;
              }
       }
}

/** ###################################################################
**     End-of-file
**  ###################################################################*/
```

Events.h

```
/** ###################################################################
**     Project   : ELEPOT - UFRJ
**     Author    : Daniel Santos de O Cabral
**  ###################################################################*/

#ifndef __Events_H
#define __Events_H
/* MODULE Events */

#include "PE_Types.h"
#include "PE_Error.h"
```

```
#include "PE_Const.h"
#include "IO_Map.h"
#include "PE_Timer.h"
#include "AS.h"
#include "Btn.h"
#include "Inhr1.h"
#include "PPG_PWM.h"
#include "I_Timer.h"
#include "R_Timer.h"
#include "ADConv.h"

void AS_OnError(void);
void AS_OnRxChar(void);
void AS_OnFullRxBuf(void);

void Btn_OnButton(void);
/*
** ===================================================================
**     Event       :  Btn_OnButton (module Events)
**
**     From bean   :  Btn [Button]
**     Description :
**         This event is called when the button is pressed.
**         If button inactivity feature (advanced view) is enabled,
**         then the next OnButton event is not generated during dead
**         time.
**     Parameters  : None
**     Returns     : Nothing
** ===================================================================
*/

/* END Events */
#endif /* __Events_H*/

/** ###################################################################
**     End-of-file
**  ###################################################################*/
```

Events.c

```
/** ###################################################################
**     Project   : ELEPOT - UFRJ
**     Author    : Daniel Santos de O Cabral
**  ###################################################################*/

/* MODULE Events */

#include "Cpu.h"
#include "Events.h"
#include "Chopper.h"
#include "Inverter.h"
#include "Rectifier.h"
#include "ELEPOT.h"

/*
** ===================================================================
**     Event       :  AS_OnError (module Events)
**
**     From bean   :  AS [AsynchroSerial]
**     Description :
**         This event is called when a channel error (not the error
**         returned by a given method) occurs. The errors can be
**         read using <GetError> method.
**     Parameters  : None
**     Returns     : Nothing
** ===================================================================
*/
#pragma interrupt called /* Comment this line if the appropriate 'Interrupt preserve
registers' property */
                         /* is set to 'yes' (#pragma interrupt saveall is generated
before the ISR)      */
void AS_OnError(void)
{
}
```

```
/*
** ===================================================================
**     Event       :  AS_OnRxChar (module Events)
**
**     From bean   :  AS [AsynchroSerial]
**     Description :
**         This event is called after a correct character is
**         received.
**         DMA mode:
**         If DMA controller is available on the selected CPU and
**         the receiver is configured to use DMA controller then
**         this event is disabled. Only OnFullRxBuf method can be
**         used in DMA mode.
**     Parameters  : None
**     Returns     : Nothing
** ===================================================================
*/
#pragma interrupt called /* Comment this line if the appropriate 'Interrupt preserve
registers' property */
                         /* is set to 'yes' (#pragma interrupt saveall is generated
before the ISR)      */
void AS_OnRxChar(void)
{
}

/*
** ===================================================================
**     Event       :  AS_OnFullRxBuf (module Events)
**
**     From bean   :  AS [AsynchroSerial]
**     Description :
**         This event is called when the input buffer is full.
**     Parameters  : None
**     Returns     : Nothing
** ===================================================================
*/
#pragma interrupt called /* Comment this line if the appropriate 'Interrupt preserve
registers' property */
                         /* is set to 'yes' (#pragma interrupt saveall is generated
before the ISR)      */
extern void setESCP(struct ESCP pack);
extern bool checkCRC(struct ESCP pack);

void AS_OnFullRxBuf(void)
{
       word Rx;
       struct ESCP receivedPack;

       AS_RecvBlock((byte*)&receivedPack, sizeof(receivedPack), &Rx);
       if (checkCRC(receivedPack))
              setESCP(receivedPack);
}

/*
** ===================================================================
**     Event       :  Btn_OnButton (module Events)
**
**     From bean   :  Btn [Button]
**     Description :
**         This event is called when the button is pressed.
**         If button inactivity feature (advanced view) is enabled,
**         then the next OnButton event is not generated during dead
**         time.
**     Parameters  : None
**     Returns     : Nothing
** ===================================================================
*/

#pragma interrupt called /* Comment this line if the appropriate 'Interrupt preserve
registers' property */
                         /* is set to 'yes' (#pragma interrupt saveall is generated
before the ISR)      */
extern void initProgram();
```

```
void Btn_OnButton(void)
{
       initProgram();
}

/** ###################################################################
**     End-of-file
**  ###################################################################*/
```

Chopper.h

```
/** ###################################################################
**     Project   : ELEPOT - UFRJ
**     Author    : Daniel Santos de O Cabral
**  ###################################################################*/

#ifndef __Chopper_H
#define __Chopper_H
/* MODULE Chopper */

#include "PE_Types.h"
#include "PE_Error.h"
#include "PE_Const.h"
#include "IO_Map.h"
#include "PE_Timer.h"
#include "AS.h"
#include "Btn.h"
#include "PPG_PWM.h"
#include "I_Timer.h"
#include "R_Timer.h"
#include "ADConv.h"
#include "Inhr1.h"

void PPG_PWM_OnEnd(void);

/* END Chopper */
#endif /* __Chopper_H*/

/** ###################################################################
**     End-of-file
**  ###################################################################*/
```

Chopper.c

```
/** ###################################################################
**     Project   : ELEPOT - UFRJ
**     Author    : Daniel Santos de O Cabral
**  ###################################################################*/

/* MODULE Chopper */

/* Including used modules for compiling procedure */
#include "Cpu.h"
#include "Events.h"
#include "Chopper.h"
#include "Inverter.h"
#include "Rectifier.h"
#include "ELEPOT.h"

/* Include shared modules, which are used for whole project */
#include "PE_Types.h"
#include "PE_Error.h"
#include "PE_Const.h"
#include "IO_Map.h"

/* global variables */
unsigned int fs = 0;
unsigned int dc = 0;

/* methods */
void setChopper(byte command, long fsw, long dcNormalized);
```

```
/* program */
void setChopper(byte command, long fsw, long dcNormalized)
{
        PPG_PWM_Disable();
        GPIOA_DATA &= CLR_BITS_0_TO_5; /* turn OFF all the leds */

        switch (command)
        {
                case 0 :
                        GPIOA_DATA |= SET_BIT_0; /* turn ON the red led */
                        break;
                case 1 :
                {
                        fs = (unsigned int) fsw;
                        dc = (unsigned int) dcNormalized;
                        PPG_PWM_Enable();
                        PPG_PWM_SetFreqHz(fs);
                        PPG_PWM_EnableEvent();
                        GPIOA_DATA |= SET_BIT_2; /* turn ON the green led */
                        break;
                }
        }
}

/*
** ===================================================================
**     Event       :  PPG_PWM_OnEnd (module Chopper)
**
**     From bean   :  PPG_PWM [PPG_PWM]
**     Description :
**         This event is called when the specified number of
**         iterations is generated. (Only when the bean is enabled -
**         <Enable> and the events are enabled - <EnableEvent>).
**     Parameters  : None
**     Returns     : Nothing
** ===================================================================
*/
#pragma interrupt called /* Comment this line if the appropriate 'Interrupt preserve
registers' property */
                         /* is set to 'yes' (#pragma interrupt saveall is generated
before the ISR)      */
void PPG_PWM_OnEnd(void)
{
        PPG_PWM_SetRatio16(dc);
        PPG_PWM_DisableEvent();
}

/** ###################################################################
**     End-of-file
**  ###################################################################*/
```

q_num.h

```
#define Q1      2
#define Q2      4
#define Q3      8
#define Q4      16
#define Q5      32
#define Q6      64
#define Q7      128
#define Q8      256
#define Q9      512
#define Q10     1024
#define Q11     2048
#define Q12     4096
#define Q13     8192
#define Q14     16384
#define Q15     32768L
#define Q16     65536L
#define Q17     131072L
#define Q18     262144L
#define Q19     524288L
#define Q20     1048576L
#define Q21     2097152L
#define Q22     4194304L
#define Q23     8388608L
```

```
#define Q24     16777216L
#define Q25     33554432L
#define Q26     67108864L
#define Q27     134217728L
#define Q28     268435456L
#define Q29     536870912L
#define Q30     1073741824L
#define Q31     2147483648L
#define Q32     4294967296L
#define Q33     8589934592L
#define Q34     17179869184L
#define Q35     34359738368L
#define Q36     68719476737L
```

sencos.h

```
#define NTAB 17

typedef struct  { unsigned short ang;
                                short int sen;
                                short int cos;
                              } sencos_st;

void sencos( sencos_st *st);
```

sencos.c

```
/*********************************************************************/
/*                                                                   */
/*  FILE        :sencos.c                                            */
/*  DATE        :Fri, May 16, 2003                                   */
/*  DESCRIPTION :function to calculate sin and cos by interpolation  */
/*               input range -1 to +1 (Q15) maps to -2*pi to +2*pi    */
/*                                                                   */
/*********************************************************************/

#include "q_num.h"
#include "sencos.h"

const short int sin_tab_Q15[] = {0, 3211, 6392, 9511, 12539, 15446, 18204,
                                 20787, 23169, 25329, 27244, 28897, 30272,
                                 31356, 32137, 32609, 32767};
void sencos(sencos_st *st)
{
        unsigned short int sect, quad, offs;
        short int ind1, ind2, ind3, ind4, z1, z2, z3, z4;

        sect = (st->ang) >> 9;
        offs = (st->ang) - (sect << 9);
        quad = sect >> 4;
        ind1 = sect - (quad << 4);
        ind3 = ind1;
        ind2 = ind1 + 1;
        ind4 = ind2;
        quad = quad - ((quad >> 2) << 2);

        if ((quad == 1) || (quad == 3))
        {
                ind1 = (NTAB - 1) - ind1;
                ind2 = ind1 - 1;
        }
        else
        {
                ind3 = (NTAB - 1) - ind3;
                ind4 = ind3 - 1;
        }

        z1 = sin_tab_Q15[ind1];
        z2 = sin_tab_Q15[ind2];
        z3 = sin_tab_Q15[ind3];
        z4 = sin_tab_Q15[ind4];
```

```
        if ((quad == 2) || (quad == 3))
        {
                z1 = -z1;
                z2 = -z2;
        }
        if ((quad == 1) || (quad == 2))
        {
                z3 = -z3;
                z4 = -z4;
        }

        (st->sen) = z1 + (short)(((long)offs * (z2 - z1)) >> 9);
        (st->cos) = z3 + (short)(((long)offs * (z4 - z3)) >> 9);
}
```

Inverter.h

```
/** ###################################################################
**     Project   : ELEPOT - UFRJ
**     Author    : Daniel Santos de O Cabral
**  ###################################################################*/

#ifndef __Inverter_H
#define __Inverter_H
/* MODULE Inverter */

#include "PE_Types.h"
#include "PE_Error.h"
#include "PE_Const.h"
#include "IO_Map.h"
#include "PE_Timer.h"
#include "AS.h"
#include "Btn.h"
#include "PPG_PWM.h"
#include "I_Timer.h"
#include "R_Timer.h"
#include "ADConv.h"
#include "Inhr1.h"

#define CTE_Tsa_i_Q30        6795L /* 158kHz at Q30 */

void I_Timer_OnInterrupt(void);

/* END Inverter */
#endif /* __Inverter_H*/

/** ###################################################################
**     End-of-file
**  ###################################################################*/
```

Inverter.c

```
/** ###################################################################
**     Project   : ELEPOT - UFRJ
**     Author    : Daniel Santos de O Cabral
**  ###################################################################*/

/* MODULE Inverter */

/* Including used modules for compiling procedure */
#include "Cpu.h"
#include "Events.h"
#include "Chopper.h"
#include "Inverter.h"
#include "Rectifier.h"
#include "sencos.h"
#include "ELEPOT.h"

/* Include shared modules, which are used for whole project */
#include "PE_Types.h"
#include "PE_Error.h"
#include "PE_Const.h"
#include "IO_Map.h"
```

```
/* global variables */
bool inverter_on = FALSE;
unsigned long t_sw_counter = 0;
unsigned long f_sw = 1;
unsigned long Tsw = 0;
unsigned long t_rf_counter = 0;
unsigned long f_rf = 1;
unsigned long Trf = 0;
long v_rf;
unsigned long w = 0;
unsigned long Tsw_1_4 = 0;
unsigned long Tsw_3_4 = 0;
long Ki = 0;

/* methods */
void setInverter(byte command, long fsw, long frf, long vrf);
int cos(unsigned long angle_Q29);

/* program */
void setInverter(byte command, long fsw, long frf, long vrf)
{
	switch (command)
	{
		case 0 :
			inverter_on = FALSE;
			break;
		case 1 :
		{
			f_sw = (unsigned long) fsw;
			f_rf = (unsigned long) frf;
			Tsw = ((unsigned long)1 << 28) / f_sw;   /* Tsw at Q28 */
			Tsw_1_4 = Tsw / (unsigned long)4;        /* Tsw_1_4 at Q28 */
			Tsw_3_4 = (unsigned long)3 * Tsw_1_4;    /* Tsw_3_4 at Q28 */
			Trf = ((unsigned long)1 << 24) / f_rf;   /* Trf at Q24 */
			v_rf = vrf;                              /* v_rf at Q14 */
			w = (2 * f_rf) << 11;                    /* w at Q11 */
			Ki = ((long)4 << 28) / (long)(Tsw >> 2); /* Ki at Q2 */
			GPIOA_DATA &= CLR_BIT_0; /* turn OFF the red led */
			GPIOA_DATA |= SET_BIT_2; /* turn ON the green led */
			inverter_on = TRUE;
			I_Timer_Enable();
			break;
		}
	}
}

int cos(unsigned long angle_Q15)
{
	sencos_st* a = NULL;

	(a->ang) = (unsigned short)angle_Q15;
	sencos(a);

	return (a->cos); /* return value at Q15 */
}

/*
** ===================================================================
**     Event       :  I_Timer_OnInterrupt (module Inverter)
**
**     From bean   :  I_Timer [TimerInt]
**     Description :
**         When a timer interrupt occurs this event is called (only
**         when the bean is enabled - "Enable" and the events are
**         enabled - "EnableEvent").
**     Parameters  : None
**     Returns     : Nothing
** ===================================================================
*/
#pragma interrupt called /* Comment this line if the appropriate 'Interrupt preserve
registers' property */
                         /* is set to 'yes' (#pragma interrupt saveall is generated
before the ISR)      */
```

```
void I_Timer_OnInterrupt(void)
{
	unsigned long t_sw = t_sw_counter * CTE_Tsa_i_Q30; /* t_sw at Q30 */
	unsigned long t_rf = t_rf_counter * CTE_Tsa_i_Q30; /* t_rf at Q30 */
	unsigned long ang;
	long Vt;
	long Vr;

	if (!inverter_on)
	{
		GPIOA_DATA &= CLR_BITS_0_TO_5; /* turn OFF all the leds */
		GPIOA_DATA |= SET_BIT_0;       /* turn ON the red led */
		I_Timer_Disable();
	}
	else
	{
		if (t_sw < (Tsw << 2))
		{
			++t_sw_counter;
		}
		else
		{
			t_sw_counter = 0;
		}

		if (t_rf < (Trf << 6))
		{
			++t_rf_counter;
		}
		else
		{
			t_rf_counter = 0;
		}

		if (t_sw <= (Tsw_1_4 << 2))
		{
			Vt = Ki * (long)(t_sw >> 3); /* Vt at Q29 */
		}
		else if (t_sw >= (Tsw_3_4 << 2))
		{
			Vt = ((Ki * (long)(t_sw >> 4)) - ((long)4 << 28)) << 1; /* Vt at
Q29 */
		}
		else
		{
			Vt = ((long)-1 * Ki * (long)(t_sw >> 3)) + ((long)2 << 29); /* Vt
at Q29 */
		}
		ang = ((t_rf >> 12) * w) >> 14; /* ang at Q15 */
		Vr = v_rf * cos(ang); /* Vr at Q29 */

		if (Vt < Vr)
		{
			GPIOA_DATA |= SET_BIT_4; /* turn ON the right orange led (output
at 3.3V) */
			GPIOA_DATA &= CLR_BIT_1; /* turn OFF the left orange led (output
at 0V) */
		}
		else
		{
			GPIOA_DATA |= SET_BIT_1; /* turn ON the left orange led (output at
3.3V) */
			GPIOA_DATA &= CLR_BIT_4; /* turn OFF the right orange led (output
at 0V) */
		}
	}
}

/* END Inverter */

/** ###################################################################
**     End-of-file
**  ###################################################################*/
```

Rectifier.h

```
/** ###################################################################
**     Project   : ELEPOT - UFRJ
**     Author    : Daniel Santos de O Cabral
**  ###################################################################*/

#ifndef __Rectifier_H
#define __Rectifier_H

/* MODULE Rectifier */

#include "PE_Types.h"
#include "PE_Error.h"
#include "PE_Const.h"
#include "IO_Map.h"
#include "PE_Timer.h"
#include "AS.h"
#include "Btn.h"
#include "R_Timer.h"
#include "ADConv.h"
#include "Inhr1.h"
#include "PPG_PWM.h"
#include "I_Timer.h"

#define CTE_Tsa_r_Q30     5368L /* 200kHz at Q30 */

void R_Timer_OnInterrupt(void);
void ADConv_OnEnd(void);

/* END Rectifier */
#endif /* __Rectifier_H*/

/** ###################################################################
**     End-of-file
**  ###################################################################*/
```

Rectifier.c

```
/** ###################################################################
**     Project   : ELEPOT - UFRJ
**     Author    : Daniel Santos de O Cabral
**  ###################################################################*/

/* MODULE Rectifier */

/* Including used modules for compiling procedure */
#include "Cpu.h"
#include "Events.h"
#include "Chopper.h"
#include "Inverter.h"
#include "Rectifier.h"
#include "ELEPOT.h"

/* Include shared modules, which are used for whole project */
#include "PE_Types.h"
#include "PE_Error.h"
#include "PE_Const.h"
#include "IO_Map.h"

/* global variables */
bool rectifier_on = FALSE;
unsigned long t_counter = 0;
unsigned long alpha_1 = 0;
unsigned long alpha_2 = 0;
unsigned int p_1 = 0;
unsigned int p_2 = 0;
unsigned long Ti = 0;
unsigned long Kr = 0;

/* methods */
void setRectifier(byte command, long alpha, long fi);
```

```
/* program */
void setRectifier(byte command, long alpha, long fi)
{
	switch (command)
	{
		case 0 :
			ADConv_Stop();
			rectifier_on = FALSE;
			break;
		case 1 :
		{
			Ti = ((unsigned long)1 << 24) / fi;          /* Ti at Q24 */
			Kr = ((unsigned long)360 << 23) / (Ti >> 8); /* Kr at Q7 */
			alpha_1 = (unsigned int)alpha;               /* alpha_1 at Q0 */
			alpha_2 = (unsigned int)alpha + 180;         /* alpha_2 at Q0 */
			GPIOA_DATA &= CLR_BITS_0_TO_5; /* turn OFF all the leds */
			GPIOA_DATA |= SET_BIT_2;       /* turn ON the green led */
			rectifier_on = TRUE;
			ADConv_Start();
			R_Timer_Enable();
			break;
		}
	}
}

/*
** ===================================================================
**     Event       :  ADConv_OnEnd (module Rectifier)
**
**     From bean   :  ADConv [ADConv]
**     Description :
**         This event is called after the measurement (which
**         consists of <1 or more conversions>) is/are finished.
**     Parameters  : None
**     Returns     : Nothing
** ===================================================================
*/
#pragma interrupt called /* Comment this line if the appropriate 'Interrupt preserve
registers' property */
                         /* is set to 'yes' (#pragma interrupt saveall is generated
before the ISR)      */
void ADConv_OnEnd(void)
{
	int Values[1];
	int sync_signal = 0;

	ADConv_GetValue16((unsigned int *)Values);
	sync_signal = Values[0] >> 2; /* sync_signal at Q14 */

	/* IMPORTANT *
	The sync_signal is the value read on the ADConv pins every time (normalized from
0 to 1)
	This variable is used to syncronize the start point of the controller with the
power signal
	As suggestion, a PLL system can be used for futher implementation
	Actually, the syncronize system resets the controller signal when any non-zero
pulse is read
	*/

	if (sync_signal > 0)
	{
		t_counter = 0;
	}
}

/*
** ===================================================================
**     Event       :  R_Timer_OnInterrupt (module Rectifier)
**
**     From bean   :  R_Timer [TimerInt]
**     Description :
**         When a timer interrupt occurs this event is called (only
**         when the bean is enabled - "Enable" and the events are
**         enabled - "EnableEvent").
**     Parameters  : None
**     Returns     : Nothing
** ===================================================================
```

```
*/
#pragma interrupt called /* Comment this line if the appropriate 'Interrupt preserve
registers' property */
                         /* is set to 'yes' (#pragma interrupt saveall is generated
before the ISR)      */
void R_Timer_OnInterrupt(void)
{
        unsigned long t = t_counter * CTE_Tsa_r_Q30; /* t at Q30 */

        if (!rectifier_on)
        {
                GPIOA_DATA &= CLR_BITS_0_TO_5; /* turn OFF all the leds */
                GPIOA_DATA |= SET_BIT_0;       /* turn ON the red led */
                R_Timer_Disable();
                ADConv_Disable();
        }
        else
        {
                GPIOA_DATA &= CLR_BIT_1; /* turn OFF the left orange led (output at 0V)
*/
                GPIOA_DATA &= CLR_BIT_4; /* turn OFF the right orange led (output at 0V)
*/

                if (t < (Ti << 6))
                {
                        ++t_counter;
                }
                else
                {
                        t_counter = 0;
                        p_1 = 0;
                        p_2 = 0;
                }

                if (((Kr * (t >> 14)) >= (alpha_1 << 23)) && (p_1 < 16))
                {
                        if ((p_1 % 2) == 0)
                        {
                                GPIOA_DATA |= SET_BIT_4; /* turn ON the right orange led
(output at 3.3V) */
                        }
                        ++p_1;
                }

                if (((Kr * (t >> 14)) >= (alpha_2 << 23)) && (p_2 < 16))
                {
                        if ((p_2 % 2) == 0)
                        {
                                GPIOA_DATA |= SET_BIT_1; /* turn ON the left orange led
(output at 3.3V) */
                        }
                        ++p_2;
                }
        }
}

/* END Rectifier */

/** ###################################################################
**     End-of-file
**  ###################################################################*/
```

Apêndice C

Listagem do código-fonte da aplicação *web*

C.1 – Arquivos de configuração

web.xml

```
<?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">
    <description>Aplicativo web do Laboratório de Eletrônica de Potência</description>
    <display-name>ELEPOT_LAB</display-name>
    <listener>
        <listener-class>br.ufrj.dee.elepot.util.listener.SessaoListener</listener-class>
    </listener>
    <servlet>
        <servlet-name>Elepot</servlet-name>
        <servlet-class>br.ufrj.dee.elepot.struts.ElepotServlet</servlet-class>
        <init-param>
            <param-name>config</param-name>
            <param-value>/WEB-INF/struts-config.xml</param-value>
        </init-param>
        <init-param>
            <param-name>debug</param-name>
            <param-value>2</param-value>
        </init-param>
        <init-param>
            <param-name>detail</param-name>
            <param-value>2</param-value>
        </init-param>
        <load-on-startup>2</load-on-startup>
        </servlet>
    <servlet-mapping>
        <servlet-name>Elepot</servlet-name>
        <url-pattern>*.do</url-pattern>
    </servlet-mapping>
    <session-config>
        <session-timeout>5</session-timeout>
    </session-config>
    <welcome-file-list>
        <welcome-file>
            index.jsp
        </welcome-file>
    </welcome-file-list>
    <error-page>
        <exception-type>java.lang.Exception</exception-type>
        <location>/WEB-INF/erro.jsp</location>
    </error-page>
    <error-page>
        <error-code>404</error-code>
        <location>/WEB-INF/erro.jsp</location>
    </error-page>
    </web-app>
```

context.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<Context path="/elepot"/>
```

struts-config.xml

```
<?xml version="1.0" encoding="ISO-8859-1" ?>

<!DOCTYPE struts-config PUBLIC
          "-//Apache Software Foundation//DTD Struts Configuration 1.2//EN"
          "http://jakarta.apache.org/struts/dtds/struts-config_1_2.dtd">
```

```
<struts-config>
    <form-beans>
        <form-bean name="LoginFormBean"
type="br.ufrj.dee.elepot.struts.form.LoginFormBean"/>
        <form-bean name="ChopperFormBean"
type="br.ufrj.dee.elepot.struts.form.ChopperFormBean"/>
        <form-bean name="InversorFormBean"
type="br.ufrj.dee.elepot.struts.form.InversorFormBean"/>
        <form-bean name="RetificadorFormBean"
type="br.ufrj.dee.elepot.struts.form.RetificadorFormBean"/>
        <form-bean name="AlunoFormBean"
type="br.ufrj.dee.elepot.struts.form.AlunoFormBean"/>
        <form-bean name="ExperimentoFormBean"
type="br.ufrj.dee.elepot.struts.form.ExperimentoFormBean"/>
        <form-bean name="ConfiguracaoFormBean"
type="br.ufrj.dee.elepot.struts.form.ConfiguracaoFormBean"/>
        <form-bean name="TrocaSenhaFormBean"
type="br.ufrj.dee.elepot.struts.form.TrocaSenhaFormBean"/>
        <form-bean name="RelatorioPraticaFormBean"
type="br.ufrj.dee.elepot.struts.form.RelatorioPraticaFormBean"/>
    </form-beans>

    <global-exceptions>

    </global-exceptions>

    <global-forwards>

    </global-forwards>

    <action-mappings>
        <!-- links de navegação -->
        <action path="/telaManutencao" forward="/WEB-INF/manutencao.jsp"/>
        <action path="/telaAlunos" forward="/WEB-INF/alunos.jsp"/>
        <action path="/telaRelatorioPratica" forward="/WEB-INF/relatorio_pratica.jsp"/>
        <action path="/telaIncluirAlunos" forward="/WEB-INF/alunos_form.jsp"/>
        <action path="/telaExperimentos" forward="/WEB-INF/experimentos.jsp"/>
        <action path="/telaIncluirExperimentos" forward="/WEB-
INF/experimentos_form.jsp"/>
        <action path="/telaSenha" forward="/WEB-INF/senha.jsp"/>

        <!-- ações da lógica de negócios -->
        <action path="/loginLab" name="LoginFormBean"
type="br.ufrj.dee.elepot.struts.action.Login" scope="request">
            <forward name="negado" path="/WEB-INF/login_erro.jsp"/>
            <forward name="aceito" path="/montaLab.do"/>
        </action>
        <action path="/telaLab_*" type="br.ufrj.dee.elepot.struts.action.MontaPratica"
scope="request">
            <forward name="ocupado" path="/WEB-INF/lab_ocupado.jsp"/>
            <forward name="ok" path="/WEB-INF/exp_{1}.jsp"/>
        </action>
        <action path="/loginMan" name="LoginFormBean"
type="br.ufrj.dee.elepot.struts.action.Login" scope="request">
            <forward name="negado" path="/WEB-INF/login_erro.jsp"/>
            <forward name="aceito" path="/WEB-INF/manutencao.jsp"/>
        </action>
        <action path="/logout" type="br.ufrj.dee.elepot.struts.action.Logout"
scope="request">
            <forward name="ok" path="/index.jsp"/>
        </action>
        <action path="/montaLab" type="br.ufrj.dee.elepot.struts.action.MontaLab"
scope="request">
            <forward name="ok" path="/WEB-INF/laboratorio.jsp"/>
        </action>
        <action path="/montaPopup" type="br.ufrj.dee.elepot.struts.action.MontaPopup"
scope="request">
            <forward name="descricao" path="/WEB-INF/popup_descricao.jsp"/>
            <forward name="instrucoes" path="/WEB-INF/popup_instrucoes.jsp"/>
        </action>
        <action path="/comandaChopper_*" name="ChopperFormBean"
type="br.ufrj.dee.elepot.struts.action.ComandaChopper" scope="request">
            <forward name="ok" path="/WEB-INF/exp_{1}.jsp"/>
        </action>
```

```
        <action path="/comandaInversor_*" name="InversorFormBean"
type="br.ufrj.dee.elepot.struts.action.ComandaInversor" scope="request">
            <forward name="ok" path="/WEB-INF/exp_{1}.jsp"/>
        </action>
        <action path="/comandaRetificador_*" name="RetificadorFormBean"
type="br.ufrj.dee.elepot.struts.action.ComandaRetificador" scope="request">
            <forward name="ok" path="/WEB-INF/exp_{1}.jsp"/>
        </action>
        <action path="/pesquisaAlunos" name="AlunoFormBean"
type="br.ufrj.dee.elepot.struts.action.PesquisaAlunos" scope="request">
            <forward name="ok" path="/WEB-INF/alunos.jsp"/>
        </action>
        <action path="/incluiAluno" name="AlunoFormBean"
type="br.ufrj.dee.elepot.struts.action.IncluiAluno" scope="request">
            <forward name="ok" path="/WEB-INF/alunos_ok.jsp"/>
            <forward name="duplicado" path="/WEB-INF/alunos_form.jsp"/>
        </action>
        <action path="/alteraAluno" name="AlunoFormBean"
type="br.ufrj.dee.elepot.struts.action.AlteraAluno" scope="request">
            <forward name="tela" path="/WEB-INF/alunos_form.jsp"/>
            <forward name="ok" path="/WEB-INF/alunos_ok.jsp"/>
            <forward name="duplicado" path="/WEB-INF/alunos_form.jsp"/>
        </action>
        <action path="/excluiAluno" name="AlunoFormBean"
type="br.ufrj.dee.elepot.struts.action.ExcluiAluno" scope="request">
            <forward name="ok" path="/WEB-INF/alunos_ok.jsp"/>
        </action>
        <action path="/geraRelatorioPratica" name="RelatorioPraticaFormBean"
type="br.ufrj.dee.elepot.struts.action.GeraRelatorioPratica" scope="request">
            <forward name="ok" path="/WEB-INF/relatorio_pratica.jsp"/>
        </action>
        <action path="/encerraTurma"
type="br.ufrj.dee.elepot.struts.action.EncerraTurma" scope="request">
            <forward name="ok" path="/WEB-INF/turma_encerrada_ok.jsp"/>
        </action>
        <action path="/pesquisaExperimentos" name="ExperimentoFormBean"
type="br.ufrj.dee.elepot.struts.action.PesquisaExperimentos" scope="request">
            <forward name="ok" path="/WEB-INF/experimentos.jsp"/>
        </action>
        <action path="/incluiExperimento" name="ExperimentoFormBean"
type="br.ufrj.dee.elepot.struts.action.IncluiExperimento" scope="request">
            <forward name="ok" path="/WEB-INF/experimentos_ok.jsp"/>
        </action>
        <action path="/alteraExperimento" name="ExperimentoFormBean"
type="br.ufrj.dee.elepot.struts.action.AlteraExperimento" scope="request">
            <forward name="tela" path="/WEB-INF/experimentos_form.jsp"/>
            <forward name="ok" path="/WEB-INF/experimentos_ok.jsp"/>
        </action>
         <action path="/excluiExperimento" name="ExperimentoFormBean"
type="br.ufrj.dee.elepot.struts.action.ExcluiExperimento" scope="request">
            <forward name="ok" path="/WEB-INF/experimentos_ok.jsp"/>
        </action>
        <action path="/telaConfiguracao" name="ConfiguracaoFormBean"
type="br.ufrj.dee.elepot.struts.action.CarregaConfiguracao" scope="request">
            <forward name="ok" path="/WEB-INF/configuracao.jsp"/>
        </action>
        <action path="/gravaConfiguracao" name="ConfiguracaoFormBean"
type="br.ufrj.dee.elepot.struts.action.GravaConfiguracao" scope="request">
            <forward name="ok" path="/WEB-INF/configuracao_ok.jsp"/>
        </action>
        <action path="/trocaSenha" name="TrocaSenhaFormBean"
type="br.ufrj.dee.elepot.struts.action.TrocaSenha" scope="request">
            <forward name="ok" path="/WEB-INF/senha_ok.jsp"/>
            <forward name="erro" path="/WEB-INF/erro.jsp"/>
        </action>
    </action-mappings>

    <message-resources parameter="br.ufrj.dee.elepot.struts.ApplicationResource"/>

</struts-config>
```

C.2 – Pacote br.ufrj.dee.elepot

Protocolo.java

```
package br.ufrj.dee.elepot;

import br.ufrj.dee.elepot.util.Conversao;

public final class Protocolo {
    private byte[] pacote;

    public Protocolo() {
    }

    public void setPacote(byte conversor, byte comando, int[] parametros) {
        byte[] pacote = new byte[23];

        // bytes particulares (parâmetros)
        byte[] p = new byte[16];
        for (int i = 0; i < 4; i++)
            for (int j = 0; j < 4; j++) {
                p[(4 * i) + j] = Conversao.IntToByte(parametros[i])[3 - j];
            }
        for (int i = 0; i < 16; i++) {
            pacote[i] = p[i];
        }
        // bytes particulares (conversor e comando)
        pacote[16] = conversor;
        pacote[17] = comando;
        // bytes de checagem (crc)
        pacote[18] = calculaCRC(conversor, comando, parametros);
        // bytes reservados
        pacote[19] = 0;
        pacote[20] = 0;
        pacote[21] = 0;
        pacote[22] = 0;

        this.pacote = pacote;
    }

    public byte[] getPacote() {
        return this.pacote;
    }
    private byte calculaCRC(byte conversor, byte comando, int[] parametros) {
        int soma = conversor + comando;
        for (int i = 0; i < parametros.length; i++) {
            soma += parametros[i];
        }
        return (byte) (soma & 0x000000000000000000000000FFFFFFFF);
    }
}
```

MC56F8013.java

```
package br.ufrj.dee.elepot;

public final class MC56F8013 {

    public static final int getDutyCycle(float dutyCycle) {
        return (int) Math.floor(65535 * (1 - dutyCycle));
    }

    public static final int getFrequency(float frequency) {
        return (int) frequency;
    }

    public static final int getAmplitude(float amplitude) {
        return (int) Math.floor(16384 * amplitude);
    }

    public static final int getAlpha(float alpha) {
        return (int) alpha;
    }
}
```

Conversor.java

```
package br.ufrj.dee.elepot;

import br.ufrj.dee.elepot.dao.ConfiguracaoDAO;
import br.ufrj.dee.elepot.model.Configuracao;
import br.ufrj.dee.elepot.util.Conversao;
import br.ufrj.dee.elepot.util.DispositivoSerial;
import java.sql.SQLException;

public abstract class Conversor {

    public void set(byte codigoConversor, byte comando, int[] parametros) {
        Protocolo protocolo = new Protocolo();
        protocolo.setPacote(codigoConversor, comando, parametros);
        try {
            ConfiguracaoDAO cfgDAO = new ConfiguracaoDAO();
            Configuracao cfg = cfgDAO.carrega();
            System.out.println("ELEPOT: transmitindo bytes pela porta serial " +
cfg.getPortaSerial() + "...");
            for (int j = 0; j < 23; j++) {
                String B =
Conversao.BitToString(Conversao.ByteToBit(protocolo.getPacote()[j]));
                System.out.println(B);
            }
            DispositivoSerial dispSerial = new DispositivoSerial(cfg.getPortaSerial(),
2, 2000);
            dispSerial.setPacote(protocolo.getPacote());
            if (dispSerial.enviarPacote() == true) {
                System.out.println("ELEPOT: envio com sucesso");
                dispSerial.encerrarDispositivo(2000);
            } else {
                System.out.println("ELEPOT: ocorreu uma falha durante o envio");
            }
        } catch (SQLException ex) {
            ex.printStackTrace();
        }
    }
}
```

Chopper.java

```
package br.ufrj.dee.elepot;

import br.ufrj.dee.elepot.util.Conversao;

public final class Chopper extends Conversor {
    public static final String CHOPPER_BUCK = "BUCK";
    public static final String CHOPPER_BOOST = "BOOST";
    public static final String CHOPPER_BUCKBOOST = "BUCKBOOST";

    public static final byte CHOPPER_COMANDO_DESLIGAR = 0;
    public static final byte CHOPPER_COMANDO_LIGAR = 1;
    private static final byte CHOPPER_CODIGO_CONVERSOR = 1;

    private static final float CHOPPER_MAX_FS = 2000;    // Hz
    private static final float CHOPPER_MIN_FS = 29.999f; // Hz
    private static final float CHOPPER_MAX_DC = 99.999f; // %
    private static final float CHOPPER_MIN_DC = 0f;      // %

    private String tipo;    // tipo do chopper
    private float fs = 1;   // frequência de chaveamento
    private float dc = 0;   // ciclo de trabalho (0 <= dc =< 1)

    public Chopper() {
        this.tipo = CHOPPER_BUCKBOOST;
    }

    public Chopper(String tipo) {
        if (tipo.equals(CHOPPER_BUCK) || tipo.equals(CHOPPER_BOOST) ||
tipo.equals(CHOPPER_BUCKBOOST)) {
            this.tipo = tipo;
        } else {
            this.tipo = CHOPPER_BUCKBOOST;
        }
    }
```

```
    public void setFs(String fs) {
        this.fs = Math.max(CHOPPER_MIN_FS, Math.min(CHOPPER_MAX_FS,
Conversao.StringToFloat(fs)));
    }

    public float getFs() {
        return this.fs;
    }

    public void setDc(String dc) {
        this.dc = Math.max(CHOPPER_MIN_DC, Math.min(CHOPPER_MAX_DC,
Conversao.StringToFloat(dc))) * 0.01f;
    }

    public float getDc() {
        return this.dc;
    }

    public String getDcString() {
        return Conversao.FloatToString(getDc() * 100f, 2);
    }

    public float getTs() {
        return (float) Math.pow(getFs(), -1);
    }

    public String getTsString() {
        return Conversao.FloatToString(getTs() * 1000, 3); // em ms
    }

    public float getTon() {
        return getDc() * getTs();
    }

    public String getTonString() {
        return Conversao.FloatToString(getTon() * 1000, 3); // em ms
    }

    public float getToff() {
        return getTs() - getTon();
    }

    public String getToffString() {
        return Conversao.FloatToString(getToff() * 1000, 3); // em ms
    }

    public float getGanho() {
        if (this.tipo.equals(CHOPPER_BUCK)) {
            return getDc();
        } else {
            if (this.tipo.equals(CHOPPER_BOOST)) {
                return (float) Math.pow(1 - getDc(), -1);
            } else {
                return (float) (getDc() * Math.pow(1 - getDc(), -1));
            }
        }
    }

    public String getGanhoString() {
        float g = getGanho();
        if (g < 1)
            return Conversao.FloatToString(getGanho(), 3);
        else
            return Conversao.FloatToString(getGanho(), 1);
    }

    public void setChopper(byte comando) {
        int[] p = new int[4];
        p[0] = MC56F8013.getFrequency(getFs());
        p[1] = MC56F8013.getDutyCycle(getDc());
        p[2] = 0;
        p[3] = 0;

        super.set(CHOPPER_CODIGO_CONVERSOR, comando, p);
    }
}
```

Inversor.java

```
package br.ufrj.dee.elepot;

import br.ufrj.dee.elepot.util.Conversao;

public final class Inversor extends Conversor {
    public static final byte INVERSOR_COMANDO_DESLIGAR = 0;
    public static final byte INVERSOR_COMANDO_LIGAR = 1;
    private static final byte INVERSOR_CODIGO_CONVERSOR = 2;

    private static final float INVERSOR_MAX_FS = 4000;     // Hz
    private static final float INVERSOR_MIN_FS = 359.999f; // Hz
    private static final float INVERSOR_MAX_FREF = 180;    // Hz
    private static final float INVERSOR_MIN_FREF = 1;      // Hz
    private static final float INVERSOR_MAX_VREF = 3;      // pu
    private static final float INVERSOR_MIN_VREF = 0;      // pu

    private float fs = 1;   // frequência de chaveamento
    private float fref = 1; // frequência de referência
    private float vref = 0; // tensão de referência

    public Inversor() {
    }

    public void setFs(String fs) {
        this.fs = Math.max(INVERSOR_MIN_FS, Math.min(INVERSOR_MAX_FS,
Conversao.StringToFloat(fs)));
    }

    public float getFs() {
        return this.fs;
    }

    public void setFref(String fref) {
        this.fref = Math.max(INVERSOR_MIN_FREF, Math.min(INVERSOR_MAX_FREF,
Conversao.StringToFloat(fref)));
    }

    public float getFref() {
        return this.fref;
    }

    public String getFrefString() {
        return Conversao.FloatToString(getFref(), 3);
    }

    public void setVref(String vref) {
        this.vref = Math.max(INVERSOR_MIN_VREF, Math.min(INVERSOR_MAX_VREF,
Conversao.StringToFloat(vref)));
    }

    public float getVref() {
        return this.vref;
    }

    public String getVrefString() {
        return Conversao.FloatToString(getVref(), 3);
    }

    public float getTs() {
        return (float) Math.pow(getFs(), -1);
    }

    public String getTsString() {
        return Conversao.FloatToString(getTs() * 1000, 3); // em ms
    }

    public float getMf() {
        return getFs() / getFref();
    }

    public String getMfString() {
        return Conversao.FloatToString(getMf(), 0);
    }
```

```
    public float getMa() {
        return getVref() * 100f;
    }
    public String getMaString() {
        return Conversao.FloatToString(getMa(), 0);
    }

    public void setInversor(byte comando) {
        int[] p = new int[4];
        p[0] = MC56F8013.getFrequency(getFs());
        p[1] = MC56F8013.getFrequency(getFref());
        p[2] = MC56F8013.getAmplitude(getVref());
        p[3] = 0;

        super.set(INVERSOR_CODIGO_CONVERSOR, comando, p);
    }
}
```

Retificador.java

```
package br.ufrj.dee.elepot;

import br.ufrj.dee.elepot.util.Conversao;

public final class Retificador extends Conversor {
    public static final byte RETIFICADOR_COMANDO_DESLIGAR = 0;
    public static final byte RETIFICADOR_COMANDO_LIGAR = 1;
    private static final byte RETIFICADOR_CODIGO_CONVERSOR = 4;

    private static final float RETIFICADOR_MAX_ALPHA = 180; // °
    private static final float RETIFICADOR_MIN_ALPHA = 0;   // °

    private float alpha = 0; // ângulo de disparo
    private float freq = 60; // frquência da rede

    public Retificador() {
    }

    public void setAlpha(String alpha) {
        this.alpha = Math.max(RETIFICADOR_MIN_ALPHA, Math.min(RETIFICADOR_MAX_ALPHA,
Conversao.StringToFloat(alpha)));
    }

    public float getAlpha() {
        return this.alpha;
    }

    public void setFreq(String freq) {
        this.freq = Conversao.StringToFloat(freq);
    }

    public float getFreq() {
        return this.freq;
    }

    public String getAlphaString() {
        return Conversao.FloatToString(getAlpha(), 0);
    }

    public String getFreqString() {
        return Conversao.FloatToString(getFreq(), 0);
    }

    public float getVo() {
        return (float) ((2 / Math.PI) * Math.cos(Math.toRadians(getAlpha())));
    }

    public String getVoString() {
        return Conversao.FloatToString(getVo(), 1);
    }
```

```
    public void setRetificador(byte comando) {
        int[] p = new int[4];
        p[0] = MC56F8013.getAlpha(getAlpha());
        p[1] = MC56F8013.getFrequency(getFreq());
        p[2] = 0;
        p[3] = 0;

        super.set(RETIFICADOR_CODIGO_CONVERSOR, comando, p);
    }
}
```

C.3 – Pacote br.ufrj.dee.elepot.bo

UsuarioBO.java

```
package br.ufrj.dee.elepot.bo;

import br.ufrj.dee.elepot.dao.UsuarioDAO;
import br.ufrj.dee.elepot.model.Usuario;
import br.ufrj.dee.elepot.struts.form.AlunoFormBean;
import br.ufrj.dee.elepot.util.LoginStatus;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.Collection;
import javax.servlet.ServletContext;
import javax.servlet.http.HttpSession;

public final class UsuarioBO {

    public static boolean isLoginDuplicado(Usuario usuario) {
        try {
            UsuarioDAO usuarioDAO = new UsuarioDAO();
            return usuarioDAO.loginDuplicado(usuario);
        } catch (SQLException e) {
            e.printStackTrace();
            return false;
        }
    }

    public static LoginStatus isLoginValido(Usuario usuario) {
        try {
            UsuarioDAO usuarioDAO = new UsuarioDAO();
            return usuarioDAO.login(usuario);
        } catch (SQLException e) {
            e.printStackTrace();
            LoginStatus loginStatus = new LoginStatus();
            return loginStatus;
        }
    }

    public static LoginStatus isLoginOnline(ServletContext servletContext, Usuario
usuario) {
        LoginStatus loginStatus = new LoginStatus();
        Collection usuariosOnline = (ArrayList)
servletContext.getAttribute("usuariosOnline");
        if (usuariosOnline.contains(usuario.getLogin())) {
            loginStatus.setLoginDuplicado();
        } else {
            loginStatus.setLoginValido();
        }
        return loginStatus;
    }

    public static void setOnline(ServletContext servletContext, Usuario usuario) {
        Collection usuariosOnline = (ArrayList)
servletContext.getAttribute("usuariosOnline");
        usuariosOnline.add(usuario.getLogin());
        servletContext.setAttribute("usuariosOnline", usuariosOnline);
    }

    public static void setOffline(ServletContext servletContext, Usuario usuario) {
        Collection usuariosOnline = (ArrayList)
servletContext.getAttribute("usuariosOnline");
        usuariosOnline.remove(usuario.getLogin());
        servletContext.setAttribute("usuariosOnline", usuariosOnline);
    }
```

```
public static boolean trocaSenha(Usuario usuarioSenha, String senha) {
    try {
        if (senha != null)
            usuarioSenha.setSenha(senha);
        UsuarioDAO usuarioDAO = new UsuarioDAO();
        usuarioDAO.trocaSenha(usuarioSenha);
        return true;
    } catch (SQLException e) {
        e.printStackTrace();
        return false;
    }
}

public static Usuario setAluno(AlunoFormBean formBean) {
    Usuario usuario = new Usuario();
    usuario.setId(formBean.getId());
    usuario.setNome(formBean.getNome());
    usuario.setIdentificacao(formBean.getIdentificacao());
    usuario.setLogin(formBean.getLogin());
    usuario.setSenha(formBean.getSenha());
    usuario.setAtivo(formBean.getAtivo());

    return usuario;
}

public static Usuario getAluno(long id) {
    try {
        UsuarioDAO usuarioDAO = new UsuarioDAO();
        return usuarioDAO.carrega(id);
    } catch (SQLException e) {
        e.printStackTrace();
        return null;
    }
}

public static Usuario getAluno(String login) {
    try {
        UsuarioDAO usuarioDAO = new UsuarioDAO();
        return usuarioDAO.getUsuario(login);
    } catch (SQLException e) {
        e.printStackTrace();
        return null;
    }
}

public static void incluiAluno(Usuario aluno) {
    try {
        UsuarioDAO usuarioDAO = new UsuarioDAO();
        usuarioDAO.inclui(aluno);
    } catch (SQLException e) {
        e.printStackTrace();
    }
}

public static void alteraAluno(Usuario aluno) {
    try {
        UsuarioDAO usuarioDAO = new UsuarioDAO();
        usuarioDAO.altera(aluno);
    } catch (SQLException e) {
        e.printStackTrace();
    }
}

public static void excluiAluno(Usuario aluno) {
    try {
        UsuarioDAO usuarioDAO = new UsuarioDAO();
        usuarioDAO.exclui(aluno);
    } catch (SQLException e) {
        e.printStackTrace();
    }
}
```

```
    public static void encerraTurma() {
        try {
            UsuarioDAO usuarioDAO = new UsuarioDAO();
            usuarioDAO.encerraTurma();
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }

    public static Collection pesquisaAlunos(AlunoFormBean formBean) {
        try {
            UsuarioDAO usuarioDAO = new UsuarioDAO();
            return usuarioDAO.lista("USU_NOME,USU_IDENTIFICACAO", formBean.getNome() +
"," + formBean.getIdentificacao(), !(formBean.getInativo()));
        } catch (SQLException e) {
            e.printStackTrace();
            return null;
        }
    }
}
```

ExperimentoBO.java

```
package br.ufrj.dee.elepot.bo;

import br.ufrj.dee.elepot.dao.ExperimentoDAO;
import br.ufrj.dee.elepot.model.Experimento;
import br.ufrj.dee.elepot.struts.form.ExperimentoFormBean;
import java.sql.SQLException;
import java.util.Collection;

public final class ExperimentoBO {

    public static Experimento setExperimento(ExperimentoFormBean formBean) {
        Experimento experimento = new Experimento();
        experimento.setId(formBean.getId());
        experimento.setNumero(formBean.getNumeroInteger());
        experimento.setNome(formBean.getNome());
        experimento.setObjetivo(formBean.getObjetivo());
        experimento.setDescricao(formBean.getDescricao());
        experimento.setInstrucoes(formBean.getInstrucoes());

        return experimento;
    }

    public static Experimento getExperimento(long id) {
        try {
            ExperimentoDAO experimentoDAO = new ExperimentoDAO();
            return experimentoDAO.carrega(id);
        } catch (SQLException e) {
            e.printStackTrace();
            return null;
        }
    }

    public static Experimento getExperimento(int numero) {
        try {
            ExperimentoDAO experimentoDAO = new ExperimentoDAO();
            return experimentoDAO.getExperimento(numero);
        } catch (SQLException e) {
            e.printStackTrace();
            return null;
        }
    }

    public static void incluiExperimento(Experimento experimento) {
        try {
            ExperimentoDAO experimentoDAO = new ExperimentoDAO();
            experimentoDAO.inclui(experimento);
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
```

```
    public static void alteraExperimento(Experimento experimento) {
        try {
            ExperimentoDAO experimentoDAO = new ExperimentoDAO();
            experimentoDAO.altera(experimento);
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }

    public static void excluiExperimento(Experimento experimento) {
        try {
            ExperimentoDAO experimentoDAO = new ExperimentoDAO();
            experimentoDAO.exclui(experimento);
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }

    public static Collection listaExperimentos() {
        try {
            ExperimentoDAO experimentoDAO = new ExperimentoDAO();
            return experimentoDAO.lista();
        } catch (SQLException e) {
            e.printStackTrace();
            return null;
        }
    }

    public static Collection pesquisaExperimentos(ExperimentoFormBean formBean) {
        try {
            ExperimentoDAO experimentoDAO = new ExperimentoDAO();
            return experimentoDAO.lista("EXP_NUMERO,EXP_NOME", formBean.getNumero() +
"," + formBean.getNome());
        } catch (SQLException e) {
            e.printStackTrace();
            return null;
        }
    }
}
```

ConfiguracaoBO.java

```
package br.ufrj.dee.elepot.bo;

import br.ufrj.dee.elepot.dao.ConfiguracaoDAO;
import br.ufrj.dee.elepot.model.Configuracao;
import br.ufrj.dee.elepot.struts.form.ConfiguracaoFormBean;
import java.sql.SQLException;

public final class ConfiguracaoBO {

    public static Configuracao setConfiguracao(ConfiguracaoFormBean formBean) {
        Configuracao configuracao = new Configuracao();
        configuracao.setSistemaOperacional(formBean.getSistemaOperacional());
        configuracao.setPortaSerial(formBean.getPortaSerial());
        configuracao.setTempoSessao(formBean.getTempoSessao());

        return configuracao;
    }

    public static Configuracao getConfiguracao() {
        try {
            ConfiguracaoDAO configuracaoDAO = new ConfiguracaoDAO();
            return configuracaoDAO.carrega();
        } catch (SQLException e) {
            e.printStackTrace();
            return null;
        }
    }
```

```
    public static boolean gravaConfiguracao(Configuracao configuracao) {
        try {
            ConfiguracaoDAO configuracaoDAO = new ConfiguracaoDAO();
            configuracaoDAO.grava(configuracao);
            return true;
        } catch (SQLException e) {
            e.printStackTrace();
            return false;
        }
    }
}
```

PraticaBO.java

```
package br.ufrj.dee.elepot.bo;

import br.ufrj.dee.elepot.dao.PraticaDAO;
import br.ufrj.dee.elepot.model.Pratica;
import br.ufrj.dee.elepot.model.Usuario;
import java.sql.SQLException;

public final class PraticaBO {

    public static Pratica getPraticaEmUso() {
        try {
            PraticaDAO praticaDAO = new PraticaDAO();
            return praticaDAO.getPraticaEmUso();
        } catch (SQLException e) {
            e.printStackTrace();
            return null;
        }
    }

    public static void setPraticaEmUso(Pratica pratica) {
        try {
            PraticaDAO praticaDAO = new PraticaDAO();
            praticaDAO.setPraticaEmUso(pratica);
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }

    public static void setPraticaSemUso() {
        try {
            PraticaDAO praticaDAO = new PraticaDAO();
            praticaDAO.setPraticaSemUso();
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }

    public static void setPraticaSemUso(Usuario usuario) {
        try {
            PraticaDAO praticaDAO = new PraticaDAO();
            Pratica pratica = praticaDAO.getPraticaEmUso();
            if ((usuario != null) & (pratica != null)) {
                if (usuario.getLogin().equals(pratica.getUsuario().getLogin())) {
                    praticaDAO.setPraticaSemUso();
                }
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
}
```

RelatorioPraticaBO.java

```
package br.ufrj.dee.elepot.bo;

import br.ufrj.dee.elepot.dao.RelatorioDAO;
import java.sql.SQLException;
import java.util.Collection;
```

```
public final class RelatorioPraticaBO {
    public static Collection getRelatorioPratica(String dataInicio, String dataFim) {
        try {
            RelatorioDAO relatorioDAO = new RelatorioDAO();
            return relatorioDAO.listaRelatorioPratica(dataInicio, dataFim);
        } catch (SQLException ex) {
            ex.printStackTrace();
            return null;
        }
    }
}
```

C.4 – Pacote br.ufrj.dee.elepot.dao

UsuarioDAO.java

```
package br.ufrj.dee.elepot.dao;

import br.ufrj.dee.elepot.jdbc.ControladorDeConexoes;
import br.ufrj.dee.elepot.model.Usuario;
import br.ufrj.dee.elepot.util.Conversao;
import br.ufrj.dee.elepot.util.LoginStatus;
import br.ufrj.dee.elepot.util.Pesquisa;
import br.ufrj.dee.elepot.util.Zebrado;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.List;
import java.util.ArrayList;

public final class UsuarioDAO {
    private Connection con;

    public UsuarioDAO() throws SQLException {
        this.con = ControladorDeConexoes.getConexao();
    }

    public void inclui(Usuario usuario) throws SQLException {
        PreparedStatement stmt = this.con.prepareStatement(
            "INSERT INTO USUARIO (USU_LOGIN, USU_SENHA, USU_NOME, USU_IDENTIFICACAO,
USU_SN_ATIVO) VALUES (?, ?, ?, ?, ?)");
        stmt.setString(1, usuario.getLogin());
        stmt.setInt(2, usuario.getSenha());
        stmt.setString(3, usuario.getNome());
        stmt.setString(4, usuario.getIdentificacao());
        stmt.setString(5, "S");

        stmt.execute();
        stmt.close();
    }

    public void altera(Usuario usuario) throws SQLException {
        PreparedStatement stmt = this.con.prepareStatement(
            "UPDATE USUARIO SET USU_LOGIN=?, USU_NOME=?, USU_IDENTIFICACAO=?,
USU_SN_ATIVO=? WHERE USU_ID = ?");
        stmt.setString(1, usuario.getLogin());
        stmt.setString(2, usuario.getNome());
        stmt.setString(3, usuario.getIdentificacao());
        stmt.setString(4, Conversao.BooleanToSN(usuario.getAtivo()));
        stmt.setLong(5, usuario.getId());

        stmt.execute();
        stmt.close();
    }

    public void exclui(Usuario usuario) throws SQLException {
        this.con.setAutoCommit(false);
        PreparedStatement stmtPratica = this.con.prepareStatement(
            "DELETE FROM PRATICA WHERE PRA_USU_ID = ?");
        stmtPratica.setLong(1, usuario.getId());
        PreparedStatement stmtUsuario = this.con.prepareStatement(
            "DELETE FROM USUARIO WHERE USU_ID = ?");
        stmtUsuario.setLong(1, usuario.getId());
        try {
            try {
```

```
                stmtPratica.execute();
                stmtUsuario.execute();
                this.con.commit();
            } catch (SQLException e) {
                this.con.rollback();
            }
        } finally {
            stmtPratica.close();
            stmtUsuario.close();
            this.con.setAutoCommit(true);
        }
    }

    public Usuario carrega(Long id) throws SQLException {
        PreparedStatement stmt = this.con.prepareStatement(
            "SELECT USU_LOGIN, USU_SENHA, USU_NOME, USU_IDENTIFICACAO, USU_SN_ATIVO FROM
USUARIO WHERE USU_ID = " + id);
        ResultSet rs = stmt.executeQuery();

        Usuario usuario = new Usuario();

        while (rs.next()) {
            usuario.setId(id);
            usuario.setLogin(rs.getString("USU_LOGIN"));
            usuario.setSenha(rs.getInt("USU_SENHA"));
            usuario.setNome(rs.getString("USU_NOME"));
            usuario.setIdentificacao(rs.getString("USU_IDENTIFICACAO"));
            usuario.setAtivo(Conversao.SNToBoolean(rs.getString("USU_SN_ATIVO")));
        }

        rs.close();
        stmt.close();

        return usuario;
    }

    public List<Zebrado> lista(String filtros, String valores, boolean somenteAtivos)
throws SQLException {
        int z = 0;

        String filtroSQL = Pesquisa.FiltroSQL(filtros, valores);
        if (filtroSQL != "")
            filtroSQL = " AND " + filtroSQL;

        if (somenteAtivos == true)
            filtroSQL = filtroSQL + " AND USU_SN_ATIVO = 'S' ";

        PreparedStatement stmt = this.con.prepareStatement(
            "SELECT USU_ID, USU_LOGIN, USU_NOME, USU_IDENTIFICACAO, USU_SN_ATIVO FROM
USUARIO WHERE (USU_LOGIN <> 'PROFESSOR') " + filtroSQL + " ORDER BY USU_NOME");
        ResultSet rs = stmt.executeQuery();

        List<Zebrado> list = new ArrayList<Zebrado>();

        while (rs.next()) {
            Usuario usuario = new Usuario();
            usuario.setId(rs.getLong("USU_ID"));
            usuario.setLogin(rs.getString("USU_LOGIN"));
            usuario.setNome(rs.getString("USU_NOME"));
            usuario.setIdentificacao(rs.getString("USU_IDENTIFICACAO"));
            usuario.setAtivo(Conversao.SNToBoolean(rs.getString("USU_SN_ATIVO")));

            Zebrado zebrado = new Zebrado();
            zebrado.setObjeto(usuario);
            zebrado.setZebrado(z);
            list.add(zebrado);

            if (z == 0)
                z = 1;
            else
                z = 0;
        }

        rs.close();
        stmt.close();
```

```
        return list;
    }

    public Usuario getUsuario(String login) throws SQLException {
        PreparedStatement stmt = this.con.prepareStatement(
            "SELECT USU_ID, USU_SENHA, USU_NOME, USU_IDENTIFICACAO, USU_SN_ATIVO FROM
USUARIO WHERE USU_LOGIN = '" + login + "'");
        ResultSet rs = stmt.executeQuery();

        Usuario usuario = new Usuario();

        while (rs.next()) {
            usuario.setId(rs.getLong("USU_ID"));
            usuario.setLogin(login);
            usuario.setSenha(rs.getInt("USU_SENHA"));
            usuario.setNome(rs.getString("USU_NOME"));
            usuario.setIdentificacao(rs.getString("USU_IDENTIFICACAO"));
            usuario.setAtivo(Conversao.SNToBoolean(rs.getString("USU_SN_ATIVO")));
        }

        rs.close();
        stmt.close();

        return usuario;
    }

    public LoginStatus login(Usuario usuario) throws SQLException {
        LoginStatus loginStatus = new LoginStatus();
        boolean ok = false;

        PreparedStatement stmt = this.con.prepareStatement(
            "SELECT USU_ID, USU_SENHA, USU_NOME, USU_IDENTIFICACAO, USU_SN_ATIVO FROM
USUARIO WHERE USU_LOGIN = ?");
        stmt.setString(1, usuario.getLogin());
        ResultSet rs = stmt.executeQuery();

        if (rs.next()) {
            if (rs.getInt(2) == usuario.getSenha()) {
                usuario.setId(rs.getLong(1));
                usuario.setNome(rs.getString(3));
                usuario.setIdentificacao(rs.getString(4));
                if (Conversao.SNToBoolean(rs.getString(5)) == true) {
                    loginStatus.setLoginValido();
                } else {
                    loginStatus.setUsuarioInativo();
                }
            } else {
                loginStatus.setSenhaInvalida();
            }
        } else {
            loginStatus.setUsuarioInexistente();
        }

        rs.close();
        stmt.close();

        return loginStatus;
    }

    public void trocaSenha(Usuario usuario) throws SQLException {
        PreparedStatement stmt = this.con.prepareStatement(
            "UPDATE USUARIO SET USU_SENHA = ? WHERE USU_ID = ?");
        stmt.setInt(1, usuario.getSenha());
        stmt.setLong(2, usuario.getId());

        stmt.execute();
        stmt.close();
    }

    public void encerraTurma() throws SQLException {
        PreparedStatement stmt = this.con.prepareStatement(
            "UPDATE USUARIO SET USU_SN_ATIVO = 'N' WHERE USU_LOGIN <> 'PROFESSOR'");

        stmt.execute();
        stmt.close();
    }
```

```
    public boolean loginDuplicado(Usuario usuario) throws SQLException {
        boolean duplicado;
        PreparedStatement stmt = this.con.prepareStatement(
            "SELECT USU_ID FROM USUARIO WHERE USU_LOGIN = ? AND USU_ID <> ?");
        stmt.setString(1, usuario.getLogin());
        stmt.setLong(2, usuario.getId());
        ResultSet rs = stmt.executeQuery();
        duplicado = rs.next();

        rs.close();
        stmt.close();

        return duplicado;
    }
}
```

ExperimentoDAO.java

```
package br.ufrj.dee.elepot.dao;

import br.ufrj.dee.elepot.jdbc.ControladorDeConexoes;
import br.ufrj.dee.elepot.model.Experimento;
import br.ufrj.dee.elepot.util.Pesquisa;
import br.ufrj.dee.elepot.util.Zebrado;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.List;
import java.util.ArrayList;

public final class ExperimentoDAO {
    private Connection con;

    public ExperimentoDAO() throws SQLException {
        this.con = ControladorDeConexoes.getConexao();
    }

    public void inclui(Experimento experiemento) throws SQLException {
        PreparedStatement stmt = this.con.prepareStatement(
            "INSERT INTO EXPERIMENTO (EXP_NUMERO, EXP_NOME, EXP_OBJETIVO, EXP_DESCRICAO,
EXP_INSTRUCOES) VALUES (?, ?, ?, ?, ?)");
        stmt.setInt(1, experiemento.getNumero());
        stmt.setString(2, experiemento.getNome());
        stmt.setString(3, experiemento.getObjetivo());
        stmt.setString(4, experiemento.getDescricao());
        stmt.setString(5, experiemento.getInstrucoes());

        stmt.execute();
        stmt.close();
    }

    public void altera(Experimento experimento) throws SQLException {
        PreparedStatement stmt = this.con.prepareStatement(
            "UPDATE EXPERIMENTO SET EXP_NUMERO=?, EXP_NOME=?, EXP_OBJETIVO=?,
EXP_DESCRICAO=?, EXP_INSTRUCOES=? WHERE EXP_ID = ?");
        stmt.setInt(1, experimento.getNumero());
        stmt.setString(2, experimento.getNome());
        stmt.setString(3, experimento.getObjetivo());
        stmt.setString(4, experimento.getDescricao());
        stmt.setString(5, experimento.getInstrucoes());
        stmt.setLong(6, experimento.getId());

        stmt.execute();
        stmt.close();
    }

    public void exclui(Experimento experimento) throws SQLException {
        this.con.setAutoCommit(false);
        PreparedStatement stmtPratica = this.con.prepareStatement(
            "DELETE FROM PRATICA WHERE PRA_EXP_ID = ?");
        stmtPratica.setLong(1, experimento.getId());
        PreparedStatement stmtExperimento = this.con.prepareStatement(
            "DELETE FROM EXPERIMENTO WHERE EXP_ID = ?");
        stmtExperimento.setLong(1, experimento.getId());
        try {
```

```
            try {
                stmtPratica.execute();
                stmtExperimento.execute();
                this.con.commit();
            } catch (SQLException e) {
                this.con.rollback();
            }
        } finally {
            stmtPratica.close();
            stmtExperimento.close();
            this.con.setAutoCommit(true);
        }
    }

    public Experimento carrega(Long id) throws SQLException {
        PreparedStatement stmt = this.con.prepareStatement(
            "SELECT EXP_NUMERO, EXP_NOME, EXP_OBJETIVO, EXP_DESCRICAO, EXP_INSTRUCOES
FROM EXPERIMENTO WHERE EXP_ID = " + id);
        ResultSet rs = stmt.executeQuery();

        Experimento experimento = new Experimento();

        while (rs.next()) {
            experimento.setId(id);
            experimento.setNumero(rs.getInt("EXP_NUMERO"));
            experimento.setNome(rs.getString("EXP_NOME"));
            experimento.setObjetivo(rs.getString("EXP_OBJETIVO"));
            experimento.setDescricao(rs.getString("EXP_DESCRICAO"));
            experimento.setInstrucoes(rs.getString("EXP_INSTRUCOES"));
        }

        rs.close();
        stmt.close();

        return experimento;
    }

    public List<Zebrado> lista(String filtros, String valores) throws SQLException {
        int z = 0;

        String filtroSQL = Pesquisa.FiltroSQL(filtros, valores);
        if (filtroSQL != "")
            filtroSQL = "WHERE " + filtroSQL;

        PreparedStatement stmt = this.con.prepareStatement(
            "SELECT EXP_ID, EXP_NUMERO, EXP_NOME FROM EXPERIMENTO " + filtroSQL + "
ORDER BY EXP_NUMERO");
        ResultSet rs = stmt.executeQuery();

        List<Zebrado> list = new ArrayList<Zebrado>();

        while (rs.next()) {
            Experimento experimento = new Experimento();
            experimento.setId(rs.getLong("EXP_ID"));
            experimento.setNumero(rs.getInt("EXP_NUMERO"));
            experimento.setNome(rs.getString("EXP_NOME"));

            Zebrado zebrado = new Zebrado();
            zebrado.setObjeto(experimento);
            zebrado.setZebrado(z);
            list.add(zebrado);

            if (z == 0)
                z = 1;
            else
                z = 0;
        }

        rs.close();
        stmt.close();

        return list;
    }
```

```
    public List<Experimento> lista() throws SQLException {
        PreparedStatement stmt = this.con.prepareStatement(
            "SELECT EXP_NUMERO, EXP_NOME, EXP_OBJETIVO FROM EXPERIMENTO ORDER BY
EXP_NUMERO");
        ResultSet rs = stmt.executeQuery();

        List<Experimento> list = new ArrayList<Experimento>();

        while (rs.next()) {
            Experimento experimento = new Experimento();
            experimento.setNumero(rs.getInt("EXP_NUMERO"));
            experimento.setNome(rs.getString("EXP_NOME"));
            experimento.setObjetivo(rs.getString("EXP_OBJETIVO"));

            list.add(experimento);
        }

        rs.close();
        stmt.close();

        return list;
    }

    public Experimento getExperimento(int numero) throws SQLException {
        Experimento experimento = new Experimento();

        PreparedStatement stmt = this.con.prepareStatement(
            "SELECT EXP_ID, EXP_NOME, EXP_OBJETIVO, EXP_DESCRICAO, EXP_INSTRUCOES FROM
EXPERIMENTO WHERE EXP_NUMERO = ?");
        stmt.setInt(1, numero);
        ResultSet rs = stmt.executeQuery();

        if (rs.next()) {
            experimento.setNumero(numero);
            experimento.setId(rs.getLong(1));
            experimento.setNome(rs.getString(2));
            experimento.setObjetivo(rs.getString(3));
            experimento.setDescricao(rs.getString(4));
            experimento.setInstrucoes(rs.getString(5));
        }

        rs.close();
        stmt.close();

        return experimento;
    }
}
```

ConfiguracaoDAO.java

```
package br.ufrj.dee.elepot.dao;

import br.ufrj.dee.elepot.jdbc.ControladorDeConexoes;
import br.ufrj.dee.elepot.model.Configuracao;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;

public final class ConfiguracaoDAO {
    private Connection con;

    public ConfiguracaoDAO() throws SQLException {
        this.con = ControladorDeConexoes.getConexao();
    }

    public Configuracao carrega() throws SQLException {
        Configuracao configuracao = new Configuracao();

        PreparedStatement stmt = this.con.prepareStatement(
            "SELECT CFG_SIST_OPER, CFG_PORTA_SERIAL, CFG_TEMPO_SESSAO FROM
CONFIGURACAO");
        ResultSet rs = stmt.executeQuery();
```

```
        if (rs.next()) {
            configuracao.setSistemaOperacional(rs.getString("CFG_SIST_OPER"));
            configuracao.setPortaSerial(rs.getString("CFG_PORTA_SERIAL"));
            configuracao.setTempoSessao(rs.getInt("CFG_TEMPO_SESSAO"));
        }

        rs.close();
        stmt.close();

        return configuracao;
    }

    public void grava(Configuracao configuracao) throws SQLException {
        PreparedStatement stmt = this.con.prepareStatement(
            "UPDATE CONFIGURACAO SET CFG_SIST_OPER = ?, CFG_PORTA_SERIAL = ?,
CFG_TEMPO_SESSAO = ?");
        stmt.setString(1, configuracao.getSistemaOperacional());
        stmt.setString(2, configuracao.getPortaSerial());
        stmt.setInt(3, configuracao.getTempoSessao());

        stmt.execute();
        stmt.close();
    }
}
```

PraticaDAO.java

```
package br.ufrj.dee.elepot.dao;

import br.ufrj.dee.elepot.jdbc.ControladorDeConexoes;
import br.ufrj.dee.elepot.model.Pratica;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;

public final class PraticaDAO {
    private Connection con;

    public PraticaDAO() throws SQLException {
        this.con = ControladorDeConexoes.getConexao();
    }

    public void setPraticaEmUso(Pratica pratica) throws SQLException {
        PreparedStatement stmt = this.con.prepareStatement(
            "INSERT INTO PRATICA (PRA_EXP_ID, PRA_USU_ID, PRA_SN_USO, PRA_DATA) VALUES
(?, ?, ?, CURDATE())");
        stmt.setLong(1, pratica.getExperimento().getId());
        stmt.setLong(2, pratica.getUsuario().getId());
        stmt.setString(3, "S");


        stmt.execute();
        stmt.close();
    }

    public void setPraticaSemUso() throws SQLException {
        PreparedStatement stmt = this.con.prepareStatement(
            "UPDATE PRATICA SET PRA_SN_USO = 'N'");

        stmt.execute();
        stmt.close();
    }

    public Pratica getPraticaEmUso() throws SQLException {
        PreparedStatement stmt = this.con.prepareStatement(
            "SELECT PRA_ID, PRA_EXP_ID, PRA_USU_ID, PRA_DATA FROM PRATICA WHERE
PRA_SN_USO = 'S' AND PRA_DATA = CURDATE()");
        ResultSet rs = stmt.executeQuery();

        Pratica pratica = new Pratica();
```

```
        if (rs.next()) {
            pratica.setId(rs.getLong(1));
            pratica.setExperimento((new ExperimentoDAO()).carrega(rs.getLong(2)));
            pratica.setUsuario((new UsuarioDAO()).carrega(rs.getLong(3)));
            pratica.setEmUso(true);
            pratica.setData(rs.getDate(4));
        } else {
            pratica = null;
        }

        rs.close();
        stmt.close();

        return pratica;
    }
}
```

RelatorioDAO.java

```
package br.ufrj.dee.elepot.dao;

import br.ufrj.dee.elepot.jdbc.ControladorDeConexoes;
import br.ufrj.dee.elepot.model.report.RelatorioPratica;
import br.ufrj.dee.elepot.util.Zebrado;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.List;

public final class RelatorioDAO {
    private Connection con;

    public RelatorioDAO() throws SQLException {
        this.con = ControladorDeConexoes.getConexao();
    }

    public List<Zebrado> listaRelatorioPratica(String dataInicio, String dataFim) throws
SQLException {
        int z = 0;

        PreparedStatement stmt = this.con.prepareStatement(
            "SELECT EXP_NUMERO, EXP_NOME, USU_NOME, USU_IDENTIFICACAO, PRA_DATA " +
            "FROM PRATICA" +
            "     INNER JOIN USUARIO ON USU_ID = PRA_USU_ID" +
            "     INNER JOIN EXPERIMENTO ON EXP_ID = PRA_EXP_ID " +
            "WHERE PRA_DATA >= ? AND PRA_DATA <= ? " +
            "ORDER BY PRA_DATA, EXP_NUMERO, USU_NOME");
        stmt.setString(1, dataInicio.substring(6) + dataInicio.substring(3, 5) +
dataInicio.substring(0, 2));
        stmt.setString(2, dataFim.substring(6) + dataFim.substring(3, 5) +
dataFim.substring(0, 2));
        ResultSet rs = stmt.executeQuery();

        List<Zebrado> list = new ArrayList<Zebrado>();

        while (rs.next()) {
            RelatorioPratica relatorioPratica = new RelatorioPratica();
            relatorioPratica.setNumeroExperimento(rs.getString("EXP_NUMERO"));
            relatorioPratica.setNomeExperimento(rs.getString("EXP_NOME"));
            relatorioPratica.setNomeUsuario(rs.getString("USU_NOME"));
            relatorioPratica.setIdentificacaoUsuario(rs.getString("USU_IDENTIFICACAO"));
            relatorioPratica.setData(rs.getDate("PRA_DATA"));

            Zebrado zebrado = new Zebrado();
            zebrado.setObjeto(relatorioPratica);
            zebrado.setZebrado(z);
            list.add(zebrado);

            if (z == 0)
                z = 1;
            else
                z = 0;
        }
```

```
        rs.close();
        stmt.close();

        return list;
    }
}
```

C.5 – Pacote br.ufrj.dee.elepot.jdbc

ControladorDeConexoes.java

```
package br.ufrj.dee.elepot.jdbc;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

public final class ControladorDeConexoes {
    private static final String URL = "jdbc:mysql://localhost/ELEPOT";
    private static final String DRIVER = "com.mysql.jdbc.Driver";
    private static final String USUARIO = "elepot_web";
    private static final String SENHA = "dfr457g";

    public static Connection getConexao() throws SQLException {
        try {
            Class.forName(DRIVER);
            return DriverManager.getConnection(URL, USUARIO, SENHA);
        } catch (ClassNotFoundException e) {
            throw new SQLException(e.getMessage());
        }
    }
}
```

C.6 – Pacote br.ufrj.dee.elepot.model

Usuario.java

```
package br.ufrj.dee.elepot.model;

public final class Usuario {
    private Long id;
    private String login;
    private int senha;
    private String nome;
    private String identificacao;
    private Boolean ativo;

    public Long getId() {
        return this.id;
    }

    public void setId(Long valor) {
        this.id = valor;
    }
    public String getLogin() {
        return this.login;
    }

    public void setLogin(String valor) {
        this.login = valor;
    }

    public int getSenha() {
        return this.senha;
    }

    public void setSenha(String valor) {
        this.senha = valor.hashCode();
    }

    public void setSenha(int valor) {
        this.senha = valor;
    }
```

```
    public String getNome() {
        return this.nome;
    }

    public void setNome(String valor) {
        this.nome = valor;
    }

    public String getIdentificacao() {
        return this.identificacao;
    }

    public void setIdentificacao(String valor) {
        this.identificacao = valor;
    }

    public Boolean getAtivo() {
        return this.ativo;
    }

    public void setAtivo(Boolean valor) {
        this.ativo = valor;
    }
}
```

Experimento.java

```
package br.ufrj.dee.elepot.model;

public final class Experimento {
    private Long id;
    private int numero;
    private String nome;
    private String objetivo;
    private String descricao;
    private String instrucoes;

    public Long getId() {
        return this.id;
    }

    public void setId(Long valor) {
        this.id = valor;
    }

    public int getNumero() {
        return this.numero;
    }

    public void setNumero(int valor) {
        this.numero = valor;
    }

    public String getNome() {
        return this.nome;
    }

    public void setNome(String valor) {
        this.nome = valor;
    }

    public String getObjetivo() {
        return this.objetivo;
    }

    public void setObjetivo(String valor) {
        this.objetivo = valor;
    }

    public String getDescricao() {
        return this.descricao;
    }

    public void setDescricao(String valor) {
        this.descricao = valor;
    }
```

```
    public String getInstrucoes() {
        return this.instrucoes;
    }

    public void setInstrucoes(String valor) {
        this.instrucoes = valor;
    }
}
```

Configuracao.java

```
package br.ufrj.dee.elepot.model;

public final class Configuracao {
    private String sistemaOperacional;
    private String portaSerial;
    private int tempoSessao;

    public String getSistemaOperacional() {
        return this.sistemaOperacional;
    }

    public void setSistemaOperacional(String valor) {
        this.sistemaOperacional = valor;
    }

    public String getPortaSerial() {
        return this.portaSerial;
    }

    public void setPortaSerial(String valor) {
        this.portaSerial = valor;
    }

    public int getTempoSessao() {
        return this.tempoSessao;
    }

    public void setTempoSessao(int valor) {
        this.tempoSessao = valor;
    }
}
```

Pratica.java

```
package br.ufrj.dee.elepot.model;

import java.util.Date;

public final class Pratica {
    private Long id;
    private Experimento experimento;
    private Usuario usuario;
    private boolean emUso;
    private Date data;

    public Long getId() {
        return this.id;
    }

    public void setId(Long valor) {
        this.id = valor;
    }

    public Experimento getExperimento() {
        return this.experimento;
    }

    public void setExperimento(Experimento valor) {
        this.experimento = valor;
    }

    public Usuario getUsuario() {
        return this.usuario;
    }
```

```
    public void setUsuario(Usuario valor) {
        this.usuario = valor;
    }

    public boolean getEmUso() {
        return this.emUso;
    }

    public void setEmUso(boolean valor) {
        this.emUso = valor;
    }

    public Date getData() {
        return this.data;
    }

    public void setData(Date valor) {
        this.data = valor;
    }
}
```

C.7 – Pacote br.ufrj.dee.elepot.model.report

RelatorioPratica.java

```
package br.ufrj.dee.elepot.model.report;

import br.ufrj.dee.elepot.util.Conversao;
import java.util.Date;

public final class RelatorioPratica {
    private String numeroExperimento;
    private String nomeExperimento;
    private String nomeUsuario;
    private String identificacaoUsuario;
    private Date data;

    public String getNumeroExperimento() {
        return this.numeroExperimento;
    }

    public void setNumeroExperimento(String valor) {
        this.numeroExperimento = valor;
    }

    public String getNomeExperimento() {
        return this.nomeExperimento;
    }

    public void setNomeExperimento(String valor) {
        this.nomeExperimento = valor;
    }

    public String getNomeUsuario() {
        return this.nomeUsuario;
    }
    public void setNomeUsuario(String valor) {
        this.nomeUsuario = valor;
    }

    public String getIdentificacaoUsuario() {
        return this.identificacaoUsuario;
    }

    public void setIdentificacaoUsuario(String valor) {
        this.identificacaoUsuario = valor;
    }

    public String getData() {
        return Conversao.DateToString(this.data, Conversao.FORMATO_DD_MM_YYYY);
    }

    public void setData(Date valor) {
        this.data = valor;
    }
}
```

C.8 – Pacote br.ufrj.dee.elepot.struts

ElepotServlet.java

```
package br.ufrj.dee.elepot.struts;

import br.ufrj.dee.elepot.bo.PraticaBO;
import java.util.ArrayList;
import java.util.Collection;
import javax.servlet.ServletConfig;
import javax.servlet.ServletException;
import org.apache.struts.action.ActionServlet;

public final class ElepotServlet extends ActionServlet {

    public void init(ServletConfig config) throws ServletException {
        super.init(config);

        PraticaBO.setPraticaSemUso();

        Collection usuariosOnline = new ArrayList();
        config.getServletContext().setAttribute("usuariosOnline", usuariosOnline);
    }
}
```

ApplicationResource.properties

```
!-- MENSAGEMS CONTIDAS NOS ARQUIVOS JSP E HTML
cabecalho.titulo=Laboratório de Eletrônica de Potência
cabecalho.ufrj=UFRJ - Universidade Federal do Rio de Janeiro
cabecalho.ufrj.ee=Centro de Tecnologia/Escola Politécnica
cabecalho.ufrj.ee.dee=Departamento de Engenharia Elétrica

rodape.elepot1=Departamento de Engenharia Elétrica (DEE / UFRJ) &mdash; dezembro de 2007
rodape.elepot2=Projeto final de graduação em Engenharia Elétrica (Daniel Santos de
Oliveira Cabral)

titulo.elepot=Laboratório de Eletrônica de Potência da UFRJ
titulo.elepot.bemvindo=Seja bem vindo ao Laboratório de Eletrônica de Potência da UFRJ

erro.atencao=ATENÇÃO:
erro.mensagem=Ocorreu um erro no sistema ao tentar executar esta operação.
erro.aviso=Procure o suporte técnico ou tente novamente mais tarde.
erro.voltar=(voltar para o início)
erro.ocupado.mensagem=No momento o laboratório está em uso.
erro.ocupado.aviso=Tente novamente mais tarde.
erro.ocupado.voltar=(voltar)

login.introducao.linha1=Esta aplicação fornece acesso aos módulos de controle e
administração da bancada de eletrônica de potência microprocessada. Os alunos deverão
solicitar suas senhas ao professor do laboratório. Este laboratório foi desenvolvido
durante um projeto final do curso de graduação em Engenharia Elétrica sob orientação do
professor Dr.-Ing. Luís Guilherme Barbosa Rolim.
login.introducao.linha2=Informações importantes para a utilização do sistema:
login.introducao.linha3=O navegador web deverá estar configurado para aceitar cookies
login.introducao.linha4=É recomendável a utilização de resolução de vídeo de 800x600
pontos ou superior
login.introducao.linha5=Para uma melhor navegabilidade utilize preferencialmente os
links da aplicação ao invés dos botões Voltar e Avançar do navegador web
login.erro.usuarioinexistente=ATENÇÃO: o usuário informado não existe! Tente novamente.
login.erro.senhainvalida=ATENÇÃO: a senha informada é inválida! Tente novamente.
login.erro.usuarioinativo=ATENÇÃO: este usuário está inativo no laboratório. Procure o
professor.
login.erro.logininvalido=ATENÇÃO: não foi possível executar o login. Tente novamente
mais tarde.
login.erro.loginduplicado=ATENÇÃO: este usuário já está com sua conta aberta no sistema.
login.laboratorio.titulo=Acessar o laboratório
login.laboratorio.titulo.obs=(alunos e professores)
login.laboratorio.form.usuario=Usuário:
login.laboratorio.form.senha=Senha:
login.laboratorio.form.entrar=Entrar
login.manutencao.titulo=Manutenção do sistema
login.manutencao.titulo.obs=(somente professores)
login.manutencao.form.senha=Senha:
login.manutencao.form.entrar=Entrar
login.links=Links úteis:
login.links.elepotgraduacao=Elepot/Graduação
```

```
login.links.elepotgraduacao.descricao=Página da disciplina "Laboratório de Eletrônica de
Potência" disponibilizada pelo DEE/UFRJ.
login.links.elepotcoppe=Elepot/COPPE-UFRJ
login.links.elepotcoppe.descricao=Página do Laboratório de Eletrônica de Potência da
COPPE/UFRJ.

laboratorio.titulo=Experiências:
laboratorio.trocasenha=Troca de senha
laboratorio.experiencia=Experiência nº
laboratorio.objetivo=Objetivo:
laboratorio.voltar=Voltar
laboratorio.popup.descricao=Descrição
laboratorio.popup.instrucoes=Instruções
laboratorio.enviarcomando=Enviar comando

laboratorio.experiencias.titulo=Experiência
laboratorio.experiencias.aluno=Aluno:
laboratorio.experiencias.paramentrosdoconversor.titulo=Outros parâmetros do conversor:
laboratorio.experiencias.paramentrosdoconversor.ts=Ts
laboratorio.experiencias.paramentrosdoconversor.ton=Ton
laboratorio.experiencias.paramentrosdoconversor.toff=Toff
laboratorio.experiencias.paramentrosdoconversor.vovi=Vo / Vi
laboratorio.experiencias.paramentrosdoconversor.vo=Vo
laboratorio.experiencias.paramentrosdoconversor.mf=índice mod. freq. (mf)
laboratorio.experiencias.paramentrosdoconversor.ma=índice mod. tensão (ma)
laboratorio.experiencias.comando=Comando:
laboratorio.experiencias.comando.ligar=Ligar
laboratorio.experiencias.comando.desligar=Desligar
laboratorio.experiencias.tipodeconversor=Tipo de conversor:
laboratorio.experiencias.tipodeconversor.buck=Buck
laboratorio.experiencias.tipodeconversor.boost=Boost
laboratorio.experiencias.frequenciachaveamento=Frequência de chaveamento:
laboratorio.experiencias.frequenciachaveamento.intervalo.chopper=(min. 30 Hz / máx. 2
kHz)
laboratorio.experiencias.frequenciachaveamento.intervalo.inversor=(min. 360 Hz / máx. 4
kHz)
laboratorio.experiencias.ciclodetrabalho=Ciclo de trabalho:
laboratorio.experiencias.frequenciareferencia=Frequência de referência:
laboratorio.experiencias.frequenciareferencia.maximo=(máximo 180 Hz)
laboratorio.experiencias.tensaoreferencia=Tensão de referência:
laboratorio.experiencias.tensaoreferencia.maximo=(máximo 2 pu)
laboratorio.experiencias.angulodisparo=Ângulo de disparo:
laboratorio.experiencias.angulodisparo.intervalo=(de 0° a 180°)
laboratorio.experiencias.frequenciarede=Frequência da rede:
laboratorio.experiencias.unidades.ms=ms
laboratorio.experiencias.unidades.percentual=%
laboratorio.experiencias.unidades.hz=Hz
laboratorio.experiencias.unidades.pu=pu
laboratorio.experiencias.unidades.graus=°

manutencao.titulo=Manutenção do Sistema
manutencao.descricao=Através deste módulo o professor do Laboratório de Eletrônica de
Potência pode cadastrar alunos, práticas de laboratório e trocar sua senha.
manutencao.cadastro.alunos=Cadastro de alunos
manutencao.relatorio.pratica=Relatório de práticas realizadas na bancada didática
manutencao.encerraturma=Encerrar turma (fim de período letivo)
manutencao.cadastro.experimentos=Cadastro de práticas de laboratório
manutencao.configuracao=Configuração da bancada didática
manutencao.trocasenha=Troca de senha do professor
manutencao.voltar=Voltar

cadastro.alunos.titulo=Cadastro de alunos
cadastro.alunos.form.titulo=Pesquisar por:
cadastro.alunos.form.nome=Nome:
cadastro.alunos.form.identificacao=Identificação:
cadastro.alunos.form.login=Login:
cadastro.alunos.form.senha=Senha:
cadastro.alunos.form.ativo=Ativo:
cadastro.alunos.form.exibirinativos=Exibir os alunos inativos
cadastro.alunos.form.pesquisar=Pesquisar
cadastro.alunos.form.incluir=Incluir
cadastro.alunos.form.alterar=Alterar
cadastro.alunos.form.excluir=Excluir
cadastro.alunos.form.gravar=Gravar
cadastro.alunos.form.retornar=Retornar
cadastro.alunos.novasenha=A nova senha deste aluno é
cadastro.alunos.novasenha.observacao=IMPORTANTE: copie esta senha e entregue ao aluno
```

```
cadastro.alunos.ok.incluido=Aluno incluído com sucesso.
cadastro.alunos.ok.alterado=Aluno alterado com sucesso.
cadastro.alunos.ok.excluido=Aluno excluído com sucesso.
cadastro.alunos.erro.duplicado=Este login já pertence a outro aluno e não poderá ser
utilizado.

relatorio.pratica.titulo=Relatório de práticas realizadas
relatorio.pratica.form.titulo=Filtrar por:
relatorio.pratica.form.periodo=Período de
relatorio.pratica.form.ate=até
relatorio.pratica.form.formatodata=(escreva a data no formato dd/mm/aaaa)
relatorio.pratica.form.consultar=Consultar
relatorio.pratica.resultado.experimento=Prática
relatorio.pratica.resultado.aluno=Aluno
relatorio.pratica.resultado.data=Data
relatorio.pratica.retornar=Retornar
relatorio.totalderegistros=Total de registros:

encerraturma.titulo=Encerrar turma
encerraturma.ok=Turma encerrada com sucesso (todos os alunos estão inativos).
encerraturma.voltar=Voltar

cadastro.experimentos.titulo=Cadastro de práticas de laboratório
cadastro.experimentos.observacao=IMPORTANTE: para incluir novos experimentos ou mesmo
fazer alterações significativas (utilização de um novo conversor por exemplo) será
necessário, além deste cadastro, a disponibilização de novos arquivos no servidor.
Consulte o manual do professor para maiores esclarecimentos.
cadastro.experimentos.form.titulo=Pesquisar por:
cadastro.experimentos.form.numero=Número:
cadastro.experimentos.form.nome=Nome:
cadastro.experimentos.form.objetivo=Objetivo:
cadastro.experimentos.form.descricao=Descrição:
cadastro.experimentos.form.instrucoes=Instruções:
cadastro.experimentos.form.pesquisar=Pesquisar
cadastro.experimentos.form.incluir=Incluir
cadastro.experimentos.form.alterar=Alterar
cadastro.experimentos.form.excluir=Excluir
cadastro.experimentos.form.gravar=Gravar
cadastro.experimentos.form.retornar=Retornar
cadastro.experimentos.ok.incluido=Prática de laboratório incluída com sucesso.
cadastro.experimentos.ok.alterado=Prática de laboratório alterada com sucesso.
cadastro.experimentos.ok.excluido=Prática de laboratório excluída com sucesso.

configuracao.titulo=Configuração da bancada didática
configuracao.form.sistemaoperacional=Sistema operacional:
configuracao.form.portaserial=Porta serial:
configuracao.form.temposessao=Tempo da sessão (min/aluno):
configuracao.form.gravar=Gravar
configuracao.form.retornar=Retornar
configuracao.ok=Configurações gravadas com sucesso.
configuracao.voltar=Voltar

trocasenha.titulo=Troca de senha
trocasenha.form.novasenha=Nova senha:
trocasenha.form.redigitesenha=Redigite a nova senha:
trocasenha.form.trocar=Trocar a senha
trocasenha.form.retornar=Retornar
trocasenha.ok=Senha trocada com sucesso.
trocasenha.voltar=Voltar
```

C.9 – Pacote br.ufrj.dee.elepot.struts.action

AlteraAluno.java

```
package br.ufrj.dee.elepot.struts.action;

import br.ufrj.dee.elepot.bo.UsuarioBO;
import br.ufrj.dee.elepot.struts.form.AlunoFormBean;
import br.ufrj.dee.elepot.model.Usuario;
import javax.servlet.http.*;
import org.apache.struts.action.*;

public final class AlteraAluno extends Action {
```

```
    public ActionForward execute (ActionMapping map, ActionForm form, HttpServletRequest
request, HttpServletResponse response) throws Exception {
        AlunoFormBean formulario = (AlunoFormBean) form;

        if (request.getParameter("tela").equals("preparar")) {
            request.setAttribute("aluno", UsuarioBO.getAluno(formulario.getId()));

            return map.findForward("tela");
        } else {
            Usuario aluno = UsuarioBO.setAluno(formulario);

            if (!UsuarioBO.isLoginDuplicado(aluno)) {
                UsuarioBO.alteraAluno(aluno);
                if (!formulario.getSenha().equals(""))
                    UsuarioBO.trocaSenha(aluno, null);
                formulario.resetAtivo();
                request.setAttribute("novaSenha", "");
                request.setAttribute("tela", "alteracao");

                return map.findForward("ok");
            } else {
                request.setAttribute("tela", "duplicado");
                request.setAttribute("aluno", aluno);

                return map.findForward("duplicado");
            }
        }
    }
}
```

AlteraExperimento.java

```
package br.ufrj.dee.elepot.struts.action;

import br.ufrj.dee.elepot.bo.ExperimentoBO;
import br.ufrj.dee.elepot.struts.form.ExperimentoFormBean;
import br.ufrj.dee.elepot.model.Experimento;
import javax.servlet.http.*;
import org.apache.struts.action.*;

public final class AlteraExperimento extends Action {

    public ActionForward execute (ActionMapping map, ActionForm form, HttpServletRequest
request, HttpServletResponse response) throws Exception {
        ExperimentoFormBean formulario = (ExperimentoFormBean) form;

        if (request.getParameter("tela").equals("preparar")) {
            request.setAttribute("experimento",
ExperimentoBO.getExperimento(formulario.getId()));

            return map.findForward("tela");
        } else {
            Experimento experimento = ExperimentoBO.setExperimento(formulario);



            ExperimentoBO.alteraExperimento(experimento);
            request.setAttribute("tela", "alteracao");

            return map.findForward("ok");
        }
    }
}
```

CarregaConfiguracao.java

```
package br.ufrj.dee.elepot.struts.action;

import br.ufrj.dee.elepot.bo.ConfiguracaoBO;
import br.ufrj.dee.elepot.model.Configuracao;
import javax.servlet.http.*;
import org.apache.struts.action.*;

public final class CarregaConfiguracao extends Action {
```

```
    public ActionForward execute (ActionMapping map, ActionForm form, HttpServletRequest
request, HttpServletResponse response) throws Exception {
        Configuracao configuracao = ConfiguracaoBO.getConfiguracao();
        request.setAttribute("configuracao", configuracao);

        return map.findForward("ok");
    }
}
```

ComandaChopper.java

```
package br.ufrj.dee.elepot.struts.action;

import br.ufrj.dee.elepot.Chopper;
import javax.servlet.http.*;
import org.apache.struts.action.*;
import br.ufrj.dee.elepot.struts.form.ChopperFormBean;

public final class ComandaChopper extends Action {

    public ActionForward execute (ActionMapping map, ActionForm form, HttpServletRequest
request, HttpServletResponse response) throws Exception {
        ChopperFormBean formulario = (ChopperFormBean) form;

        Chopper chopper = new Chopper(formulario.getTipoChopper());
        if (formulario.getLigado()) {
            chopper.setFs(formulario.getFs());
            chopper.setDc(formulario.getDc());
            chopper.setChopper(Chopper.CHOPPER_COMANDO_LIGAR);
            request.setAttribute("parametros", chopper);
            formulario.setDc(chopper.getDcString());
        } else {
            chopper.setChopper(Chopper.CHOPPER_COMANDO_DESLIGAR);
            formulario.setFs("");
            formulario.setDc("");
            formulario.setTipoChopper(null);
        }

        request.setAttribute("tituloExperiencia", formulario.getTituloExperiencia());
        request.setAttribute("nomeAluno", formulario.getNomeAluno());

        return map.findForward("ok");
    }
}
```

ComandaInversor.java

```
package br.ufrj.dee.elepot.struts.action;

import br.ufrj.dee.elepot.Inversor;
import javax.servlet.http.*;
import org.apache.struts.action.*;
import br.ufrj.dee.elepot.struts.form.InversorFormBean;

public final class ComandaInversor extends Action {

    public ActionForward execute (ActionMapping map, ActionForm form, HttpServletRequest
request, HttpServletResponse response) throws Exception {
        InversorFormBean formulario = (InversorFormBean) form;

        Inversor inversor = new Inversor();
        if (formulario.getLigado()) {
            inversor.setFs(formulario.getFs());
            inversor.setFref(formulario.getFref());
            inversor.setVref(formulario.getVref());
            inversor.setInversor(Inversor.INVERSOR_COMANDO_LIGAR);
            request.setAttribute("parametros", inversor);
            formulario.setFref(inversor.getFrefString());
            formulario.setVref(inversor.getVrefString());
        } else {
            inversor.setInversor(Inversor.INVERSOR_COMANDO_DESLIGAR);
            formulario.setFs("");
            formulario.setFref("");
            formulario.setVref("");
        }
```

```
        request.setAttribute("tituloExperiencia", formulario.getTituloExperiencia());
        request.setAttribute("nomeAluno", formulario.getNomeAluno());

        return map.findForward("ok");
    }
}
```

ComandaRetificador.java

```
package br.ufrj.dee.elepot.struts.action;

import br.ufrj.dee.elepot.Retificador;
import javax.servlet.http.*;
import org.apache.struts.action.*;
import br.ufrj.dee.elepot.struts.form.RetificadorFormBean;

public final class ComandaRetificador extends Action {

    public ActionForward execute (ActionMapping map, ActionForm form, HttpServletRequest
request, HttpServletResponse response) throws Exception {
        RetificadorFormBean formulario = (RetificadorFormBean) form;

        Retificador retificador = new Retificador();
        if (formulario.getLigado()) {
            retificador.setAlpha(formulario.getAlpha());
            retificador.setFreq(formulario.getFreq());
            retificador.setRetificador(Retificador.RETIFICADOR_COMANDO_LIGAR);
            request.setAttribute("parametros", retificador);
            formulario.setAlpha(retificador.getAlphaString());
            formulario.setFreq(retificador.getFreqString());
        } else {
            retificador.setRetificador(Retificador.RETIFICADOR_COMANDO_DESLIGAR);
            formulario.setAlpha("");
            formulario.setFreq("60");
        }

        request.setAttribute("tituloExperiencia", formulario.getTituloExperiencia());
        request.setAttribute("nomeAluno", formulario.getNomeAluno());

        return map.findForward("ok");
    }
}
```

EncerraTurma.java

```
package br.ufrj.dee.elepot.struts.action;

import br.ufrj.dee.elepot.bo.UsuarioBO;
import javax.servlet.http.*;
import org.apache.struts.action.*;

public final class EncerraTurma extends Action {

    public ActionForward execute (ActionMapping map, ActionForm form, HttpServletRequest
request, HttpServletResponse response) throws Exception {
        UsuarioBO.encerraTurma();

        return map.findForward("ok");
    }
}
```

ExcluiAluno.java

```
package br.ufrj.dee.elepot.struts.action;

import br.ufrj.dee.elepot.bo.UsuarioBO;
import br.ufrj.dee.elepot.struts.form.AlunoFormBean;
import javax.servlet.http.*;
import org.apache.struts.action.*;

public final class ExcluiAluno extends Action {

    public ActionForward execute (ActionMapping map, ActionForm form, HttpServletRequest
request, HttpServletResponse response) throws Exception {
        AlunoFormBean formulario = (AlunoFormBean) form;
```

```
        UsuarioBO.excluiAluno(UsuarioBO.getAluno(formulario.getId()));
        request.setAttribute("novaSenha", "");
        request.setAttribute("tela", "exclusao");

        return map.findForward("ok");
    }
}
```

ExcluiExperimento.java

```
package br.ufrj.dee.elepot.struts.action;

import br.ufrj.dee.elepot.bo.ExperimentoBO;
import br.ufrj.dee.elepot.struts.form.ExperimentoFormBean;
import javax.servlet.http.*;
import org.apache.struts.action.*;

public final class ExcluiExperimento extends Action {

    public ActionForward execute (ActionMapping map, ActionForm form, HttpServletRequest
request, HttpServletResponse response) throws Exception {
        ExperimentoFormBean formulario = (ExperimentoFormBean) form;

ExperimentoBO.excluiExperimento(ExperimentoBO.getExperimento(formulario.getId()));
        request.setAttribute("tela", "exclusao");

        return map.findForward("ok");
    }
}
```

GeraRelatorioPratica.java

```
package br.ufrj.dee.elepot.struts.action;

import br.ufrj.dee.elepot.bo.RelatorioPraticaBO;
import br.ufrj.dee.elepot.struts.form.RelatorioPraticaFormBean;
import java.util.Collection;
import javax.servlet.http.*;
import org.apache.struts.action.*;

public final class GeraRelatorioPratica extends Action {

    public ActionForward execute (ActionMapping map, ActionForm form, HttpServletRequest
request, HttpServletResponse response) throws Exception {
        RelatorioPraticaFormBean formulario = (RelatorioPraticaFormBean) form;

        request.setAttribute("dataInicio", formulario.getDataInicio());
        request.setAttribute("dataFim", formulario.getDataFim());
        Collection resultado =
RelatorioPraticaBO.getRelatorioPratica(formulario.getDataInicio(),
formulario.getDataFim());
        request.setAttribute("lista", resultado);
        request.setAttribute("total", resultado.size());

        return map.findForward("ok");
    }
}
```

GravaConfiguracao.java

```
package br.ufrj.dee.elepot.struts.action;

import br.ufrj.dee.elepot.bo.ConfiguracaoBO;
import javax.servlet.http.*;
import org.apache.struts.action.*;
import br.ufrj.dee.elepot.struts.form.ConfiguracaoFormBean;

public final class GravaConfiguracao extends Action {

    public ActionForward execute (ActionMapping map, ActionForm form, HttpServletRequest
request, HttpServletResponse response) throws Exception {
        ConfiguracaoFormBean formulario = (ConfiguracaoFormBean) form;

        ConfiguracaoBO.gravaConfiguracao(ConfiguracaoBO.setConfiguracao(formulario));
```

```
        return map.findForward("ok");
    }
}
```

IncluiAluno.java

```
package br.ufrj.dee.elepot.struts.action;

import br.ufrj.dee.elepot.bo.UsuarioBO;
import br.ufrj.dee.elepot.struts.form.AlunoFormBean;
import br.ufrj.dee.elepot.model.Usuario;
import br.ufrj.dee.elepot.util.Senha;
import javax.servlet.http.*;
import org.apache.struts.action.*;

public final class IncluiAluno extends Action {

    public ActionForward execute (ActionMapping map, ActionForm form, HttpServletRequest
request, HttpServletResponse response) throws Exception {
        AlunoFormBean formulario = (AlunoFormBean) form;

        String novaSenha = Senha.geraSenha();
        formulario.setId(Long.valueOf("-1"));
        formulario.setSenha(novaSenha);

        Usuario aluno = UsuarioBO.setAluno(formulario);

        if (!UsuarioBO.isLoginDuplicado(aluno)) {
            UsuarioBO.incluiAluno(aluno);
            request.setAttribute("novaSenha", novaSenha);
            request.setAttribute("tela", "inclusao");

            return map.findForward("ok");
        } else {
            request.setAttribute("tela", "duplicado");
            request.setAttribute("formulario", formulario);

            return map.findForward("duplicado");
        }
    }
}
```

IncluiExperimento.java

```
package br.ufrj.dee.elepot.struts.action;

import br.ufrj.dee.elepot.bo.ExperimentoBO;
import br.ufrj.dee.elepot.struts.form.ExperimentoFormBean;
import br.ufrj.dee.elepot.model.Experimento;
import javax.servlet.http.*;
import org.apache.struts.action.*;

public final class IncluiExperimento extends Action {

    public ActionForward execute (ActionMapping map, ActionForm form, HttpServletRequest
request, HttpServletResponse response) throws Exception {
        ExperimentoFormBean formulario = (ExperimentoFormBean) form;

        Experimento experimento = ExperimentoBO.setExperimento(formulario);
        ExperimentoBO.incluiExperimento(experimento);
        request.setAttribute("tela", "inclusao");

        return map.findForward("ok");
    }
}
```

Login.java

```
package br.ufrj.dee.elepot.struts.action;

import br.ufrj.dee.elepot.bo.UsuarioBO;
import br.ufrj.dee.elepot.util.LoginStatus;
import javax.servlet.http.*;
import org.apache.struts.action.*;
import br.ufrj.dee.elepot.struts.form.LoginFormBean;
```

```
import br.ufrj.dee.elepot.model.Usuario;

public final class Login extends Action {

    public ActionForward execute (ActionMapping map, ActionForm form, HttpServletRequest
request, HttpServletResponse response) throws Exception {
        LoginFormBean formulario = (LoginFormBean) form;
        Usuario usuarioLogin = new Usuario();
        usuarioLogin.setLogin(formulario.getLogin());
        usuarioLogin.setSenha(formulario.getSenha());

        LoginStatus loginStatusValidade = UsuarioBO.isLoginValido(usuarioLogin);
        if (loginStatusValidade.getUsuarioInexistente()) {
            request.setAttribute("loginStatus", "usuarioinexistente");
            return map.findForward("negado");
        } else
            if (loginStatusValidade.getSenhaInvalida()) {
                request.setAttribute("loginStatus", "senhainvalida");
                return map.findForward("negado");
            } else
                if (loginStatusValidade.getUsuarioInativo()) {
                    request.setAttribute("loginStatus", "usuarioinativo");
                    return map.findForward("negado");
                } else
                    if (!loginStatusValidade.getLoginValido()) {
                        request.setAttribute("loginStatus", "logininvalido");
                        return map.findForward("negado");
                    } else {
                        HttpSession sessao = request.getSession();
                        if (sessao.getAttribute("usuario") == null) {
                            if (!usuarioLogin.getLogin().equals("professor")) {
                                LoginStatus loginStatusDuplicidade =
UsuarioBO.isLoginOnline(getServlet().getServletContext(), usuarioLogin);
                                if (loginStatusDuplicidade.getLoginDuplicado()) {
                                    request.setAttribute("loginStatus",
"logindupli cado");
                                    return map.findForward("negado");
                                } else {

UsuarioBO.setOnline(getServlet().getServletContext(), usuarioLogin);
                                    sessao.setAttribute("usuario", usuarioLogin);
                                    return map.findForward("aceito");
                                }
                            } else {
                                sessao.setAttribute("usuario", usuarioLogin);
                                return map.findForward("aceito");
                            }
                        } else {
                            return map.findForward("aceito");
                        }
                    }
    }
}
```

Logout.java

```
package br.ufrj.dee.elepot.struts.action;

import javax.servlet.http.*;
import org.apache.struts.action.*;

public final class Logout extends Action {

    public ActionForward execute (ActionMapping map, ActionForm form, HttpServletRequest
request, HttpServletResponse response) throws Exception {
        HttpSession sessao = request.getSession(false);
        sessao.invalidate();
        return map.findForward("ok");
    }
}
```

MontaLab.java

```
package br.ufrj.dee.elepot.struts.action;

import br.ufrj.dee.elepot.bo.ConfiguracaoBO;
import br.ufrj.dee.elepot.bo.ExperimentoBO;
import br.ufrj.dee.elepot.bo.PraticaBO;
import br.ufrj.dee.elepot.model.Usuario;
import javax.servlet.http.*;
import org.apache.struts.action.*;

public final class MontaLab extends Action {

    public ActionForward execute (ActionMapping map, ActionForm form, HttpServletRequest
request, HttpServletResponse response) throws Exception {
        super.execute(map, form, request, response);

        HttpSession sessao = request.getSession(false);
        if (sessao != null) {
            PraticaBO.setPraticaSemUso((Usuario) sessao.getAttribute("usuario"));
            sessao.setAttribute("ok", true);

sessao.setMaxInactiveInterval(ConfiguracaoBO.getConfiguracao().getTempoSessao() *
60000);
        }

        request.setAttribute("lista", ExperimentoBO.listaExperimentos());

        return map.findForward("ok");
    }
}
```

MontaPopup.java

```
package br.ufrj.dee.elepot.struts.action;

import br.ufrj.dee.elepot.bo.ExperimentoBO;
import br.ufrj.dee.elepot.model.Experimento;
import javax.servlet.http.*;
import org.apache.struts.action.*;

public final class MontaPopup extends Action {

    public ActionForward execute (ActionMapping map, ActionForm form, HttpServletRequest
request, HttpServletResponse response) throws Exception {
        Experimento experimento =
ExperimentoBO.getExperimento(Integer.valueOf(request.getParameter("exp")));

        if (request.getParameter("janela").equals("descricao")) {
            request.setAttribute("descricao", experimento.getDescricao());
            return map.findForward("descricao");
        } else if (request.getParameter("janela").equals("instrucoes")) {
            request.setAttribute("instrucoes", experimento.getInstrucoes());
            return map.findForward("instrucoes");
        } else {
            return map.findForward("");
        }
    }
}
```

MontaPratica.java

```
package br.ufrj.dee.elepot.struts.action;

import br.ufrj.dee.elepot.bo.ConfiguracaoBO;
import br.ufrj.dee.elepot.bo.ExperimentoBO;
import br.ufrj.dee.elepot.bo.PraticaBO;
import br.ufrj.dee.elepot.bo.UsuarioBO;
import br.ufrj.dee.elepot.model.Experimento;
import br.ufrj.dee.elepot.model.Pratica;
import br.ufrj.dee.elepot.model.Usuario;
import javax.servlet.http.*;
import org.apache.struts.action.*;

public final class MontaPratica extends Action {
```

```
    public ActionForward execute (ActionMapping map, ActionForm form, HttpServletRequest
request, HttpServletResponse response) throws Exception {
        Pratica pratica = PraticaBO.getPraticaEmUso();

        if (pratica != null) {
            return map.findForward("ocupado");
        } else {
            Experimento experimento =
ExperimentoBO.getExperimento(Integer.valueOf(request.getParameter("exp")));
            Usuario aluno = UsuarioBO.getAluno(((Usuario)
request.getSession(false).getAttribute("usuario")).getLogin());
            Pratica praticaNova = new Pratica();
            praticaNova.setExperimento(experimento);
            praticaNova.setUsuario(aluno);

            PraticaBO.setPraticaEmUso(praticaNova);

request.getSession(false).setMaxInactiveInterval(ConfiguracaoBO.getConfiguracao().getTem
poSessao() * 60000);

            request.setAttribute("tituloExperiencia", experimento.getNumero() + " - " +
experimento.getNome());
            request.setAttribute("nomeAluno", aluno.getNome());

            return map.findForward("ok");
        }
    }
}
```

PesquisaAlunos.java

```
package br.ufrj.dee.elepot.struts.action;

import br.ufrj.dee.elepot.bo.UsuarioBO;
import br.ufrj.dee.elepot.struts.form.AlunoFormBean;
import java.util.Collection;
import javax.servlet.http.*;
import org.apache.struts.action.*;

public final class PesquisaAlunos extends Action {

    public ActionForward execute (ActionMapping map, ActionForm form, HttpServletRequest
request, HttpServletResponse response) throws Exception {
        AlunoFormBean formulario = (AlunoFormBean) form;

        request.setAttribute("nome", formulario.getNome());
        request.setAttribute("identificacao", formulario.getIdentificacao());
        Collection alunos = UsuarioBO.pesquisaAlunos(formulario);
        if (formulario.getInativo() == true)
            formulario.resetInativo();

        request.setAttribute("lista", alunos);

        return map.findForward("ok");
    }
}
```

PesquisaExperimentos.java

```
package br.ufrj.dee.elepot.struts.action;

import br.ufrj.dee.elepot.bo.ExperimentoBO;
import br.ufrj.dee.elepot.struts.form.ExperimentoFormBean;
import javax.servlet.http.*;
import org.apache.struts.action.*;

public final class PesquisaExperimentos extends Action {

    public ActionForward execute (ActionMapping map, ActionForm form, HttpServletRequest
request, HttpServletResponse response) throws Exception {
        ExperimentoFormBean formulario = (ExperimentoFormBean) form;

        request.setAttribute("numero", formulario.getNumero());
        request.setAttribute("nome", formulario.getNome());
        request.setAttribute("lista", ExperimentoBO.pesquisaExperimentos(formulario));
```

```
        return map.findForward("ok");
    }
}
```

TrocaSenha.java

```
package br.ufrj.dee.elepot.struts.action;

import br.ufrj.dee.elepot.bo.UsuarioBO;
import javax.servlet.http.*;
import org.apache.struts.action.*;
import br.ufrj.dee.elepot.struts.form.TrocaSenhaFormBean;
import br.ufrj.dee.elepot.model.Usuario;

public final class TrocaSenha extends Action {

    public ActionForward execute (ActionMapping map, ActionForm form, HttpServletRequest
request, HttpServletResponse response) throws Exception {
        HttpSession sessao = request.getSession(false);

        TrocaSenhaFormBean formulario = (TrocaSenhaFormBean) form;

        if (UsuarioBO.trocaSenha((Usuario) sessao.getAttribute("usuario"),
formulario.getSenha())) {
            if (formulario.getTelaOrigem().equals("laboratorio"))
                request.setAttribute("telaOrigem", "laboratorio");
            else
                request.setAttribute("telaOrigem", "manutencao");
            return map.findForward("ok");
        } else {
            return map.findForward("erro");
        }
    }
}
```

C.10 – Pacote br.ufrj.dee.elepot.struts.form

AlunoFormBean.java

```
package br.ufrj.dee.elepot.struts.form;

import org.apache.struts.action.ActionForm;

public final class AlunoFormBean extends ActionForm {
    private Long id;
    private String nome;
    private String identificacao;
    private String login;
    private String senha;
    private boolean ativo = false;
    private boolean inativo = false;

    public void setId(Long valor) {
        this.id = valor;
    }

    public Long getId() {
        return this.id;
    }

    public void setNome(String valor) {
        this.nome = valor;
    }

    public String getNome() {
        return this.nome;
    }

    public void setIdentificacao(String valor) {
        this.identificacao = valor;
    }

    public String getIdentificacao() {
        return this.identificacao;
    }
```

```
    public void setLogin(String valor) {
        this.login = valor;
    }

    public String getLogin() {
        return this.login;
    }

    public void setSenha(String valor) {
        this.senha = valor;
    }

    public String getSenha() {
        return this.senha;
    }

    public void setAtivo(boolean ativo) {
        this.ativo = ativo;
    }

    public boolean getAtivo() {
        return this.ativo;
    }

    public void resetAtivo() {
        this.ativo = false;
    }

    public void setInativo(boolean inativo) {
        this.inativo = inativo;
    }

    public boolean getInativo() {
        return this.inativo;
    }

    public void resetInativo() {
        this.inativo = false;
    }
}
```

ExperimentoFormBean.java

```
package br.ufrj.dee.elepot.struts.form;

import org.apache.struts.action.ActionForm;

public final class ExperimentoFormBean extends ActionForm {
    private Long id;
    private String numero;
    private String nome;
    private String objetivo;
    private String descricao;
    private String instrucoes;

    public void setId(Long valor) {
        this.id = valor;
    }

    public Long getId() {
        return this.id;
    }

    public void setNumero(String valor) {
        this.numero = valor;
    }

    public String getNumero() {
        return this.numero;
    }
```

```
    public int getNumeroInteger() {
        if ((this.numero == "") || (this.numero == null)) {
            return 0;
        } else {
            return Integer.valueOf(this.numero);
        }
    }

    public void setNome(String valor) {
        this.nome = valor;
    }

    public String getNome() {
        return this.nome;
    }

    public void setObjetivo(String valor) {
        this.objetivo = valor;
    }

    public String getObjetivo() {
        return this.objetivo;
    }

    public void setDescricao(String valor) {
        this.descricao = valor;
    }

    public String getDescricao() {
        return this.descricao;
    }

    public void setInstrucoes(String valor) {
        this.instrucoes = valor;
    }

    public String getInstrucoes() {
        return this.instrucoes;
    }
}
```

ConfiguracaoFormBean.java

```
package br.ufrj.dee.elepot.struts.form;

import org.apache.struts.action.ActionForm;

public final class ConfiguracaoFormBean extends ActionForm {
    private String sistemaOperacional;
    private String portaSerial;
    private int tempoSessao;

    public String getSistemaOperacional() {
        return this.sistemaOperacional;
    }

    public void setSistemaOperacional(String valor) {
        this.sistemaOperacional = valor;
    }

    public String getPortaSerial() {
        return this.portaSerial;
    }

    public void setPortaSerial(String valor) {
        this.portaSerial = valor;
    }

    public int getTempoSessao() {
        return this.tempoSessao;
    }

    public void setTempoSessao(int valor) {
        this.tempoSessao = valor;
    }
}
```

PraticaFormBean.java

```
package br.ufrj.dee.elepot.struts.form;

import org.apache.struts.action.ActionForm;

public class PraticaFormBean extends ActionForm {
    private String tituloExperiencia;
    private String nomeAluno;

    public void setTituloExperiencia(String valor) {
        this.tituloExperiencia = valor;
    }


    public String getTituloExperiencia() {
        return this.tituloExperiencia;
    }

    public void setNomeAluno(String valor) {
        this.nomeAluno = valor;
    }

    public String getNomeAluno() {
        return this.nomeAluno;
    }
}
```

RelatorioPraticaFormBean.java

```
package br.ufrj.dee.elepot.struts.form;

import org.apache.struts.action.ActionForm;

public final class RelatorioPraticaFormBean extends ActionForm {
    private String dataInicio;
    private String dataFim;

    public void setDataInicio(String valor) {
        this.dataInicio = valor;
    }

    public String getDataInicio() {
        return this.dataInicio;
    }

    public void setDataFim(String valor) {
        this.dataFim = valor;
    }

    public String getDataFim() {
        return this.dataFim;
    }
}
```

ConversorFormBean.java

```
package br.ufrj.dee.elepot.struts.form;

public class ConversorFormBean extends PraticaFormBean {
    private boolean ligado;

    public void setLigado(boolean valor) {
        this.ligado = valor;
    }

    public boolean getLigado() {
        return this.ligado;
    }
}
```

ChopperFormBean.java

```
package br.ufrj.dee.elepot.struts.form;

public final class ChopperFormBean extends ConversorFormBean {
    private String fs;
    private String dc;
    private String tipoChopper;

    public void setFs(String valor) {
        this.fs = valor;
    }

    public String getFs() {
        return this.fs;
    }

    public void setDc(String valor) {
        this.dc = valor;
    }

    public String getDc() {
        return this.dc;
    }

    public void setTipoChopper(String valor) {
        this.tipoChopper = valor;
    }

    public String getTipoChopper() {
        if (tipoChopper == null)
            return "BUCK";
        else
            return this.tipoChopper;
    }
}
```

InversorFormBean.java

```
package br.ufrj.dee.elepot.struts.form;

public final class InversorFormBean extends ConversorFormBean {
    private String fs;
    private String fref;
    private String vref;

    public void setFs(String valor) {
        this.fs = valor;
    }

    public String getFs() {
        return this.fs;
    }

    public void setFref(String valor) {
        this.fref = valor;
    }

    public String getFref() {
        return this.fref;
    }

    public void setVref(String valor) {
        this.vref = valor;
    }

    public String getVref() {
        return this.vref;
    }
}
```

RetificadorFormBean.java

```
package br.ufrj.dee.elepot.struts.form;

public final class RetificadorFormBean extends ConversorFormBean {
    private String alpha;
    private String freq;
    public void setAlpha(String valor) {
        this.alpha = valor;
    }

    public String getAlpha() {
        return this.alpha;
    }

    public void setFreq(String valor) {
        this.freq = valor;
    }

    public String getFreq() {
        if (freq == null)
            return "60";
        else
            return this.freq;
    }
}
```

LoginFormBean.java

```
package br.ufrj.dee.elepot.struts.form;

import org.apache.struts.action.ActionForm;

public final class LoginFormBean extends ActionForm {
    private String login;
    private String senha;

    public void setLogin(String valor) {
        this.login = valor;
    }

    public void setSenha(String valor) {
        this.senha = valor;
    }

    public String getLogin() {
        return this.login;
    }

    public String getSenha() {
        return this.senha;
    }
}
```

TrocaSenhaFormBean.java

```
package br.ufrj.dee.elepot.struts.form;

import org.apache.struts.action.ActionForm;

public final class TrocaSenhaFormBean extends ActionForm {
    private String senha;
    private String telaOrigem;

    public void setSenha(String valor) {
        this.senha = valor;
    }

    public String getSenha() {
        return this.senha;
    }

    public void setTelaOrigem(String valor) {
        this.telaOrigem = valor;
    }
```

```
    public String getTelaOrigem() {
        return this.telaOrigem;
    }
}
```

C.11 – Pacote br.ufrj.dee.elepot.util

Conversao.java

```
package br.ufrj.dee.elepot.util;

import java.text.DecimalFormat;
import java.text.NumberFormat;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.List;
import java.util.ArrayList;
import java.util.Locale;

public final class Conversao {
    public static final String FORMATO_DD_MM_YYYY = "dd/MM/yyyy";
    public static final String FORMATO_YYYY_MM_DD = "yyyy/MM/dd";
    public static final String FORMATO_DD_MM_YYYY_HH_MM_SS = "dd/MM/yyyy HH:mm:ss";

    /* DateToString(data, formato)
     * converte um Date em uma String usando uma máscara de formatação
     */
    public static String DateToString(Date data, String formato) {
        return new SimpleDateFormat(formato).format(data);
    }

    /* StringToDate(data, formato)
     * converte uma String em um Date usando uma máscara de formatação
     */
    public static Date StringToDate(String data, String formato) {
        try {
            return new SimpleDateFormat(formato).parse(data);
        } catch (ParseException ex) {
            ex.printStackTrace();
            return null;
        }
    }

    /* FloatToString(numero)
     * converte um float (4 bytes) em uma String usando a formatação do Locale default
     */
    public static String FloatToString(float numero) {
        NumberFormat nfi = new DecimalFormat().getInstance(Locale.getDefault());
        return nfi.format(numero);
    }

    /* FloatToString(numero, qtdCasasDecimais)
     * converte um float (4 bytes) em uma String usando a formatação do Locale default
     * e com apenas a quantidade de casas decimais especificada
     */
    public static String FloatToString(float numero, int qtdCasasDecimais) {
        NumberFormat nfi = new DecimalFormat().getInstance(Locale.getDefault());
        nfi.setMaximumFractionDigits(qtdCasasDecimais);
        return nfi.format(numero);
    }

    /* StringToFloat(numero)
     * converte uma String em um float (4 bytes), podendo o String estar com vírgula
     * ou ponto como separador decimal (não usar separador de agrupamento)
     */
    public static float StringToFloat(String numero) {
        String numeroSemCaracteres = "";
        for (int i = 0; i < numero.length(); i++) {
            if ((numero.charAt(i) == ',') || (numero.charAt(i) == '.'))
                numeroSemCaracteres += "-";
            else
                if ((numero.charAt(i) >= '0') && (numero.charAt(i) <= '9'))
                    numeroSemCaracteres += String.valueOf(numero.charAt(i));
        }
        int separador = numeroSemCaracteres.indexOf('-');
```

```
        if (separador > -1) {
            String parteInteira = numeroSemCaracteres.substring(0, separador);
            String parteDecimal = numeroSemCaracteres.substring(separador + 1);
            return (float) (Integer.valueOf(parteInteira).intValue() + (Math.pow(0.1,
parteDecimal.length()) * Integer.valueOf(parteDecimal).intValue()));
        } else {

            return Integer.valueOf(numeroSemCaracteres).intValue();
        }
    }

    /* BooleanToSN(valor)
     * converte um valor boolean em um char tipo "S/N"
     */
    public static String BooleanToSN(Boolean valor) {
        if (valor == true)
            return "S";
        else
            return "N";
    }

    /* SNToBoolean(valor)
     * converte um char tipo "S/N" em um valor boolean
     */
    public static Boolean SNToBoolean(String valor) {
        if (valor.equals("S") || valor.equals("s"))
            return true;
        else
            return false;
    }

    /* IntToByte(valor)
     * converte um int (4 bytes) em um vetor de bytes onde
     * o byte de índice 0 é o mais significativo e assim por diante
     */
    public static byte[] IntToByte(int valor) {
        int valorInt = valor;
        byte[] valorByte = new byte[4];
        for (int i = 3; i >= 0; i--) {
            valorByte[i] = (byte)(valorInt & 0xFF);
            valorInt = valorInt >> 8;
        }
        return valorByte;
    }

    /* ByteToBit(valor)
     * converte um byte em um vetor de booleans representando o valor em bits
     * o bit de índice 0 é o mais significativo e assim por diante
     */
    public static boolean[] ByteToBit(byte valor) {
        boolean[] bits = new boolean[8];
        for (int i = 0; i < bits.length; i++) {
            bits[i] = ((valor & (1 << (7 - i))) != 0);
        }
        return bits;
    }

    /* BitToString(valor)
     * converte um vetor de booleans representando um valor em bits em uma String
     * o bits true são transformados em 1s e os bits falses em 0s
     */
    public static String BitToString(boolean[] valor) {
        String bitString = "";
        for (int i = 0; i < valor.length; i++) {
            bitString = bitString + ((valor[i]) ? "1" : "0");
        }
        return bitString;
    }
```

```
    /* StringToList(valor)
     * converte uma String separada por vírgulas em um List
     */
    public static List<String> StringToList(String valor) {
        String str = "";
        char separador = ',';
        List<String> lista = new ArrayList<String>();
        if (valor.length() != 0) {
            for (int i = 0; i < valor.length(); i++) {
                if (valor.charAt(i) == separador) {
                    lista.add(str);
                    str = "";
                } else {

                    str = str + valor.charAt(i);
                }
            }
            lista.add(str);
        }
        return lista;
    }
}
```

DispositivoSerial.java

```
package br.ufrj.dee.elepot.util;

import java.io.*;
import java.util.*;
import gnu.io.*;

public final class DispositivoSerial {
    private static int BAUD_RATE = 4800;

    private byte[]       pacote;
    private SerialPort   portaSerial;
    private OutputStream outputStream;
    private InputStream  inputStream;
    private Thread       readThread;
    private boolean      isValido;
    private boolean      isLivre;
    private boolean      isOk;

    public DispositivoSerial(String nomePorta, int tamanhoPacote, int timeout_ms) {
        this.isValido = false;
        this.isLivre = false;
        this.isOk = false;
        this.pacote = new byte[tamanhoPacote];

        Enumeration listaPortas = CommPortIdentifier.getPortIdentifiers();
        while (listaPortas.hasMoreElements()) {
            CommPortIdentifier idPorta = (CommPortIdentifier) listaPortas.nextElement();
            if (idPorta.getPortType() == CommPortIdentifier.PORT_SERIAL) {
                if (idPorta.getName().equals(nomePorta)) {
                    this.isValido = true;
                    iniciarDispositivo(idPorta, "ELEPOT", 2000);
                }
            }
        }
    }

    private void iniciarDispositivo(CommPortIdentifier portaId, String identificador,
int timeout) {
        try {
            this.portaSerial = (SerialPort) portaId.open(identificador, timeout);
            this.isLivre = true;

            try {
                this.outputStream = this.portaSerial.getOutputStream();
                this.inputStream = this.portaSerial.getInputStream();
            } catch (IOException e) {}

            try {
                this.portaSerial.setSerialPortParams(BAUD_RATE, SerialPort.DATABITS_8,
SerialPort.STOPBITS_1, SerialPort.PARITY_NONE);
```

```
                try {
                    this.portaSerial.notifyOnOutputEmpty(true);

                    this.isOk = true;

                } catch (Exception e) {
                    e.printStackTrace();
                }
            } catch (UnsupportedCommOperationException e) {
                e.printStackTrace();
            }
        } catch (PortInUseException e) {
            e.printStackTrace();
        };
    }

    public void encerrarDispositivo(int waittime) {
        try {
            try {
                Thread.sleep(waittime);
            } catch (InterruptedException ex) {
                ex.printStackTrace();
            }
        } finally {
            this.portaSerial.close();
        }
    }

    public void setPacote(byte[] pacote) {
        this.pacote = pacote;
    }

    public byte[] getPacote() {
        return this.pacote;
    }

    public boolean isOk() {
        return this.isOk;
    }

    public boolean enviarPacote() {
        boolean transmitido = false;
        if (this.isValido && this.isLivre && this.isOk) {
            try {
                this.outputStream.write(this.pacote);
                this.outputStream.flush();
                transmitido = true;
            } catch (IOException e) {}
        }
        return transmitido;
    }
}
```

LoginStatus.java

```
package br.ufrj.dee.elepot.util;

public final class LoginStatus {
    private boolean loginValido;
    private boolean usuarioInexistente;
    private boolean senhaInvalida;
    private boolean usuarioInativo;
    private boolean loginDuplicado;

    public LoginStatus() {
        this.loginValido = false;
        this.usuarioInexistente = false;
        this.senhaInvalida = false;
        this.usuarioInativo = false;
        this.loginDuplicado = false;
    }

    public void setLoginValido() {
        this.loginValido = true;
    }
```

```
    public boolean getLoginValido() {
        return this.loginValido;
    }

    public void setUsuarioInexistente() {
        this.usuarioInexistente = true;
    }

    public boolean getUsuarioInexistente() {
        return this.usuarioInexistente;
    }

    public void setSenhaInvalida() {
        this.senhaInvalida = true;
    }
    public boolean getSenhaInvalida() {
        return this.senhaInvalida;
    }

    public void setUsuarioInativo() {
        this.usuarioInativo = true;
    }

    public boolean getUsuarioInativo() {
        return this.usuarioInativo;
    }

    public void setLoginDuplicado() {
        this.loginDuplicado = true;
    }

    public boolean getLoginDuplicado() {
        return this.loginDuplicado;
    }
}
```

Pesquisa.java

```
package br.ufrj.dee.elepot.util;

import java.util.List;

public final class Pesquisa {

    public static String FiltroSQL(String campos, String valores) {
        String filtro = "";
        List<String> filtro_campos = Conversao.StringToList(campos);

        if (!filtro_campos.isEmpty()) {
            List<String> filtro_valores = Conversao.StringToList(valores);
            for (int i = 0; i < filtro_campos.size(); i++) {
                if (filtro_valores.get(i) != "") {
                    if (filtro != "") {
                        filtro = filtro + " AND ";
                    }
                    filtro = filtro + "(" + filtro_campos.get(i) + " LIKE '%" +
filtro_valores.get(i) + "%')";
                }
            }
        }
        return filtro;
    }
}
```

Senha.java

```
package br.ufrj.dee.elepot.util;

import java.util.Random;

public final class Senha {

    public static String geraSenha() {
        String senha = "";
        Integer n = 0;
        boolean x = false;
```

```
        Random random = new Random();

        for (int i = 0; i < 4; i++) {
            while (!x) {
                x = random.nextBoolean();
                if (n < 9)
                    n = n + 1;
                else
                    n = 0;
            }
            x = false;
            senha = senha + n.toString();
        }
        return senha;
    }
}
```

Zebrado.java

```
package br.ufrj.dee.elepot.util;

public final class Zebrado {
    private Object objeto;
    private int zebrado;

    public Object getObjeto() {
        return this.objeto;
    }

    public void setObjeto(Object objeto) {
        this.objeto = objeto;
    }

    public int getZebrado() {
        return this.zebrado;
    }

    public void setZebrado(int zebrado) {
        this.zebrado = zebrado;
    }
}
```

TestaDispositivoSerial.java

```
package br.ufrj.dee.elepot.util;

import br.ufrj.dee.elepot.Protocolo;
import br.ufrj.dee.elepot.dao.ConfiguracaoDAO;
import br.ufrj.dee.elepot.model.Configuracao;
import java.sql.SQLException;

public class TestaDispositivoSerial {

    public static void main(String[] args) throws SQLException {

        byte[] teste = {98, 98};

        int[] param = {1650614882, 1650614882, 1650614882, 1650614882};
        Protocolo protocolo = new Protocolo();
        protocolo.setPacote(teste[0], teste[1], param);

        for (int i = 0; i < 23; i++) {

System.out.println(Conversao.BitToString(Conversao.ByteToBit(protocolo.getPacote()[i])))
;
        }

        ConfiguracaoDAO cfgDAO = new ConfiguracaoDAO();
        Configuracao cfg = cfgDAO.carrega();

        DispositivoSerial com = new DispositivoSerial(cfg.getPortaSerial(), 2, 2000);
        com.setPacote(protocolo.getPacote());
        if (com.enviarPacote() == true) {
            System.out.println("Envio com sucesso");
            com.encerrarDispositivo(2000);
        } else {
```

```
            System.out.println("Ocorreu um erro");
        }
    }
}
```

C.12 – Pacote br.ufrj.dee.elepot.util.listener

SessaoListener.java

```
package br.ufrj.dee.elepot.util.listener;

import br.ufrj.dee.elepot.bo.PraticaBO;
import br.ufrj.dee.elepot.bo.UsuarioBO;
import br.ufrj.dee.elepot.model.Usuario;
import br.ufrj.dee.elepot.struts.form.LoginFormBean;
import javax.servlet.http.HttpSession;
import javax.servlet.http.HttpSessionListener;
import javax.servlet.http.HttpSessionEvent;
public final class SessaoListener implements HttpSessionListener {

    public void sessionCreated(HttpSessionEvent se) {
        System.out.println("ELEPOT: sessão iniciada ->
ID=".concat(se.getSession().getId()));
    }

    public void sessionDestroyed(HttpSessionEvent se) {
        HttpSession sessao = se.getSession();
        if (sessao != null) {
            Usuario usuario = (Usuario) sessao.getAttribute("usuario");
            if (usuario != null) {
                PraticaBO.setPraticaSemUso(usuario);
                UsuarioBO.setOffline(sessao.getServletContext(), usuario);
            }
        }
        System.out.println("ELEPOT: sessão destruída -> ID=".concat(sessao.getId()));
    }
}
```

C.13 – Arquivos de páginas *web* estáticas e dinâmicas (HTML e JSP)

index.jsp

```
<%@ include file="../WEB-INF/bloqueia_cache.jsp" %>
<%@ taglib uri="http://struts.apache.org/tags-html" prefix="html" %>
<%@ taglib uri="http://struts.apache.org/tags-bean" prefix="bean" %>
<html:html>
<%@ include file="cabecalho.html" %>
<center>
<b><font size="3"><bean:message key="titulo.elepot.bemvindo"/></font></b>
</center>
<hr size="2">
        <font size="2"><br>
        </font>
        <center>
        </center>
        <p><font size="2"><left>
            <bean:message key="login.introducao.linha1"/>
        </left><left></left></font></p>
        <font size="2"><left></left></font><left>
        <center>
          <table>
            <tr>
              <td>
                  <ul>
                    <li><font size="2"><strong><font color="#0000FF">
                        <bean:message key="login.laboratorio.titulo"/>
                    </font></strong><bean:message
key="login.laboratorio.titulo.obs"/><br>
                      <br>
                    </font></li>
                    <html:form action="/loginLab.do">
                    <table width="75%" border="1" cellpadding="0" cellspacing="0"
bgcolor="#CCCCCC">
                      <tr>
                        <td><table width="91%" border="0">
```

```
                          <tr>
                            <td width="31%"><font size="2"><bean:message
key="login.laboratorio.form.usuario"/></font></td>
                            <td width="52%"><font size="2">
                              <html:text property="login" value="" size="10"
maxlength="10" tabindex="1" onkeypress="if (event.keyCode == 13) {
document.forms[0].senha.focus() }" />
                              </font></td>
                            <td width="17%"> <font size="2">  </font> <font
size="2"> 
                              </font></td>
                          </tr>
                          <tr>
                            <td><font size="2"><bean:message
key="login.laboratorio.form.senha"/></font></td>
                            <td><font size="2">
                              <html:password property="senha" value="" size="10"
maxlength="10" tabindex="2" onkeypress="if (event.keyCode == 13) {
document.forms[0].submit() }" />
                              </font></td>
                            <td align="right"><font size="2">
                              <html:link href="javascript:document.forms[0].submit();"
tabindex="3" accesskey="ENTER">
                                <bean:message key="login.laboratorio.form.entrar"/>
                              </html:link></font></td>
                          </tr>
                        </table></td>
                    </tr>
                  </table>
                  </html:form>
                </ul>
                <ul>
                  <li><font size="2"><strong><font color="#0000FF">
                      <bean:message key="login.manutencao.titulo"/>
                  </font></strong><bean:message
key="login.manutencao.titulo.obs"/><br>
                    <br>
                    </font></li>
                  <html:form action="/loginMan.do">
                  <table width="75%" border="1" cellpadding="0" cellspacing="0"
bgcolor="#CCCCCC">
                    <tr>
                      <td><table width="92%" border="0">
                          <tr>
                            <td width="29%"><font size="2"><bean:message
key="login.manutencao.form.senha"/></font></td>
                            <td width="51%"><font size="2">
                              <html:hidden property="login" value="professor" />
                              <html:password property="senha" value="" size="10"
maxlength="10" tabindex="4" />
                              </font></td>
                            <td align="right"> <font size="2">
                              <html:link href="javascript:document.forms[1].submit();"
tabindex="5">
                                <bean:message key="login.manutencao.form.entrar"/>
                              </html:link></font></td>
                          </tr>
                        </table></td>
                    </tr>
                  </table>
                  </html:form>
                </ul>
              </td>
            </tr>
            <tr>
              <td> </td>
            </tr>
          </table>
        </center>
        </left>
        <center>
            <p><font size="0">
                <b><bean:message key="login.introducao.linha2"/></b><br><br>
                <bean:message key="login.introducao.linha3"/><br>
                <bean:message key="login.introducao.linha4"/><br>
                <bean:message key="login.introducao.linha5"/><br>
            </font></p><br>
```

```
            <font size="2"><b><bean:message key="login.links"/><br></b></font>
          </center>
          <font size="2"><left> </left></font><left>
          <ul>
            <li><font size="2"><a target="_blank" href="http://www.dee.ufrj.br/elepot/">
              <bean:message key="login.links.elepotgraduacao"/></a> -
              <bean:message key="login.links.elepotgraduacao.descricao"/>
            </font></li>
            <li><font size="2"><a target="_blank" href="http://www.coe.ufrj.br/gep/">
              <bean:message key="login.links.elepotcoppe"/></a> -
              <bean:message key="login.links.elepotcoppe.descricao"/>
            </font></li>
          </ul>
          </left></p>
<%@ include file="rodape.html" %>
</html:html>
```

cabecalho.html

```
<head>
<title><bean:message key="cabecalho.titulo"/></title>
</head>
<body bgcolor="#FFFFFF" text="#000000" link="#0000FF" vlink="#000080" alink="#FF0000" >
<center>
<table border="0" cellspacing="0" cellpadding="6" width="800" height="500" valign="top"
align="left" style="margin-top: -15; margin-bottom: -15; margin-left: -5; margin-right:
-5">
<tr>
<td background="/elepot/img/cabecalho.jpg" height="200">
    <center>
          <font face="Verdana, Arial, Helvetica, sans-serif"> <font
color="#000099"><font size=3>
          <strong><bean:message key="cabecalho.ufrj"/><br>
          <bean:message key="cabecalho.ufrj.ee"/><br>
          <bean:message key="cabecalho.ufrj.ee.dee"/><br>
          <br>
          <bean:message key="cabecalho.titulo"/></strong></font></font></font>
    </center>
</td>
</tr>
<tr>
<td>
<br>
```

rodape.html

```
</td></tr>
<tr>
<td background="/elepot/img/pano_inferior.jpg">
    <center><hr><font size="1">
    <bean:message key="rodape.elepot1"/><br>
    <bean:message key="rodape.elepot2"/><br>
    </font></center>
</td>
</tr>
</table>
</center>
</body>
```

cabecalho_lab.html

```
<head>
<title><bean:message key="cabecalho.titulo"/></title>
</head>
<body bgcolor="#FFFFFF" text="#000000" link="#0000FF" vlink="#000080" alink="#FF0000"
onload="abrePopups();" >
<center>
<table border="0" cellspacing="0" cellpadding="0" width="800" height="500" valign="top"
align="left">
<tr>
<td background="/elepot/img/cabecalho.jpg" height="200">
    <center>
          <font face="Verdana, Arial, Helvetica, sans-serif"> <font
color="#000099"><font size=3>
          <strong><bean:message key="cabecalho.ufrj"/><br>
```

```
          <bean:message key="cabecalho.ufrj.ee"/><br>
          <bean:message key="cabecalho.ufrj.ee.dee"/><br>
          <br>
          <bean:message key="cabecalho.titulo"/></strong></font></font></font>
    </center>
</td>
</tr>
<tr>
<td>
<br>
```

cabecalho_reduzido.html

```
<center>
<table border="0" cellspacing="0" cellpadding="6" width="800" height="500" valign="top"
align="left" style="margin-top: -15; margin-bottom: -15; margin-left: -5; margin-right:
-5">
<tr>
<td background="/elepot/img/cabecalho.jpg" height="200">
    <center>
          <font face="Verdana, Arial, Helvetica, sans-serif"> <font
color="#000099"><font size=3>
          <strong><bean:message key="cabecalho.ufrj"/><br>
          <bean:message key="cabecalho.ufrj.ee"/><br>
          <bean:message key="cabecalho.ufrj.ee.dee"/><br>
          <br>
          <bean:message key="cabecalho.titulo"/></strong></font></font></font>
    </center>
</td>
</tr>
<tr>
<td>
<br>
```

bloqueia_cache.jsp

```
<%
    response.setHeader("Cache-Control","no-cache"); //HTTP 1.1
    response.setHeader("Pragma","no-cache"); //HTTP 1.0
    response.setDateHeader ("Expires", 0); //prevents caching
    response.setHeader("Cache-Control","no-store"); //HTTP 1.1
%>
```

valida_sessao.jsp

```
<META HTTP-EQUIV="PRAGMA" CONTENT="NO-CACHE">
<%
    if (session.getAttribute("usuario") == null) {
        response.sendRedirect("index.jsp");
    }
%>
```

erro.jsp

```
<%@ taglib uri="http://struts.apache.org/tags-html" prefix="html" %>
<%@ taglib uri="http://struts.apache.org/tags-bean" prefix="bean" %>
<html>
<%@ include file="../cabecalho.html" %>
<center>
<b><font size="3"><bean:message key="titulo.elepot"/></font></b>
</center>
<hr size="2">
        <font size="2"><br>
        </font>
        <center>
          <font size="2"><left></left></font>
        </center>
        <p align="center"><font size="2"><left><strong><font color="#FF0000">
            <bean:message key="erro.atencao"/></font></strong></left></font></p>
        <p align="center"><font size="2"><left></left></font></p>
        <p align="center"><font size="2"><left><strong><font color="#FF0000">
            <bean:message key="erro.mensagem"/></font></strong></left></font></p>
        <p align="center"><font size="2"><left></left></font></p>
        <p align="center"><font size="2"><left><strong><font color="#FF0000">
```

```
            <bean:message key="erro.aviso"/></font></strong></left></font></p>
        <p><font size="2"><left></left></font><font size="2"><left></left></font></p>
        <p><font size="2"><left></left></font></p>
        <p><font size="2"><left></left></font></p>
        <p><font size="2"><left></left></font></p>
        <center>
            <html:link href="index.jsp"><font size="2"><bean:message
key="erro.voltar"/></font></html:link>
        </center>
<%@ include file="../rodape.html" %>
</html>
```

login_erro.jsp

```
<%@ taglib uri="http://struts.apache.org/tags-html" prefix="html" %>
<%@ taglib uri="http://struts.apache.org/tags-bean" prefix="bean" %>
<%@ taglib uri="http://struts.apache.org/tags-logic" prefix="logic" %>
<html:html>
<%@ include file="../cabecalho.html" %>
<br>
<center>
        <b><font size="3"><bean:message key="titulo.elepot.bemvindo"/></font></b>
        </center>
        <hr size="2">
        <font size="2"><br>
        </font>
        <center>
        </center>
        <p><font size="2"><left>
            <bean:message key="login.introducao.linha1"/>
        </left><left></left></font></p>
        <p><font size="2"><left><strong><font color="#FF0000">
        <center>
        <logic:equal name="loginStatus" value="usuarioinexistente"><bean:message
key="login.erro.usuarioinexistente" /></logic:equal>
        <logic:equal name="loginStatus" value="senhainvalida"><bean:message
key="login.erro.senhainvalida" /></logic:equal>
        <logic:equal name="loginStatus" value="usuarioinativo"><bean:message
key="login.erro.usuarioinativo" /></logic:equal>
        <logic:equal name="loginStatus" value="logininvalido"><bean:message
key="login.erro.logininvalido" /></logic:equal>
        <logic:equal name="loginStatus" value="loginduplicado"><bean:message
key="login.erro.loginduplicado" /></logic:equal>
        </center>
        </font></strong></left></font></p>
        <font size="2"><left></left></font><left>
        <center>
          <table>
            <tr>
              <td>
                  <ul>
                    <li><font size="2"><strong><font color="#0000FF">
                        <bean:message key="login.laboratorio.titulo"/>
                    </font></strong><bean:message
key="login.laboratorio.titulo.obs"/><br>
                    <br>
                    </font></li>
                    <html:form action="/loginLab.do">
                    <table width="75%" border="1" cellpadding="0" cellspacing="0"
bgcolor="#CCCCCC">
                      <tr>
                        <td><table width="91%" border="0">
                            <tr>
                              <td width="31%"><font size="2"><bean:message
key="login.laboratorio.form.usuario"/></font></td>
                              <td width="52%"><font size="2">
                                <html:text property="login" value="" size="10"
maxlength="10" onkeypress="if (event.keyCode == 13) { document.forms[0].senha.focus() }"
/>
                              </font></td>
                              <td width="17%"> <font size="2">  </font> <font
size="2"> 
                              </font></td>
                            </tr>
                            <tr>
```

```
                              <td><font size="2"><bean:message
key="login.laboratorio.form.senha"/></font></td>
                              <td><font size="2">
                                <html:password property="senha" value="" size="10"
maxlength="10" onkeypress="if (event.keyCode == 13) { document.forms[0].submit() }" />
                                </font></td>
                              <td align="right"><font size="2">
                                <html:link
href="javascript:document.forms[0].submit();">
                                    <bean:message key="login.laboratorio.form.entrar"/>
                                </html:link></font></td>
                            </tr>
                          </table></td>
                      </tr>
                    </table>
                    </html:form>
                  </ul>
                  <ul>
                    <li><font size="2"><strong><font color="#0000FF">
                      <bean:message key="login.manutencao.titulo"/>
                      </font></strong><bean:message
key="login.manutencao.titulo.obs"/><br>
                      <br>
                      </font></li>
                    <html:form action="/loginMan.do">
                    <table width="75%" border="1" cellpadding="0" cellspacing="0"
bgcolor="#CCCCCC">
                      <tr>
                        <td><table width="92%" border="0">
                            <tr>
                              <td width="29%"><font size="2"><bean:message
key="login.manutencao.form.senha"/></font></td>
                              <td width="51%"><font size="2">
                                <html:hidden property="login" value="professor" />
                                <html:password property="senha" value="" size="10"
maxlength="10" tabindex="3" />
                                </font></td>
                              <td align="right"><font size="2">
                              <html:link href="javascript:document.forms[1].submit();">
                                <bean:message key="login.manutencao.form.entrar"/>
                              </html:link></font></td>
                            </tr>
                          </table></td>
                      </tr>
                    </table>
                    </html:form>
                  </ul>
                </td>
            </tr>
            <tr>
              <td> </td>
            </tr>
          </table>
        </center>
        </left>
        <center>
            <p><font size="0">
                <b><bean:message key="login.introducao.linha2"/></b><br><br>
                <bean:message key="login.introducao.linha3"/><br>
                <bean:message key="login.introducao.linha4"/><br>
                <bean:message key="login.introducao.linha5"/><br>
            </font></p><br>
            <font size="2"><b><bean:message key="login.links"/><br></b></font>
        </center>
        <font size="2"><left> </left></font><left>
        <ul>
          <li><font size="2"><a target="_blank" href="http://www.dee.ufrj.br/elepot/">
            <bean:message key="login.links.elepotgraduacao"/></a> -
            <bean:message key="login.links.elepotgraduacao.descricao"/>
          </font></li>
          <li><font size="2"><a target="_blank" href="http://www.coe.ufrj.br/gep/">
            <bean:message key="login.links.elepotcoppe"/></a> -
            <bean:message key="login.links.elepotcoppe.descricao"/>
          </font></li>
        </ul>
        </left></p><%@ include file="../rodape.html" %>
</html:html>
```

laboratorio.jsp

```
<%@ include file="valida_sessao.jsp" %>
<%@ include file="bloqueia_cache.jsp" %>
<%@ taglib uri="http://struts.apache.org/tags-html" prefix="html" %>
<%@ taglib uri="http://struts.apache.org/tags-bean" prefix="bean" %>
<%@ taglib uri="http://struts.apache.org/tags-logic" prefix="logic" %>
<script>
function abrePopupDescricao(exp) {
    popUp = window.open('<%= response.encodeURL("montaPopup.do?janela=descricao&exp=")
%>'+exp, 'Atenção',
'menubar=no,location=no,resizable=no,scrollbars=yes,status=no,width=300,height=400');
    return;
}
function abrePopupInstrucoes(exp) {
    popUp = window.open('<%= response.encodeURL("montaPopup.do?janela=instrucoes&exp=")
%>'+exp, 'Atenção',
'menubar=no,location=no,resizable=no,scrollbars=yes,status=no,width=500,height=400');
    return;
}
</script>
<html:html>
<%@ include file="../cabecalho.html" %>
<center><b><bean:message key="laboratorio.titulo" /></b></center>
<br>
<left><font size="2">
    <ul>
        <logic:iterate id="experimento" name="lista"
type="br.ufrj.dee.elepot.model.Experimento">
        <li>
            <a href='telaLab_<bean:write name="experimento" property="numero"
/>.do;jsessionid=<%=
response.encodeURL("").substring(response.encodeURL("").indexOf("=") + 1)
%>?exp=<bean:write name="experimento" property="numero" />'>
                <bean:message key="laboratorio.experiencia" /> <bean:write
name="experimento" property="numero" />
            </a> &mdash; <bean:write name="experimento" property="nome" /><br>
            <b><bean:message key="laboratorio.objetivo" /></b>
            <bean:write name="experimento" property="objetivo" />
            <br>
            <a href="javascript:abrePopupInstrucoes(<bean:write name='experimento'
property='numero' />);"><bean:message key="laboratorio.popup.instrucoes" /></a>
             
            <a href="javascript:abrePopupDescricao(<bean:write name='experimento'
property='numero' />);"><bean:message key="laboratorio.popup.descricao" /></a>
            <br><br>
        </li>
        </logic:iterate>
        <li><font size="2"><html:link href='<%=
response.encodeURL("telaSenha.do?telaOrigem=laboratorio") %>'><bean:message
key="laboratorio.trocasenha"/></html:link></font><font size="2"><br>
        <br></font></li>
    </ul>
</font></left>
<br>
<div align="right"><html:link href='<%= response.encodeURL("logout.do") %>'><font
size="2"><bean:message key="laboratorio.voltar"/></font></html:link>
</div>
</td>
</center>
<%@ include file="../rodape.html" %>
</html:html>
```

manutencao.jsp

```
<%@ include file="valida_sessao.jsp" %>
<%@ include file="bloqueia_cache.jsp" %>
<%@ taglib uri="http://struts.apache.org/tags-html" prefix="html" %>
<%@ taglib uri="http://struts.apache.org/tags-bean" prefix="bean" %>
<html>
<%@ include file="../cabecalho.html" %>
<center>
    <b><font size="3"><bean:message key="titulo.elepot"/><br>
    <bean:message key="manutencao.titulo"/></font></b>
</center>
<hr size="2">
        <font size="2"><br>
```

```
        </font>
        <center>
        </center>
        <p><font size="2"><left>
           <bean:message key="manutencao.descricao"/>
        </left><left></left></font><font size="2"><left></left></font></p>
        <left>
        <center>
          <table>
            <tr>
              <td><ul>
                  <li> <font size="2"><html:link href='<%=
response.encodeURL("telaAlunos.do") %>'><bean:message
key="manutencao.cadastro.alunos"/></html:link><br>
                    <br></font></li>
                  <li> <font size="2"><html:link href='<%=
response.encodeURL("telaRelatorioPratica.do") %>'><bean:message
key="manutencao.relatorio.pratica"/></html:link><br>
                    <br></font></li>
                  <li> <font size="2"><html:link href='<%=
response.encodeURL("encerraTurma.do") %>'><bean:message
key="manutencao.encerraturma"/></html:link><br>
                    <br></font></li>
                  <li><font size="2"><html:link href='<%=
response.encodeURL("telaExperimentos.do") %>'><bean:message
key="manutencao.cadastro.experimentos"/><br></html:link>
                    <br></font></li>
                  <li><font size="2"><html:link href='<%=
response.encodeURL("telaConfiguracao.do") %>'><bean:message
key="manutencao.configuracao"/></html:link></font><font size="2"><br>
                    <br></font></li>
                  <li><font size="2"><html:link href='<%=
response.encodeURL("telaSenha.do") %>'><bean:message
key="manutencao.trocasenha"/></html:link></font><font size="2"><br>
                    <br></font></li>
                </ul></td>
            </tr>
          </table>
        </center>
        </left> <font size="2"><br>
        </font>
        <center>
          <font size="2"><b> </b></font>
        </center>
        <div align="right"><html:link href='<%= response.encodeURL("logout.do")
%>'><font size="2"><bean:message key="manutencao.voltar"/></font></html:link>
        </div>
<%@ include file="../rodape.html" %>
</html>
```

alunos.jsp

```
<%@ include file="valida_sessao.jsp" %>
<%@ taglib uri="http://struts.apache.org/tags-html" prefix="html" %>
<%@ taglib uri="http://struts.apache.org/tags-bean" prefix="bean" %>
<%@ taglib uri="http://struts.apache.org/tags-logic" prefix="logic" %>
<html:html>
<script>
    function getId() {
        var id = document.forms[3].escolhaId;
        if (id.length >= 2) {
            for (var i = 0; i < id.length; i++) {
                if (id[i].checked) {
                    document.forms[1].id.value = id[i].value;
                    document.forms[2].id.value = id[i].value;
                    break;
                }
            }
        } else {
            document.forms[1].id.value = id.value;
            document.forms[2].id.value = id.value;
        }
        return false;
    }
    function validar(formulario) {
        if (formulario.id.value == "") {
```

```
            alert('Para esta operação é necessário selecionar um item.');
        } else {
            return formulario.submit();
        }
    }
</script>
<%@ include file="../cabecalho.html" %>
<center>
    <b><font size="3"><bean:message key="titulo.elepot"/><br>
    <bean:message key="manutencao.titulo"/> - <bean:message
key="cadastro.alunos.titulo"/></font></b>
</center>
<hr size="2">
    <font size="2"><br>
    </font>
    <html:form action='<%= response.encodeURL("/pesquisaAlunos.do") %>'><center>
        <table width="310px" border="1" cellpadding="0" cellspacing="0"
bgcolor="#CCCCCC">
        <tr>
            <td><table width="100%" border="0">
                <tr>
                    <td colspan="3"><font size="2"><bean:message
key="cadastro.alunos.form.titulo"/></font></td>
                </tr>
                <tr>
                    <td><font size="2"><bean:message key="cadastro.alunos.form.nome"/>
                        <logic:present name="lista"><html:text property="nome" size="10"
maxlength="50" /></logic:present>
                        <logic:notPresent name="lista"><input type="text" name="nome"
value="" size="10" maxlength="50" ></logic:notPresent>
                    </font></td>
                    <td><font size="2"><bean:message
key="cadastro.alunos.form.identificacao"/>
                        <logic:present name="lista"><html:text property="identificacao"
size="10" maxlength="20" /></logic:present>
                        <logic:notPresent name="lista"><input type="text"
name="identificacao" value="" size="10" maxlength="20" ></logic:notPresent>
                    </font></td>
                </tr>
                </table>
                <table width="100%" border="0">
                <tr>
                    <td align="left"><font size="1">
                        <html:checkbox property="inativo" /><bean:message
key="cadastro.alunos.form.exibirinativos"/></font>
                    </font></td>
                    <td align="right"><font size="2">
                        <html:link href="javascript:document.forms[0].submit();"><font
size="2">
                            <bean:message key="cadastro.alunos.form.pesquisar"/>
                        </html:link></font>
                    </td>
                </tr>
                </table>
            </td>
        </tr>
        </table></center>
    </html:form>
    <html:form action='<%= response.encodeURL("/alteraAluno.do?tela=preparar")
%>'><html:hidden property="id" value="" /></html:form>
    <html:form action='<%= response.encodeURL("/excluiAluno.do") %>'><html:hidden
property="id" value="" /></html:form>
    <form name="frmLista">
    <center>
    <table width="100%" border="0" cellpadding="0" cellspacing="0" bgcolor="#CCCCCC">
    <tr>
        <td bordercolor="#FFFFFF" bgcolor="#FFFFFF">
            <div align="center">
            <table width="350" border="0">
            <logic:present name="lista">
                <tr>
                    <td td width="4%" bgcolor="#FFE5CA"><div align="center"></td>
                    <td bgcolor="#FFE5CA"><div align="left">
                    <font size="2"><b><bean:message
key="cadastro.alunos.form.nome"/></b></font>
                    </div></td>
                    <td width="41%" bgcolor="#FFE5CA"><div align="left">
```

```
                    <font size="2"><b><bean:message
key="cadastro.alunos.form.identificacao"/></b></font>
                    </div></td>
                </tr>
                <logic:iterate id="usuario" name="lista"
type="br.ufrj.dee.elepot.util.Zebrado">
                <tr>
                    <logic:equal name="usuario" property="zebrado" value="0"><td
bgcolor="#FFFFFF"></logic:equal>
                    <logic:notEqual name="usuario" property="zebrado" value="0"><td
bgcolor="#FFE5CA"></logic:notEqual>
                    <div align="left">
                        <input type="radio" name="escolhaId" value='<bean:write
name="usuario" property="objeto.id" />' unchecked onclick="getId(this);">
                    </div></td>
                    <logic:equal name="usuario" property="zebrado" value="0"><td
bgcolor="#FFFFFF"></logic:equal>
                    <logic:notEqual name="usuario" property="zebrado" value="0"><td
bgcolor="#FFE5CA"></logic:notEqual>
                    <div align="left">
                        <font size="2"><bean:write name="usuario" property="objeto.nome"
/></font>
                    </div></td>
                    <logic:equal name="usuario" property="zebrado" value="0"><td
bgcolor="#FFFFFF"></logic:equal>
                    <logic:notEqual name="usuario" property="zebrado" value="0"><td
bgcolor="#FFE5CA"></logic:notEqual>
                    <div align="left">
                        <font size="2"><bean:write name="usuario"
property="objeto.identificacao" /></font>
                    </div></td>
                </tr>
                </logic:iterate>
            </logic:present>
            </table>
            </div>
        </td>
    </tr>
    </table>
    </form>
    <br>
    <div align="center">
        <html:link href='<%= response.encodeURL("telaIncluirAlunos.do") %>'><font
size="2">
            <bean:message key="cadastro.alunos.form.incluir"/></font>
        </html:link>
         | 
        <html:link href="javascript:validar(document.forms[1]);" ><font size="2">
            <bean:message key="cadastro.alunos.form.alterar"/></font>
        </html:link>
         | 
        <html:link href="javascript:validar(document.forms[2]);" ><font size="2">
            <bean:message key="cadastro.alunos.form.excluir"/></font>
        </html:link>
         | 
        <html:link href='<%= response.encodeURL("telaManutencao.do") %>'><font size="2">
            <bean:message key="cadastro.alunos.form.retornar"/></font>
        </html:link>
    </div>
<%@ include file="../rodape.html" %>
</html:html>
```

alunos_form.jsp

```
<%@ include file="valida_sessao.jsp" %>
<%@ taglib uri="http://struts.apache.org/tags-html" prefix="html" %>
<%@ taglib uri="http://struts.apache.org/tags-bean" prefix="bean" %>
<%@ taglib uri="http://struts.apache.org/tags-logic" prefix="logic" %>
<html:html>
<script language="javascript" src="javascript/funcoesGenericas.js"></script>
<script language="javascript" src="javascript/validaAlunoFormBean.js"></script>
<script>
    function getSenha() {
        document.forms[0].senha.value = document.forms[0].novaSenha.value;
        return false;
    }
```

```
    function incluir(f) {
        document.forms[0].action='<%= response.encodeURL("incluiAluno.do") %>';
        validarInclusao(document.forms[0]);
    }
    function alterar(f) {
        document.forms[0].action='<%= response.encodeURL("alteraAluno.do?tela=gravar")
%>';
        validarAlteracao(document.forms[0]);
    }
</script>
<%@ include file="../cabecalho.html" %>
<center>
    <b><font size="3"><bean:message key="titulo.elepot"/><br>
    <bean:message key="manutencao.titulo"/> - <bean:message
key="cadastro.alunos.titulo"/></font></b>
</center>
<hr size="2">
    <font size="2"><br>
    </font>
    <html:form action='<%= response.encodeURL("/incluiAluno.do") %>'>
    <center>
        <logic:present name="aluno"><html:hidden name="aluno" property="id"
/></logic:present>
        <table width="100%" border="0" cellpadding="0" cellspacing="0"
bgcolor="#CCCCCC">
        <tr>
            <td bordercolor="#FFFFFF" bgcolor="#FFFFFF"><div align="center">
                <table width="250" border="0">
                <logic:equal name="tela" value="duplicado"><tr>
                    <td colspan="2" bgcolor="#FFFFFF"><div align="left"><font
size="2"><left><strong><font color="#FF0000">
                        <center><bean:message key="cadastro.alunos.erro.duplicado"
/></center>
                    </font></strong></left></font></div></td>
                </tr></logic:equal>
                <tr>
                    <td bgcolor="#FFFFFF"><div align="left">
                        <font size="2"><bean:message
key="cadastro.alunos.form.nome"/></font>
                    </div></td>
                    <td width="41%" bgcolor="#FFFFFF"><div align="left"><font size="2">
                        <logic:notPresent name="aluno">
                            <logic:equal name="tela" value="duplicado"><html:text
name="formulario" property="nome" size="40" maxlength="50" /></logic:equal>
                            <logic:notEqual name="tela" value="duplicado"><html:text
property="nome" value="" size="40" maxlength="50" /></logic:notEqual>
                        </logic:notPresent>
                        <logic:present name="aluno">
                            <html:text name="aluno" property="nome" size="40"
maxlength="50" />
                        </logic:present>
                    </font></div></td>
                </tr>
                <tr>
                    <td bgcolor="#FFFFFF"><div align="left">
                        <font size="2"><bean:message
key="cadastro.alunos.form.identificacao"/></font>
                    </div></td>
                    <td width="41%" bgcolor="#FFFFFF"><div align="left"><font size="2">
                        <logic:notPresent name="aluno">
                            <logic:equal name="tela" value="duplicado"><html:text
name="formulario" property="identificacao" size="20" maxlength="20" /></logic:equal>
                            <logic:notEqual name="tela" value="duplicado"><html:text
property="identificacao" value="" size="20" maxlength="20" /></logic:notEqual>
                        </logic:notPresent>
                        <logic:present name="aluno">
                            <html:text name="aluno" property="identificacao" size="20"
maxlength="20" />
                        </logic:present>
                    </font></div></td>
                </tr>
                <tr>
                    <td bgcolor="#FFFFFF"><div align="left">
                        <font size="2"><bean:message
key="cadastro.alunos.form.login"/></font>
                    </div></td>
                    <td width="41%" bgcolor="#FFFFFF"><div align="left"><font size="2">
```

```
                        <logic:notPresent name="aluno">
                            <logic:equal name="tela" value="duplicado"><html:text
name="formulario" property="login" size="20" maxlength="10" /></logic:equal>
                            <logic:notEqual name="tela" value="duplicado"><html:text
property="login" value="" size="20" maxlength="10" /></logic:notEqual>
                        </logic:notPresent>
                        <logic:present name="aluno">
                            <html:text name="aluno" property="login" size="20"
maxlength="10" />
                        </logic:present>
                    </font></div></td>
                </tr>
                <logic:present name="aluno">
                <tr>
                    <td bgcolor="#FFFFFF"><div align="left">
                        <font size="2"><bean:message
key="cadastro.alunos.form.senha"/></font>
                    </div></td>
                    <td width="41%" bgcolor="#FFFFFF"><div align="left"><font size="2">
                        <input type="password" name="novaSenha" value="00000000"
size="10" maxlength="10" onchange="getSenha();">
                        <input type="hidden" name="senha">
                    </font></div></td>
                </tr>
                <tr>
                    <td bgcolor="#FFFFFF"><div align="left">
                        <font size="2"><bean:message
key="cadastro.alunos.form.ativo"/></font>
                    </div></td>
                    <td width="41%" bgcolor="#FFFFFF"><div align="left"><font size="2">
                        <html:checkbox name="aluno" property="ativo" value="true" />
                    </font></div></td>
                </tr>
                </logic:present>
                </table>
            </div></td>
        </tr>
        </table></center>
        <br>
        <div align="center">
          <logic:notPresent name="aluno">
            <html:link href="javascript:incluir(this);">
              <font size="2"><bean:message key="cadastro.alunos.form.gravar"/></font>
            </html:link>
          </logic:notPresent>
          <logic:present name="aluno">
            <html:link href="javascript:alterar(this);">
              <font size="2"><bean:message key="cadastro.alunos.form.gravar"/></font>
            </html:link>
          </logic:present>
           | 
          <html:link href='<%= response.encodeURL("telaAlunos.do") %>'><font size="2">
            <bean:message key="cadastro.alunos.form.retornar"/></font>
          </html:link>
        </div>
    </html:form>
    <br>
<%@ include file="../rodape.html" %>
</html:html>
```

alunos_ok.jsp

```
<%@ include file="valida_sessao.jsp" %>
<%@ taglib uri="http://struts.apache.org/tags-html" prefix="html" %>
<%@ taglib uri="http://struts.apache.org/tags-bean" prefix="bean" %>
<%@ taglib uri="http://struts.apache.org/tags-logic" prefix="logic" %>
<html>
<%@ include file="../cabecalho.html" %>
<center>
    <b><font size="3"><bean:message key="titulo.elepot"/><br>
    <bean:message key="manutencao.titulo"/> - <bean:message
key="cadastro.alunos.titulo"/></font></b>
</center>
<hr size="2">
<font size="2"><br></font>
<logic:notEqual name="novaSenha" value="">
```

```
    <p><font size="2">
        <bean:message key="cadastro.alunos.ok.incluido"/>
        <br><br>
        <bean:message key="cadastro.alunos.novasenha"/>
        </font><font size="4"><bean:write name="novaSenha" /></font>
    </p>
    <p><font size="2"><bean:message
key="cadastro.alunos.novasenha.observacao"/></font></p>
</logic:notEqual>
<logic:equal name="novaSenha" value="">
    <p><font size="2">
        <logic:equal name="tela" value="alteracao">
            <bean:message key="cadastro.alunos.ok.alterado"/>
        </logic:equal>
        <logic:equal name="tela" value="exclusao">
            <bean:message key="cadastro.alunos.ok.excluido"/>
        </logic:equal>
    </font></p>
</logic:equal>
<div align="right">
    <html:link href='<%= response.encodeURL("telaAlunos.do") %>'>
        <font size="2"><bean:message key="cadastro.alunos.form.retornar"/></font>
    </html:link>
</div>
<%@ include file="../rodape.html" %>
</html>
```

experimentos.jsp

```
<%@ include file="valida_sessao.jsp" %>
<%@ taglib uri="http://struts.apache.org/tags-html" prefix="html" %>
<%@ taglib uri="http://struts.apache.org/tags-bean" prefix="bean" %>
<%@ taglib uri="http://struts.apache.org/tags-logic" prefix="logic" %>
<html:html>
<script>
    function getId() {
        var id = document.forms[3].escolhaId;
        if (id.length >= 2) {
            for (var i = 0; i < id.length; i++) {
                if (id[i].checked) {
                    document.forms[1].id.value = id[i].value;
                    document.forms[2].id.value = id[i].value;
                    break;
                }
            }
        } else {
            document.forms[1].id.value = id.value;
            document.forms[2].id.value = id.value;
        }
        return false;
    }
    function validar(formulario) {
        if (formulario.id.value == "") {
            alert('Para esta operação é necessário selecionar um item.');
        } else {
            return formulario.submit();
        }
    }
</script>
<%@ include file="../cabecalho.html" %>
<center>
    <b><font size="3"><bean:message key="titulo.elepot"/><br>
    <bean:message key="manutencao.titulo"/> - <bean:message
key="cadastro.experimentos.titulo"/></font></b>
</center>
<hr size="2">
    <font size="2"><br>
    </font>
    <font size="2"><p><left><bean:message
key="cadastro.experimentos.observacao"/></left></p></font>
    <br>
    <html:form action='<%= response.encodeURL("/pesquisaExperimentos.do") %>'><center>
        <table width="310px" border="1" cellpadding="0" cellspacing="0"
bgcolor="#CCCCCC">
        <tr>
            <td><table width="100%" border="0">
```

```
                <tr>
                    <td colspan="3"><font size="2"><bean:message
key="cadastro.experimentos.form.titulo"/></font></td>
                </tr>
                <tr>
                    <td><font size="2"><bean:message
key="cadastro.experimentos.form.numero"/>
                        <logic:present name="lista"><html:text property="numero"
size="3" maxlength="3" /></logic:present>
                        <logic:notPresent name="lista"><input type="text" name="numero"
value="" size="3" maxlength="3" ></logic:notPresent>
                    </font></td>
                    <td><font size="2"><bean:message
key="cadastro.experimentos.form.nome"/>
                        <logic:present name="lista"><html:text property="nome" size="25"
maxlength="50" /></logic:present>
                        <logic:notPresent name="lista"><input type="text" name="nome"
value="" size="25" maxlength="50" ></logic:notPresent>
                    </font></td>
                </tr>
                <tr>
                    <td colspan="2" align="right"><font size="2">
                        <html:link href="javascript:document.forms[0].submit();"><font
size="2">
                            <bean:message key="cadastro.experimentos.form.pesquisar"/>
                        </html:link></font>
                    </td>
                </tr>
                </table>
            </td>
        </tr>
        </table></center>
    </html:form>
    <html:form action='<%= response.encodeURL("/alteraExperimento.do?tela=preparar")
%>'><html:hidden property="id" value="" /></html:form>
    <html:form action='<%= response.encodeURL("/excluiExperimento.do") %>'><html:hidden
property="id" value="" /></html:form>
    <form name="frmLista">
    <center>
    <table width="100%" border="0" cellpadding="0" cellspacing="0" bgcolor="#CCCCCC">
    <tr>
        <td bordercolor="#FFFFFF" bgcolor="#FFFFFF">
            <div align="center">
            <table width="350" border="0">
            <logic:present name="lista">
                <tr>
                    <td td width="4%" bgcolor="#FFE5CA"><div align="center"></td>
                    <td bgcolor="#FFE5CA"><div align="center">
                    <font size="2"><b><bean:message
key="cadastro.experimentos.form.numero"/></b></font>
                    </div></td>
                    <td width="71%" bgcolor="#FFE5CA"><div align="left">
                    <font size="2"><b><bean:message
key="cadastro.experimentos.form.nome"/></b></font>
                    </div></td>
                </tr>
                <logic:iterate id="experimento" name="lista"
type="br.ufrj.dee.elepot.util.Zebrado">
                <tr>
                    <logic:equal name="experimento" property="zebrado" value="0"><td
bgcolor="#FFFFFF"></logic:equal>
                    <logic:notEqual name="experimento" property="zebrado" value="0"><td
bgcolor="#FFE5CA"></logic:notEqual>
                    <div align="left">
                        <input type="radio" name="escolhaId" value='<bean:write
name="experimento" property="objeto.id" />' unchecked onclick="getId();">
                    </div></td>
                    <logic:equal name="experimento" property="zebrado" value="0"><td
bgcolor="#FFFFFF"></logic:equal>
                    <logic:notEqual name="experimento" property="zebrado" value="0"><td
bgcolor="#FFE5CA"></logic:notEqual>
                    <div align="center">
                        <font size="2"><bean:write name="experimento"
property="objeto.numero" /></font>
                    </div></td>
                    <logic:equal name="experimento" property="zebrado" value="0"><td
bgcolor="#FFFFFF"></logic:equal>
```

```
                    <logic:notEqual name="experimento" property="zebrado" value="0"><td
bgcolor="#FFE5CA"></logic:notEqual>
                    <div align="left">
                        <font size="2"><bean:write name="experimento"
property="objeto.nome" /></font>
                    </div></td>
                </tr>
                </logic:iterate>
            </logic:present>
            </table>
            </div>
        </td>
    </tr>
    </table>
    </form>
    <br>
    <div align="center">
        <html:link href='<%= response.encodeURL("telaIncluirExperimentos.do") %>'><font
size="2">
            <bean:message key="cadastro.experimentos.form.incluir"/></font>
        </html:link>
         | 
        <html:link href="javascript:validar(document.forms[1]);" ><font size="2">
            <bean:message key="cadastro.experimentos.form.alterar"/></font>
        </html:link>
         | 
        <html:link href="javascript:validar(document.forms[2]);" ><font size="2">
            <bean:message key="cadastro.experimentos.form.excluir"/></font>
        </html:link>
         | 
        <html:link href='<%= response.encodeURL("telaManutencao.do") %>'><font size="2">
            <bean:message key="cadastro.experimentos.form.retornar"/></font>
        </html:link>
    </div>
<%@ include file="../rodape.html" %>
</html:html>
```

experimentos_form.jsp

```
<%@ include file="valida_sessao.jsp" %>
<%@ taglib uri="http://struts.apache.org/tags-html" prefix="html" %>
<%@ taglib uri="http://struts.apache.org/tags-bean" prefix="bean" %>
<%@ taglib uri="http://struts.apache.org/tags-logic" prefix="logic" %>
<html:html>
<script language="javascript" src="javascript/funcoesGenericas.js"></script>
<script language="javascript" src="javascript/validaExperimentoFormBean.js"></script>
<script>
    function incluir(f) {
        document.forms[0].action='<%= response.encodeURL("incluiExperimento.do") %>';
        validar(document.forms[0]);
    }
    function alterar(f) {
        document.forms[0].action='<%=
response.encodeURL("alteraExperimento.do?tela=gravar") %>';
        validar(document.forms[0]);
    }
</script>
<%@ include file="../cabecalho.html" %>
<center>
    <b><font size="3"><bean:message key="titulo.elepot"/><br>
    <bean:message key="manutencao.titulo"/> - <bean:message
key="cadastro.experimentos.titulo"/></font></b>
</center>
<hr size="2">
    <font size="2"><br>
    </font>
    <html:form action='<%= response.encodeURL("/incluiExperimento.do") %>'>
    <center>
        <logic:present name="experimento"><html:hidden name="experimento" property="id"
/></logic:present>
        <table width="100%" border="0" cellpadding="0" cellspacing="0"
bgcolor="#CCCCCC">
        <tr>
            <td bordercolor="#FFFFFF" bgcolor="#FFFFFF"><div align="center">
                <table width="250" border="0">
                <tr>
```

```
                    <td bgcolor="#FFFFFF"><div align="left">
                        <font size="2"><bean:message
key="cadastro.experimentos.form.numero"/></font>
                    </div></td>
                    <td width="41%" bgcolor="#FFFFFF"><div align="left"><font size="2">
                        <logic:notPresent name="experimento"><html:text
property="numero" value="" size="3" maxlength="3" /></logic:notPresent>
                        <logic:present name="experimento"><html:text name="experimento"
property="numero" size="3" maxlength="3" /></logic:present>
                    </font></div></td>
                </tr>
                <tr>
                    <td bgcolor="#FFFFFF"><div align="left">
                        <font size="2"><bean:message
key="cadastro.experimentos.form.nome"/></font>
                    </div></td>
                    <td width="41%" bgcolor="#FFFFFF"><div align="left"><font size="2">
                        <logic:notPresent name="experimento"><html:text property="nome"
value="" size="30" maxlength="50" /></logic:notPresent>
                        <logic:present name="experimento"><html:text name="experimento"
property="nome" size="30" maxlength="50" /></logic:present>
                    </font></div></td>
                </tr>
                <tr>
                    <td bgcolor="#FFFFFF"><div align="left">
                        <font size="2"><bean:message
key="cadastro.experimentos.form.objetivo"/></font>
                    </div></td>
                    <td width="41%" bgcolor="#FFFFFF"><div align="left"><font size="2">
                        <logic:notPresent name="experimento"><html:text
property="objetivo" value="" size="50" maxlength="150" /></logic:notPresent>
                        <logic:present name="experimento"><html:text name="experimento"
property="objetivo" size="50" maxlength="150" /></logic:present>
                    </font></div></td>
                </tr>
                <tr>
                    <td bgcolor="#FFFFFF"><div align="left">
                        <font size="2"><bean:message
key="cadastro.experimentos.form.descricao"/></font>
                    </div></td>
                    <td width="41%" bgcolor="#FFFFFF"><div align="left"><font size="2">
                        <logic:notPresent name="experimento"><html:textarea
property="descricao" value="" rows="5" cols="50" /></logic:notPresent>
                        <logic:present name="experimento"><html:textarea
name="experimento" property="descricao" rows="5" cols="50" /></logic:present>
                    </font></div></td>
                </tr>
                <tr>
                    <td bgcolor="#FFFFFF"><div align="left">
                        <font size="2"><bean:message
key="cadastro.experimentos.form.instrucoes"/></font>
                    </div></td>
                    <td width="41%" bgcolor="#FFFFFF"><div align="left"><font size="2">
                        <logic:notPresent name="experimento"><html:textarea
property="instrucoes" value="" rows="5" cols="50" /></logic:notPresent>
                        <logic:present name="experimento"><html:textarea
name="experimento" property="instrucoes" rows="5" cols="50" /></logic:present>
                    </font></div></td>
                </tr>
                </table>
            </div></td>
        </tr>
        </table></center>
        <br>
        <div align="center">
          <logic:notPresent name="experimento">
            <html:link href="javascript:incluir(this);">
              <font size="2"><bean:message
key="cadastro.experimentos.form.gravar"/></font>
            </html:link>
          </logic:notPresent>
          <logic:present name="experimento">
            <html:link href="javascript:alterar(this);">
              <font size="2"><bean:message
key="cadastro.experimentos.form.gravar"/></font>
            </html:link>
          </logic:present>
```

```
           | 
          <html:link href='<%= response.encodeURL("telaExperimentos.do") %>'><font
size="2">
            <bean:message key="cadastro.experimentos.form.retornar"/></font>
          </html:link>
        </div>
    </html:form>
    <br>
<%@ include file="../rodape.html" %>
</html:html>
```

experimentos_ok.jsp

```
<%@ include file="valida_sessao.jsp" %>
<%@ taglib uri="http://struts.apache.org/tags-html" prefix="html" %>
<%@ taglib uri="http://struts.apache.org/tags-bean" prefix="bean" %>
<%@ taglib uri="http://struts.apache.org/tags-logic" prefix="logic" %>
<html>
<%@ include file="../cabecalho.html" %>
<center>
    <b><font size="3"><bean:message key="titulo.elepot"/><br>
    <bean:message key="manutencao.titulo"/> - <bean:message
key="cadastro.experimentos.titulo"/></font></b>
</center>
<hr size="2">
    <font size="2"><br></font>
    <p><font size="2">
        <logic:equal name="tela" value="inclusao">
            <bean:message key="cadastro.experimentos.ok.incluido"/>
        </logic:equal>
        <logic:equal name="tela" value="alteracao">
            <bean:message key="cadastro.experimentos.ok.alterado"/>
        </logic:equal>
        <logic:equal name="tela" value="exclusao">
            <bean:message key="cadastro.experimentos.ok.excluido"/>
        </logic:equal>
    <div align="right">
        <html:link href='<%= response.encodeURL("telaExperimentos.do") %>'>
            <font size="2"><bean:message
key="cadastro.experimentos.form.retornar"/></font>
        </html:link>
    </div>
<%@ include file="../rodape.html" %>
</html>
```

configuracao.jsp

```
<%@ include file="valida_sessao.jsp" %>
<%@ taglib uri="http://struts.apache.org/tags-html" prefix="html" %>
<%@ taglib uri="http://struts.apache.org/tags-bean" prefix="bean" %>
<html:html>
<script language="javascript" src="javascript/funcoesGenericas.js"></script>
<script language="javascript" src="javascript/validaConfiguracaoFormBean.js"></script>
<script>
    function trocaPortaSerial(formulario) {
        if (formulario.sistemaOperacional.value == "WINDOWS") {
            formulario.portaSerial.value = "COM1";
        } else {
            formulario.portaSerial.value = "/dev/ttyS0";
        }
    }
</script>
<%@ include file="../cabecalho.html" %>
<center>
    <b><font size="3"><bean:message key="titulo.elepot"/><br>
    <bean:message key="manutencao.titulo"/> - <bean:message
key="configuracao.titulo"/></font></b>
</center>
<hr size="2">
        <font size="2"><br>
        </font>
        <html:form action='<%= response.encodeURL("/gravaConfiguracao.do") %>'>
          <table width="100%" border="0" cellpadding="0" cellspacing="0"
bgcolor="#CCCCCC">
            <tr>
              <td bordercolor="#FFFFFF" bgcolor="#FFFFFF"> <div align="center">
```

```
                  <table width="320px" border="0">
                    <tr>
                      <td bgcolor="#FFFFFF"><div align="left"><font size="2">
                        <bean:message
key="configuracao.form.sistemaoperacional"/></font></div></td>
                      <td bgcolor="#FFFFFF"><div align="left"><font size="2">
                        <html:select name="configuracao" property="sistemaOperacional"
onchange="trocaPortaSerial(document.forms[0]);">
                            <html:option value="WINDOWS">WINDOWS</html:option>
                            <html:option value="LINUX">LINUX</html:option>
                        </html:select>
                        </font></div>
                      </td>
                    </tr>
                    <tr>
                      <td bgcolor="#FFFFFF"><div align="left"><font size="2">
                        <bean:message
key="configuracao.form.portaserial"/></font></div></td>
                      <td bgcolor="#FFFFFF"><div align="left"><font size="2">
                        <html:text name="configuracao" property="portaSerial" size="20"
maxlength="50" />
                        </font></div></td>
                    </tr>
                    <tr>
                      <td bgcolor="#FFFFFF"><div align="left"><font size="2">
                        <bean:message
key="configuracao.form.temposessao"/></font></div></td>
                      <td bgcolor="#FFFFFF"><div align="left"><font size="2">
                        <html:text name="configuracao" property="tempoSessao" size="3"
maxlength="3" />
                        </font></div></td>
                    </tr>
                  </table>
                </div>
              </td>
            </tr>
          </table>
          <br>
          <div align="center">
            <html:link href="javascript:validar(document.forms[0]);"><font size="2">
              <bean:message key="configuracao.form.gravar"/></font>
            </html:link>
             | 
            <html:link href='<%= response.encodeURL("telaManutencao.do") %>'><font
size="2">
              <bean:message key="configuracao.form.retornar"/></font>
            </html:link>
          </div>
          </html:form>
<%@ include file="../rodape.html" %>
</html:html>
```

configuracao_ok.jsp

```
<%@ include file="valida_sessao.jsp" %>
<%@ taglib uri="http://struts.apache.org/tags-html" prefix="html" %>
<%@ taglib uri="http://struts.apache.org/tags-bean" prefix="bean" %>
<html>
<%@ include file="../cabecalho.html" %>
<center>
    <b><font size="3"><bean:message key="titulo.elepot"/><br>
    <bean:message key="manutencao.titulo"/> - <bean:message
key="configuracao.titulo"/></font></b>
</center>
<hr size="2">
    <font size="2"><br></font>
    <p><font size="2"><bean:message key="configuracao.ok"/></font></p>
    <div align="right">
        <html:link href='<%= response.encodeURL("telaManutencao.do") %>'>
            <font size="2"><bean:message key="configuracao.voltar"/></font>
        </html:link>
    </div>
<%@ include file="../rodape.html" %>
</html>
```

relatorio_pratica.jsp

```
<%@ include file="valida_sessao.jsp" %>
<%@ taglib uri="http://struts.apache.org/tags-html" prefix="html" %>
<%@ taglib uri="http://struts.apache.org/tags-bean" prefix="bean" %>
<%@ taglib uri="http://struts.apache.org/tags-logic" prefix="logic" %>
<html:html>
<script language="javascript" src="javascript/funcoesGenericas.js"></script>
<script language="javascript"
src="javascript/validaRelatorioPraticaFormBean.js"></script>
<script>
    function getDataAtual() {
        data = new Date();
        dia = data.getDate();
        mes = data.getMonth() + 1;
        ano = data.getFullYear();
        if (dia < 10) { dia = "0" + dia }
        if (mes < 10) { mes = "0" + mes }
        document.forms[0].dataInicio.value=dia+"/"+mes+"/"+ano;
        document.forms[0].dataFim.value=dia+"/"+mes+"/"+ano;
        return null;
    }
</script>
<logic:present name="dataInicio">
    <%@ include file="../cabecalho.html" %>
</logic:present>
<logic:notPresent name="dataInicio">
    <head>
    <title><bean:message key="cabecalho.titulo"/></title>
    </head>
    <body bgcolor="#FFFFFF" text="#000000" link="#0000FF" vlink="#000080"
alink="#FF0000" onload="getDataAtual();">
    <%@ include file="../cabecalho_reduzido.html" %>
</logic:notPresent>
<center>
    <b><font size="3"><bean:message key="titulo.elepot"/><br>
    <bean:message key="manutencao.titulo"/> - <bean:message
key="relatorio.pratica.titulo"/></font></b>
</center>
<hr size="2">
    <font size="2"><br>
    </font>
    <html:form action='<%= response.encodeURL("geraRelatorioPratica.do") %>'><center>
        <table width="310px" border="1" cellpadding="0" cellspacing="0"
bgcolor="#CCCCCC">
        <tr>
            <td><table width="100%" border="0">
                <tr>
                    <td colspan="3"><font size="2"><bean:message
key="relatorio.pratica.form.titulo"/></font></td>
                </tr>
                <tr>
                    <td><font size="2">
                        <bean:message key="relatorio.pratica.form.periodo"/>
                        <logic:present name="dataInicio">
                            <html:text property="dataInicio" size="10" maxlength="10" />
                        </logic:present>
                        <logic:notPresent name="dataInicio">
                            <input type="text" name="dataInicio" size="10"
maxlength="10" >
                        </logic:notPresent>
                         
                        <bean:message key="relatorio.pratica.form.ate"/>
                         
                        <logic:present name="dataFim">
                            <html:text property="dataFim" size="10" maxlength="10" />
                        </logic:present>
                        <logic:notPresent name="dataFim">
                            <input type="text" name="dataFim" size="10" maxlength="10" >
                        </logic:notPresent>
                    </font></td>
                </tr>
                <tr>
                    <td align="center"><font size="2">
                        <bean:message key="relatorio.pratica.form.formatodata"/></font>
                    </font></td>
                </tr>
```

```
                </table>
                <table width="100%" border="0">
                <tr>
                    <td align="right"><font size="2">
                        <html:link href="javascript:validar(document.forms[0]);"><font
size="2">
                            <bean:message key="relatorio.pratica.form.consultar"/>
                        </html:link></font>
                    </td>
                </tr>
                </table>
            </td>
        </tr>
        </table></center>
    </html:form>
    <center>
    <table width="100%" border="0" cellpadding="0" cellspacing="0" bgcolor="#CCCCCC">
    <tr>
        <td bordercolor="#FFFFFF" bgcolor="#FFFFFF">
            <div align="center">
            <table width="750" border="0">
            <logic:present name="lista">
                <tr>
                    <td bgcolor="#FFE5CA"><div align="left">
                    <font size="2"><b><bean:message
key="relatorio.pratica.resultado.experimento"/></b></font>
                    </div></td>
                    <td bgcolor="#FFE5CA"><div align="left">

                    <font size="2"><b><bean:message
key="relatorio.pratica.resultado.aluno"/></b></font>
                    </div></td>
                    <td bgcolor="#FFE5CA"><div align="center">
                    <font size="2"><b><bean:message
key="relatorio.pratica.resultado.data"/></b></font>
                    </div></td>
                </tr>
                <logic:iterate id="relatorio" name="lista"
type="br.ufrj.dee.elepot.util.Zebrado">
                <tr>
                    <logic:equal name="relatorio" property="zebrado" value="0"><td
bgcolor="#FFFFFF"></logic:equal>
                    <logic:notEqual name="relatorio" property="zebrado" value="0"><td
bgcolor="#FFE5CA"></logic:notEqual>
                    <div align="left">
                        <font size="2">
                            <bean:write name="relatorio"
property="objeto.numeroExperimento" />
                             &ndash; 
                            <bean:write name="relatorio"
property="objeto.nomeExperimento" />
                        </font>
                    </div></td>
                    <logic:equal name="relatorio" property="zebrado" value="0"><td
bgcolor="#FFFFFF"></logic:equal>
                    <logic:notEqual name="relatorio" property="zebrado" value="0"><td
bgcolor="#FFE5CA"></logic:notEqual>
                    <div align="left">
                        <font size="2">
                            <bean:write name="relatorio" property="objeto.nomeUsuario"
/>
                             &ndash; 
                            <bean:write name="relatorio"
property="objeto.identificacaoUsuario" />
                        </font>
                    </div></td>
                    <logic:equal name="relatorio" property="zebrado" value="0"><td
bgcolor="#FFFFFF"></logic:equal>
                    <logic:notEqual name="relatorio" property="zebrado" value="0"><td
bgcolor="#FFE5CA"></logic:notEqual>
                    <div align="center">
                        <font size="2"><bean:write name="relatorio"
property="objeto.data" /></font>
                    </div></td>
                </tr>
                </logic:iterate>
                <tr><td colspan="3"></td></tr>
```

```
                <tr>
                    <td colspan="3"><div align="left"><font size="2"><b>
                        <bean:message key="relatorio.totalderegistros"
/> <bean:write name="total" />
                    </b></font></div></td>
                </tr>
            </logic:present>
            </table>
            </div>
        </td>
    </tr>
    </table>
    <br>
    <div align="center">
        <html:link href='<%= response.encodeURL("telaManutencao.do") %>'><font size="2">
            <bean:message key="relatorio.pratica.retornar"/></font>
        </html:link>
    </div>
<%@ include file="../rodape.html" %>
</html:html>
```

senha.jsp

```
<%@ include file="valida_sessao.jsp" %>
<%@ taglib uri="http://struts.apache.org/tags-html" prefix="html" %>
<%@ taglib uri="http://struts.apache.org/tags-bean" prefix="bean" %>
<%@ taglib uri="http://struts.apache.org/tags-logic" prefix="logic" %>
<html:html>
<script language="javascript" src="javascript/funcoesGenericas.js"></script>
<script language="javascript" src="javascript/validaTrocaSenhaFormBean.js"></script>
<%@ include file="../cabecalho.html" %>
<center>
    <b><font size="3"><bean:message key="titulo.elepot"/><br>
    <bean:message key="manutencao.titulo"/> - <bean:message
key="trocasenha.titulo"/></font></b>
</center>
<hr size="2">
        <font size="2"><br>
        </font>
        <html:form method="POST" action='<%= response.encodeURL("/trocaSenha.do") %>' >
          <logic:equal parameter="telaOrigem" value="laboratorio"><html:hidden
property="telaOrigem" value="laboratorio" /></logic:equal>
          <logic:notEqual parameter="telaOrigem" value="laboratorio"><html:hidden
property="telaOrigem" value="" /></logic:notEqual>
          <table width="100%" border="0" cellpadding="0" cellspacing="0"
bgcolor="#CCCCCC">
            <tr>
              <td bordercolor="#FFFFFF" bgcolor="#FFFFFF"> <div align="center">
                  <table width="240" border="0">
                    <tr>
                      <td bgcolor="#FFFFFF"><div align="left"><font size="2">
                        <bean:message
key="trocasenha.form.novasenha"/></font></div></td>
                      <td width="41%" bgcolor="#FFFFFF"><div align="left"><font
size="2">
                          <html:password property="senha" value="" size="15"
maxlength="10" />
                          </font></div></td>
                    </tr>
                    <tr>
                      <td bgcolor="#FFFFFF"><div align="left"><font size="2">
                        <bean:message
key="trocasenha.form.redigitesenha"/></font></div></td>
                      <td bgcolor="#FFFFFF"><div align="left"><font size="2">
                          <input type="password" name="senhaConfirma" maxlength="10"
size="15" value="">
                          </font></div></td>
                    </tr>
                  </table>
                </div>
              </td>
            </tr>
          </table>
        <br>
        <div align="center">
          <html:link href="javascript:validar(document.forms[0]);"><font size="2">
```

```
            <bean:message key="trocasenha.form.trocar"/></font>
          </html:link>
           | 
          <logic:equal parameter="telaOrigem" value="laboratorio">
            <html:link href='<%= response.encodeURL("montaLab.do") %>'><font
size="2"><bean:message key="trocasenha.form.retornar"/></font></html:link>
          </logic:equal>
          <logic:notEqual parameter="telaOrigem" value="laboratorio">
            <html:link href='<%= response.encodeURL("telaManutencao.do") %>'><font
size="2"><bean:message key="trocasenha.form.retornar"/></font></html:link>
          </logic:notEqual>
        </div>
        </html:form>
<%@ include file="../rodape.html" %>
</html:html>
```

senha_ok.jsp

```
<%@ include file="valida_sessao.jsp" %>
<%@ taglib uri="http://struts.apache.org/tags-html" prefix="html" %>
<%@ taglib uri="http://struts.apache.org/tags-bean" prefix="bean" %>
<%@ taglib uri="http://struts.apache.org/tags-logic" prefix="logic" %>
<html>
<%@ include file="../cabecalho.html" %>
<center>
<b><font size="3"><bean:message key="titulo.elepot"/><br>
<bean:message key="manutencao.titulo"/> - <bean:message
key="trocasenha.titulo"/></font></b>
</center>
<hr size="2">
<font size="2"><br></font>
<p><font size="2"><bean:message key="trocasenha.ok"/></font></p>
<div align="right">
<logic:equal name="telaOrigem" value="laboratorio">
    <html:link href='<%= response.encodeURL("montaLab.do") %>'><font
size="2"><bean:message key="trocasenha.form.retornar"/></font></html:link>
</logic:equal>
<logic:notEqual name="telaOrigem" value="laboratorio">
    <html:link href='<%= response.encodeURL("telaManutencao.do") %>'><font
size="2"><bean:message key="trocasenha.form.retornar"/></font></html:link>
</logic:notEqual>
</div>
<%@ include file="../rodape.html" %>
</html>
```

turma_encerrada.jsp

```
<%@ include file="valida_sessao.jsp" %>
<%@ taglib uri="http://struts.apache.org/tags-html" prefix="html" %>
<%@ taglib uri="http://struts.apache.org/tags-bean" prefix="bean" %>
<html>
<%@ include file="../cabecalho.html" %>
<center>
<b><font size="3"><bean:message key="titulo.elepot"/><br>
<bean:message key="manutencao.titulo"/> - <bean:message
key="encerraturma.titulo"/></font></b>
</center>
<hr size="2">
<font size="2"><br></font>
<p><font size="2"><bean:message key="encerraturma.ok"/></font></p>
<div align="right">
<html:link href='<%= response.encodeURL("telaManutencao.do") %>'><font size="2">
  <bean:message key="encerraturma.voltar"/></font>
</html:link>
</div>
<%@ include file="../rodape.html" %>
</html>
```

lab_ocupado.jsp

```
<%@ include file="valida_sessao.jsp" %>
<%@ taglib uri="http://struts.apache.org/tags-html" prefix="html" %>
<%@ taglib uri="http://struts.apache.org/tags-bean" prefix="bean" %>
<html>
<%@ include file="../cabecalho.html" %>
```

```
<center>
<b><font size="3"><bean:message key="titulo.elepot"/></font></b>
</center>
<hr size="2">
        <font size="2"><br>
        </font>
        <center>
          <font size="2"><left></left></font>
        </center>
        <p align="center"><font size="2"><left><strong><font color="#FF0000">
            <bean:message key="erro.atencao"/></font></strong></left></font></p>
        <p align="center"><font size="2"><left></left></font></p>
        <p align="center"><font size="2"><left><strong><font color="#FF0000">
            <bean:message
key="erro.ocupado.mensagem"/></font></strong></left></font></p>
        <p align="center"><font size="2"><left></left></font></p>
        <p align="center"><font size="2"><left><strong><font color="#FF0000">
            <bean:message key="erro.ocupado.aviso"/></font></strong></left></font></p>
        <p><font size="2"><left></left></font><font size="2"><left></left></font></p>
        <p><font size="2"><left></left></font></p>
        <p><font size="2"><left></left></font></p>
        <p><font size="2"><left></left></font></p>
        <center>
            <html:link href='<%= response.encodeURL("montaLab.do") %>'><font
size="2"><bean:message key="erro.ocupado.voltar"/></font></html:link>
        </center>
<%@ include file="../rodape.html" %>
</html>
```

popup_descricao.jsp

```
<%@ include file="valida_sessao.jsp" %>
<%@ taglib uri="http://struts.apache.org/tags-html" prefix="html" %>
<%@ taglib uri="http://struts.apache.org/tags-bean" prefix="bean" %>
<%@ taglib uri="http://struts.apache.org/tags-logic" prefix="logic" %>
<html:html>
<center><b><bean:message key="laboratorio.popup.descricao" /></b></center>
<br>
<left><font size="2">
    <logic:present name="descricao">
        <bean:write name="descricao" filter="false" />
    </logic:present>
</font></left>
</html:html>
```

popup_instrucoes.jsp

```
<%@ include file="valida_sessao.jsp" %>
<%@ taglib uri="http://struts.apache.org/tags-html" prefix="html" %>
<%@ taglib uri="http://struts.apache.org/tags-bean" prefix="bean" %>
<%@ taglib uri="http://struts.apache.org/tags-logic" prefix="logic" %>
<html:html>
<center><b><bean:message key="laboratorio.popup.instrucoes" /></b></center>
<br>
<left><font size="2">
    <logic:present name="instrucoes">
        <bean:write name="instrucoes" filter="false" />
    </logic:present>
</font></left>
</html:html>
```

exp_1.jsp

```
<%@ include file="bloqueia_cache.jsp" %>
<%@ include file="valida_sessao.jsp" %>
<%@ taglib uri="http://struts.apache.org/tags-html" prefix="html" %>
<%@ taglib uri="http://struts.apache.org/tags-bean" prefix="bean" %>
<%@ taglib uri="http://struts.apache.org/tags-logic" prefix="logic" %>
<html:html>
<script language="javascript" src="javascript/funcoesGenericas.js"></script>
<script language="javascript" src="javascript/validaChopper.js"></script>
<%@ include file="../cabecalho.html" %>
<center>
<b><font size="3"><bean:message key="titulo.elepot"/><br>
```

```
<bean:message key="laboratorio.experiencias.titulo"/> <bean:write
name="tituloExperiencia"/></font></b>
</center><br>
<hr size="2">
<center>
<table bgcolor="#FFFFFF">
    <tr>
        <td width="400">
        <div align="center"><font size="2"><b>
            <bean:message key="laboratorio.experiencias.aluno"/> <bean:write
name="nomeAluno"/>
        </b></font></div>
        <br>
        <html:form action='<%= response.encodeURL("/comandaChopper_1.do") %>'>
          <input type="hidden" name="tituloExperiencia" value='<bean:write
name="tituloExperiencia" />' >
          <input type="hidden" name="nomeAluno" value='<bean:write name="nomeAluno" />'
>
          <table width="100%" border="0" cellpadding="0" cellspacing="0"
bgcolor="#FFFFFF">
            <tr>
              <td bordercolor="#FFFFFF" bgcolor="#FFFFFF"> <div align="center">
                <table width="350" border="0">
                  <tr>
                    <td><div align="left"><font size="2">
                        <bean:message
key="laboratorio.experiencias.comando"/></font></div></td>
                    <td width="42%"><div align="left"><font size="2">
                        <input type="radio" name="ligado" value="true" <logic:equal
name="ChopperFormBean" property="ligado" value="true" >checked</logic:equal>
><bean:message key="laboratorio.experiencias.comando.ligar"/>
                        <input type="radio" name="ligado" value="false" <logic:equal
name="ChopperFormBean" property="ligado" value="false"
>checked</logic:equal>><bean:message key="laboratorio.experiencias.comando.desligar"/>
                    </font></div></td>
                  </tr>
                  <tr>
                    <td><div align="left"><font size="2">
                        <bean:message
key="laboratorio.experiencias.tipodeconversor"/></font></div></td>
                    <td><div align="left"><font size="2">
                        <input type="radio" name="tipoChopper" value="BUCK" <logic:equal
name="ChopperFormBean" property="tipoChopper" value="BUCK" >checked</logic:equal>
><bean:message key="laboratorio.experiencias.tipodeconversor.buck"/>
                        <input type="radio" name="tipoChopper" value="BOOST"
<logic:equal name="ChopperFormBean" property="tipoChopper" value="BOOST"
>checked</logic:equal>><bean:message
key="laboratorio.experiencias.tipodeconversor.boost"/>
                    </font></div></td>
                  </tr>
                  <tr>
                    <td><div align="left"><font size="2">
                      <bean:message
key="laboratorio.experiencias.frequenciachaveamento"/><br>     
<bean:message key="laboratorio.experiencias.frequenciachaveamento.intervalo.chopper"/>
                    </font></div></td>
                    <td><div align="left"><font size="2">
                        <html:text name="ChopperFormBean" property="fs" size="5"
maxlength="4" /> <bean:message key="laboratorio.experiencias.unidades.hz"/>
                    </font></div></td>
                  </tr>
                  <tr>
                    <td><div align="left"><font size="2">
                      <bean:message key="laboratorio.experiencias.ciclodetrabalho"/>
                    </font></div></td>
                    <td><div align="left"><font size="2">
                        <html:text name="ChopperFormBean" property="dc" size="6"
maxlength="6" /> <bean:message key="laboratorio.experiencias.unidades.percentual"/>
                    </font></div></td>
                  </tr>
                  </table>
                </div>
              </td>
            </tr>
          </table>
        <br>
        <div align="center">
```

```
          <html:link href="javascript:validar(document.forms[0]);"><font size="2">
            <bean:message key="laboratorio.enviarcomando"/></font>
          </html:link>
           | 
          <html:link href='<%= response.encodeURL("montaLab.do") %>'><font size="2">
            <bean:message key="laboratorio.voltar"/></font>
          </html:link>
        </div>
        </html:form>
        </td>
    </tr>
    <logic:present name="parametros">
      <tr><td align="center"><font size="2">
        <bean:message key="laboratorio.experiencias.paramentrosdoconversor.titulo" />
        <table width="200" border="1" cellpadding="0" cellspacing="0" bgcolor="#CCCCCC">
        <tr>
          <td><div align="center"><font size="2">
            <bean:message key="laboratorio.experiencias.paramentrosdoconversor.ts" />
             = 
            <bean:write name="parametros" property="tsString" /> <bean:message
key="laboratorio.experiencias.unidades.ms" />
            <br>
            <bean:message key="laboratorio.experiencias.paramentrosdoconversor.ton" />
             = 
            <bean:write name="parametros" property="tonString" /> <bean:message
key="laboratorio.experiencias.unidades.ms" />
            <br>
            <bean:message key="laboratorio.experiencias.paramentrosdoconversor.toff" />
             = 

            <bean:write name="parametros" property="toffString" /> <bean:message
key="laboratorio.experiencias.unidades.ms" />
            <br>
            <bean:message key="laboratorio.experiencias.paramentrosdoconversor.vovi" />
             = 
            <bean:write name="parametros" property="ganhoString" />
          </font></div></td>
        </tr>
        </table>
      </font></td></tr>
    </logic:present>
</table>
</center>
<font size="2"><br></font>
<%@ include file="../rodape.html" %>
</html:html>
```

exp_2.jsp

```
<%@ include file="bloqueia_cache.jsp" %>
<%@ include file="valida_sessao.jsp" %>
<%@ taglib uri="http://struts.apache.org/tags-html" prefix="html" %>
<%@ taglib uri="http://struts.apache.org/tags-bean" prefix="bean" %>
<%@ taglib uri="http://struts.apache.org/tags-logic" prefix="logic" %>
<html:html>
<script language="javascript" src="javascript/funcoesGenericas.js"></script>
<script language="javascript" src="javascript/validaChopper.js"></script>
<%@ include file="../cabecalho.html" %>
<center>
<b><font size="3"><bean:message key="titulo.elepot"/><br>
<bean:message key="laboratorio.experiencias.titulo"/> <bean:write
name="tituloExperiencia"/></font></b>
</center><br>
<hr size="2">
<center>
<table bgcolor="#FFFFFF">
    <tr>
        <td width="400">
        <div align="center"><font size="2"><b>
            <bean:message key="laboratorio.experiencias.aluno"/> <bean:write
name="nomeAluno"/>
        </b></font></div>
        <br>
        <html:form action='<%= response.encodeURL("/comandaChopper_2.do") %>'>
          <input type="hidden" name="tituloExperiencia" value='<bean:write
name="tituloExperiencia" />' >
```

```
          <input type="hidden" name="nomeAluno" value='<bean:write name="nomeAluno" />'
>
          <html:hidden property="tipoChopper" value="BUCKBOOST"/>
          <table width="100%" border="0" cellpadding="0" cellspacing="0"
bgcolor="#FFFFFF">
            <tr>
              <td bordercolor="#FFFFFF" bgcolor="#FFFFFF"> <div align="center">
                <table width="350" border="0">
                  <tr>
                    <td><div align="left"><font size="2">
                        <bean:message
key="laboratorio.experiencias.comando"/></font></div></td>
                    <td width="42%"><div align="left"><font size="2">
                        <input type="radio" name="ligado" value="true" <logic:equal
name="ChopperFormBean" property="ligado" value="true" >checked</logic:equal>
><bean:message key="laboratorio.experiencias.comando.ligar"/>
                        <input type="radio" name="ligado" value="false" <logic:equal
name="ChopperFormBean" property="ligado" value="false"
>checked</logic:equal>><bean:message key="laboratorio.experiencias.comando.desligar"/>
                    </font></div></td>
                  </tr>
                  <tr>
                    <td><div align="left"><font size="2">
                      <bean:message
key="laboratorio.experiencias.frequenciachaveamento"/><br>     
<bean:message key="laboratorio.experiencias.frequenciachaveamento.intervalo.chopper"/>
                    </font></div></td>
                    <td><div align="left"><font size="2">
                        <html:text name="ChopperFormBean" property="fs" size="5"
maxlength="4" /> <bean:message key="laboratorio.experiencias.unidades.hz"/>
                    </font></div></td>
                  </tr>
                  <tr>
                    <td><div align="left"><font size="2">
                      <bean:message
key="laboratorio.experiencias.ciclodetrabalho"/></font></div></td>
                    <td><div align="left"><font size="2">
                        <html:text name="ChopperFormBean" property="dc" size="6"
maxlength="6" /> <bean:message key="laboratorio.experiencias.unidades.percentual"/>
                    </font></div></td>
                  </tr>
                  </table>
                </div>
              </td>
            </tr>
          </table>
        <br>
        <div align="center">
          <html:link href="javascript:validar(document.forms[0]);"><font size="2">
            <bean:message key="laboratorio.enviarcomando"/></font>
          </html:link>
           | 
          <html:link href='<%= response.encodeURL("montaLab.do") %>'><font size="2">
            <bean:message key="laboratorio.voltar"/></font>
          </html:link>
        </div>
        </html:form>
        </td>
    </tr>
    <logic:present name="parametros">
      <tr><td align="center"><font size="2">
        <bean:message key="laboratorio.experiencias.paramentrosdoconversor.titulo" />
        <table width="200" border="1" cellpadding="0" cellspacing="0" bgcolor="#CCCCCC">
        <tr>
          <td><div align="center"><font size="2">
            <bean:message key="laboratorio.experiencias.paramentrosdoconversor.ts" />
             = 
            <bean:write name="parametros" property="tsString" /> <bean:message
key="laboratorio.experiencias.unidades.ms" />
            <br>
            <bean:message key="laboratorio.experiencias.paramentrosdoconversor.ton" />
             = 
            <bean:write name="parametros" property="tonString" /> <bean:message
key="laboratorio.experiencias.unidades.ms" />
            <br>
            <bean:message key="laboratorio.experiencias.paramentrosdoconversor.toff" />
             = 
```

```
            <bean:write name="parametros" property="toffString" /> <bean:message
key="laboratorio.experiencias.unidades.ms" />
            <br>
            <bean:message key="laboratorio.experiencias.paramentrosdoconversor.vovi" />
             = 
            <bean:write name="parametros" property="ganhoString" />
          </font></div></td>
        </tr>
        </table>
      </font></td></tr>
    </logic:present>
</table>
</center>
<font size="2"><br></font>
<%@ include file="../rodape.html" %>
</html:html>
```

exp_3.jsp

```
<%@ include file="bloqueia_cache.jsp" %>
<%@ include file="valida_sessao.jsp" %>
<%@ taglib uri="http://struts.apache.org/tags-html" prefix="html" %>
<%@ taglib uri="http://struts.apache.org/tags-bean" prefix="bean" %>
<%@ taglib uri="http://struts.apache.org/tags-logic" prefix="logic" %>
<html:html>
<script language="javascript" src="javascript/funcoesGenericas.js"></script>
<script language="javascript" src="javascript/validaInversor.js"></script>
<%@ include file="../cabecalho.html" %>
<center>
<b><font size="3"><bean:message key="titulo.elepot"/><br>
<bean:message key="laboratorio.experiencias.titulo"/> <bean:write
name="tituloExperiencia"/></font></b>
</center><br>
<hr size="2">
<center>
<table bgcolor="#FFFFFF">
    <tr>
        <td width="400">
        <div align="center"><font size="2"><b>
            <bean:message key="laboratorio.experiencias.aluno"/> <bean:write
name="nomeAluno"/>
        </b></font></div>
        <br>
        <html:form action='<%= response.encodeURL("/comandaInversor_3.do") %>'>
          <input type="hidden" name="tituloExperiencia" value='<bean:write
name="tituloExperiencia" />' >
          <input type="hidden" name="nomeAluno" value='<bean:write name="nomeAluno" />'
>
          <table width="100%" border="0" cellpadding="0" cellspacing="0"
bgcolor="#FFFFFF">
            <tr>
              <td bordercolor="#FFFFFF" bgcolor="#FFFFFF"> <div align="center">
                <table width="350" border="0">
                  <tr>
                    <td><div align="left"><font size="2">
                        <bean:message
key="laboratorio.experiencias.comando"/></font></div></td>
                    <td width="42%"><div align="left"><font size="2">
                        <input type="radio" name="ligado" value="true" <logic:equal
name="InversorFormBean" property="ligado" value="true" >checked</logic:equal>
><bean:message key="laboratorio.experiencias.comando.ligar"/>
                        <input type="radio" name="ligado" value="false" <logic:equal
name="InversorFormBean" property="ligado" value="false"
>checked</logic:equal>><bean:message key="laboratorio.experiencias.comando.desligar"/>
                    </font></div></td>
                  </tr>
                  <tr>
                    <td><div align="left"><font size="2">
                      <bean:message
key="laboratorio.experiencias.frequenciachaveamento"/><br>     
<bean:message key="laboratorio.experiencias.frequenciachaveamento.intervalo.inversor"/>
                    </font></div></td>
                    <td><div align="left"><font size="2">
                        <html:text name="InversorFormBean" property="fs" size="5"
maxlength="4" /> <bean:message key="laboratorio.experiencias.unidades.hz"/>
                    </font></div></td>
```

```
                </tr>
                <tr>
                  <td><div align="left"><font size="2">
                    <bean:message
key="laboratorio.experiencias.frequenciareferencia"/><br>     <
bean:message key="laboratorio.experiencias.frequenciareferencia.maximo"/>
                  </font></div></td>
                  <td><div align="left"><font size="2">
                      <html:text name="InversorFormBean" property="fref" size="5"
maxlength="3" /> <bean:message key="laboratorio.experiencias.unidades.hz"/>
                  </font></div></td>
                </tr>
                <tr>
                  <td><div align="left"><font size="2">
                    <bean:message
key="laboratorio.experiencias.tensaoreferencia"/><br>     <bean
:message key="laboratorio.experiencias.tensaoreferencia.maximo"/>
                  </font></div></td>
                  <td><div align="left"><font size="2">
                      <html:text name="InversorFormBean" property="vref" size="5"
maxlength="4" /> <bean:message key="laboratorio.experiencias.unidades.pu"/>
                  </font></div></td>
                </tr>
                </table>
              </div>
            </td>
          </tr>
        </table>
      <br>
      <div align="center">
        <html:link href="javascript:validar(document.forms[0]);"><font size="2">
          <bean:message key="laboratorio.enviarcomando"/></font>
        </html:link>
         | 
        <html:link href='<%= response.encodeURL("montaLab.do") %>'><font size="2">
          <bean:message key="laboratorio.voltar"/></font>
        </html:link>
      </div>
      </html:form>
      </td>
  </tr>
  <logic:present name="parametros">
    <tr><td align="center"><font size="2">
      <bean:message key="laboratorio.experiencias.paramentrosdoconversor.titulo" />
      <table width="200" border="1" cellpadding="0" cellspacing="0" bgcolor="#CCCCCC">
      <tr>
        <td><div align="center"><font size="2">
          <bean:message key="laboratorio.experiencias.paramentrosdoconversor.ts" />
           = 
          <bean:write name="parametros" property="tsString" /> <bean:message
key="laboratorio.experiencias.unidades.ms" />
          <br>
          <bean:message key="laboratorio.experiencias.paramentrosdoconversor.mf" />
           = 
          <bean:write name="parametros" property="mfString" />
          <br>
          <bean:message key="laboratorio.experiencias.paramentrosdoconversor.ma" />
           = 
          <bean:write name="parametros" property="maString" />
        </font></div></td>
      </tr>
      </table>
    </font></td></tr>
  </logic:present>
</table>
</center>
<font size="2"><br></font>
<%@ include file="../rodape.html" %>
</html:html>
```

exp_4.jsp

```
<%@ include file="bloqueia_cache.jsp" %>
<%@ include file="valida_sessao.jsp" %>
<%@ taglib uri="http://struts.apache.org/tags-html" prefix="html" %>
<%@ taglib uri="http://struts.apache.org/tags-bean" prefix="bean" %>
<%@ taglib uri="http://struts.apache.org/tags-logic" prefix="logic" %>
<html:html>
<script language="javascript" src="javascript/funcoesGenericas.js"></script>
<script language="javascript" src="javascript/validaRetificador.js"></script>
<%@ include file="../cabecalho.html" %>
<center>
<b><font size="3"><bean:message key="titulo.elepot"/><br>
<bean:message key="laboratorio.experiencias.titulo"/> <bean:write
name="tituloExperiencia"/></font></b>
</center><br>
<hr size="2">
<center>
<table bgcolor="#FFFFFF">
    <tr>
        <td width="400">
        <div align="center"><font size="2"><b>
            <bean:message key="laboratorio.experiencias.aluno"/> <bean:write
name="nomeAluno"/>
        </b></font></div>
        <br>
        <html:form action='<%= response.encodeURL("/comandaRetificador_4.do") %>'>
          <input type="hidden" name="tituloExperiencia" value='<bean:write
name="tituloExperiencia" />' >
          <input type="hidden" name="nomeAluno" value='<bean:write name="nomeAluno" />'
>
          <table width="100%" border="0" cellpadding="0" cellspacing="0"
bgcolor="#FFFFFF">
            <tr>
              <td bordercolor="#FFFFFF" bgcolor="#FFFFFF"> <div align="center">
                <table width="350" border="0">
                  <tr>
                    <td><div align="left"><font size="2">
                        <bean:message
key="laboratorio.experiencias.comando"/></font></div></td>
                    <td width="42%"><div align="left"><font size="2">
                        <input type="radio" name="ligado" value="true" <logic:equal
name="RetificadorFormBean" property="ligado" value="true" >checked</logic:equal>
><bean:message key="laboratorio.experiencias.comando.ligar"/>
                        <input type="radio" name="ligado" value="false" <logic:equal
name="RetificadorFormBean" property="ligado" value="false"
>checked</logic:equal>><bean:message key="laboratorio.experiencias.comando.desligar"/>
                    </font></div></td>
                  </tr>
                  <tr>
                    <td><div align="left"><font size="2">
                      <bean:message
key="laboratorio.experiencias.angulodisparo"/><br>     <bean:me
ssage key="laboratorio.experiencias.angulodisparo.intervalo"/>
                    </font></div></td>
                    <td><div align="left"><font size="2">
                        <html:text name="RetificadorFormBean" property="alpha" size="6"
maxlength="5" /> <bean:message key="laboratorio.experiencias.unidades.graus"/>
                    </font></div></td>
                  </tr>
                  <tr>
                    <td><div align="left"><font size="2">
                        <bean:message key="laboratorio.experiencias.frequenciarede"/>
                    </font></div></td>
                    <td><div align="left"><font size="2">
                        <html:select name="RetificadorFormBean" property="freq">
                            <html:option value="50"> 50 Hz</html:option>
                            <html:option value="60"> 60 Hz</html:option>
                        </html:select>
                    </font></div></td>
                  </tr>
                  </table>
                </div>
              </td>
            </tr>
          </table>
        <br>
```

```
        <div align="center">
          <html:link href="javascript:validar(document.forms[0]);"><font size="2">
            <bean:message key="laboratorio.enviarcomando"/></font>
          </html:link>
           | 
          <html:link href='<%= response.encodeURL("montaLab.do") %>'><font size="2">
            <bean:message key="laboratorio.voltar"/></font>
          </html:link>
        </div>
        </html:form>
        </td>
    </tr>
    <logic:present name="parametros">
      <tr><td align="center"><font size="2">
        <bean:message key="laboratorio.experiencias.paramentrosdoconversor.titulo" />
        <table width="200" border="1" cellpadding="0" cellspacing="0" bgcolor="#CCCCCC">
        <tr>
          <td><div align="center"><font size="2">
            <bean:message key="laboratorio.experiencias.paramentrosdoconversor.vo" />
             = 
            <bean:write name="parametros" property="voString" /> <bean:message
key="laboratorio.experiencias.unidades.pu" />
          </font></div></td>
        </tr>
        </table>
      </font></td></tr>
    </logic:present>
</table>
</center>
<font size="2"><br></font>
<%@ include file="../rodape.html" %>
</html:html>
```

exp_5.jsp

```
<%@ include file="bloqueia_cache.jsp" %>
<%@ include file="valida_sessao.jsp" %>
<%@ taglib uri="http://struts.apache.org/tags-html" prefix="html" %>
<%@ taglib uri="http://struts.apache.org/tags-bean" prefix="bean" %>
<%@ taglib uri="http://struts.apache.org/tags-logic" prefix="logic" %>
<html:html>
<script language="javascript" src="javascript/funcoesGenericas.js"></script>
<script language="javascript" src="javascript/validaRetificador.js"></script>
<%@ include file="../cabecalho.html" %>
<center>
<b><font size="3"><bean:message key="titulo.elepot"/><br>
<bean:message key="laboratorio.experiencias.titulo"/> <bean:write
name="tituloExperiencia"/></font></b>
</center><br>
<hr size="2">
<center>
<table bgcolor="#FFFFFF">
    <tr>
        <td width="400">
        <div align="center"><font size="2"><b>
            <bean:message key="laboratorio.experiencias.aluno"/> <bean:write
name="nomeAluno"/>
        </b></font></div>
        <br>
        <html:form action='<%= response.encodeURL("/comandaRetificador_5.do") %>'>
          <input type="hidden" name="tituloExperiencia" value='<bean:write
name="tituloExperiencia" />' >
          <input type="hidden" name="nomeAluno" value='<bean:write name="nomeAluno" />'
>
          <table width="100%" border="0" cellpadding="0" cellspacing="0"
bgcolor="#FFFFFF">
            <tr>
              <td bordercolor="#FFFFFF" bgcolor="#FFFFFF"> <div align="center">
                <table width="350" border="0">
                  <tr>
                    <td><div align="left"><font size="2">
                        <bean:message
key="laboratorio.experiencias.comando"/></font></div></td>
                    <td width="42%"><div align="left"><font size="2">
```

```
                        <input type="radio" name="ligado" value="true" <logic:equal
name="RetificadorFormBean" property="ligado" value="true" >checked</logic:equal>
><bean:message key="laboratorio.experiencias.comando.ligar"/>
                        <input type="radio" name="ligado" value="false" <logic:equal
name="RetificadorFormBean" property="ligado" value="false"
>checked</logic:equal>><bean:message key="laboratorio.experiencias.comando.desligar"/>
                    </font></div></td>
                  </tr>
                  <tr>
                    <td><div align="left"><font size="2">
                      <bean:message
key="laboratorio.experiencias.angulodisparo"/><br>     <bean:me
ssage key="laboratorio.experiencias.angulodisparo.intervalo"/>
                    </font></div></td>
                    <td><div align="left"><font size="2">
                        <html:text name="RetificadorFormBean" property="alpha" size="6"
maxlength="5" /> <bean:message key="laboratorio.experiencias.unidades.graus"/>
                    </font></div></td>
                  </tr>
                  <tr>
                    <td><div align="left"><font size="2">
                        <bean:message key="laboratorio.experiencias.frequenciarede"/>
                    </font></div></td>
                    <td><div align="left"><font size="2">
                        <html:select name="RetificadorFormBean" property="freq">
                            <html:option value="50"> 50 Hz</html:option>
                            <html:option value="60"> 60 Hz</html:option>
                        </html:select>
                    </font></div></td>
                  </tr>
                  </table>
                </div>
              </td>
            </tr>
          </table>
        <br>
        <div align="center">
          <html:link href="javascript:validar(document.forms[0]);"><font size="2">
            <bean:message key="laboratorio.enviarcomando"/></font>
          </html:link>
           | 
          <html:link href='<%= response.encodeURL("montaLab.do") %>'><font size="2">
            <bean:message key="laboratorio.voltar"/></font>
          </html:link>
        </div>
        </html:form>
        </td>
    </tr>
    <logic:present name="parametros">
      <tr><td align="center"><font size="2">
        <bean:message key="laboratorio.experiencias.paramentrosdoconversor.titulo" />
        <table width="200" border="1" cellpadding="0" cellspacing="0" bgcolor="#CCCCCC">
        <tr>
          <td><div align="center"><font size="2">
            <bean:message key="laboratorio.experiencias.paramentrosdoconversor.vo" />
             = 
            <bean:write name="parametros" property="voString" /> <bean:message
key="laboratorio.experiencias.unidades.pu" />
          </font></div></td>
        </tr>
        </table>
      </font></td></tr>
    </logic:present>
</table>
</center>
<font size="2"><br></font>
<%@ include file="../rodape.html" %>
</html:html>
```

C.14 – Arquivos Javascript

funcoesGenericas.js

```
function isEspaco(caractere) {
    var espaco = " \t\n\r\f";
    return (espaco.indexOf(caractere) != -1);
}
function ltrim(str) {
    for (var k = 0; k < str.length && isEspaco(str.charAt(k)); k++);
    return str.substring(k, str.length);
}
function rtrim(str) {
    for (var j = str.length - 1; j >= 0 && isEspaco(str.charAt(j)); j--);
    return str.substring(0, j + 1);
}
function trim(str) {
    return ltrim(rtrim(str));
}
```

validaAlunoFormBean.js

```
function validarInclusao(formulario) {
    if ((nomeOk(formulario.nome) == true) && (loginOk(formulario.login) == true)) {
        return formulario.submit();
    }
}
function validarAlteracao(formulario) {
    if ((nomeOk(formulario.nome) == true) && (loginOk(formulario.login) == true) &&
(senhaOk(formulario.novaSenha) == true)) {
        return formulario.submit();
    }
}
function loginOk(login) {
    var str = trim(login.value);
    if (str.length > 0) {
        return true;
    } else
        alert('Informe o login do aluno.');
}
function senhaOk(senha) {
    var str = trim(senha.value);
    if (str.length > 0) {
        return true;
    } else
        alert('Informe a senha do aluno.');
}
function nomeOk(nome) {
    var str = trim(nome.value);
    if (str.length > 0) {
        return true;
    } else
        alert('Informe o nome do aluno.');
}
```

validaExperimentoFormBean.js

```
function validar(formulario) {
    if ((numeroOk(formulario.numero) == true) && (nomeOk(formulario.nome) == true) &&
(objetivoOk(formulario.objetivo) == true)) {
        return formulario.submit();
    }
}
function numeroOk(numero) {
    var str = trim(numero.value);
    if (str.length > 0) {
        return true;
    } else
        alert('Informe o número do experimento.');
}
function nomeOk(nome) {
    var str = trim(nome.value);
    if (str.length > 0) {
        return true;
    } else
        alert('Informe o nome do experimento.');
```

```
}
function objetivoOk(objetivo) {
    var str = trim(objetivo.value);
    if (str.length > 0) {
        return true;
    } else
        alert('Informe o objetivo do experimento.');
}
```

validaConfiguracaoFormBean.js

```
function validar(formulario) {
    if ((portaSerialOk(formulario.portaSerial) == true) &&
(tempoSessaoOk(formulario.tempoSessao) == true) && (portaSerialConfereOk(formulario) ==
true)) {
        return formulario.submit();
    }
}
function portaSerialOk(portaSerial) {
    var str = trim(portaSerial.value);
    if (str.length > 0) {
        return true;
    } else
        alert('Informe a porta serial.');
}
function tempoSessaoOk(tempoSessao) {
    var str = trim(tempoSessao.value);
    if (str.length > 0) {
        return true;
    } else
        alert('Informe o tempo de sessão.');
}
function portaSerialConfereOk(formulario) {
    var str1 = trim(formulario.sistemaOperacional.value);
    var str2 = trim(formulario.portaSerial.value);
    if ((str1 == "LINUX" && str2.substring(0, 4) == "/dev") || (str1 == "WINDOWS" &&
str2.substring(0, 3) == "COM") || (str1 == "WINDOWS" && str2.substring(0, 3) == "com"))
{
        return true;
    } else
        alert('Porta serial inválida. Confira o sistema operacional escolhido e
verifique se a porta serial foi digitada corretamente.');
}
```

validaRelatorioPraticaFormBean.js

```
function validar(formulario) {
    if ((dataInicioOk(formulario.dataInicio) == true) && (dataFimOk(formulario.dataFim)
== true)) {
        return formulario.submit();
    }
}
function dataInicioOk(dataInicio) {
    var str = trim(dataInicio.value);
    if (str.length > 0) {
        return true;
    } else
        alert('Informe a data inicial do período.');
}
function dataFimOk(dataFim) {
    var str = trim(dataFim.value);
    if (str.length > 0) {
        return true;
    } else
        alert('Informe a data final do período.');
}
```

validaTrocaSenhaFormBean.js

```
function validar(formulario) {
    if ((senhaOk(formulario.senha) == true) &&
(senhaConfirmaOk(formulario.senhaConfirma) == true) && (senhaConfereOk(formulario) ==
true)) {
        return formulario.submit();
    }
```

```
}
function senhaOk(senha) {
    var str = trim(senha.value);
    if (str.length > 0) {
        return true;
    } else
        alert('Informe a nova senha.');
}
function senhaConfirmaOk(senhaConfirma) {
    var str = trim(senhaConfirma.value);
    if (str.length > 0) {
        return true;
    } else
        alert('Redigite a nova senha.');
}
function senhaConfereOk(formulario) {
    var str1 = trim(formulario.senha.value);
    var str2 = trim(formulario.senhaConfirma.value);
    if (str1 == str2) {
        return true;
    } else
        alert('As senhas não conferem. Redigite.');
}
```

validaChopper.js

```
function validar(formulario) {
    if (formulario.ligado[0].checked) {
        if (dcOk(formulario.dc) == true) {
            return formulario.submit();
        }
    } else {
        return formulario.submit();
    }
}
function dcOk(dc) {
    var str = trim(dc.value);
    if (str.length > 0) {
        return true;
    } else
        alert('Informe o ciclo de trabalho.');
}
```

validaInversor.js

```
function validar(formulario) {
    if (formulario.ligado[0].checked) {
        if ((frefOk(formulario.fref) == true) && (vrefOk(formulario.vref) == true)) {
            return formulario.submit();
        }
    } else {
        return formulario.submit();
    }
}
function frefOk(fref) {
    var str = trim(fref.value);
    if (str.length > 0) {
        return true;
    } else
        alert('Informe a frequência de referência.');
}
function vrefOk(vref) {
    var str = trim(vref.value);
    if (str.length > 0) {
        return true;
    } else
        alert('Informe a tensão de referência.');
}
```

validaRetificador.js

```
function validar(formulario) {
    if (formulario.ligado[0].checked) {
        if (alphaOk(formulario.alpha) == true) {
            return formulario.submit();
```

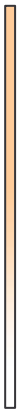
```
        }
    } else {
        return formulario.submit();
    }
}
function alphaOk(alpha) {
    var str = trim(alpha.value);
    if (str.length > 0) {
        return true;
    } else
        alert('Informe o ângulo de disparo.');
}
```

C.15 – Imagens

cabecalho.jpg

minerva.jpg

pano_inferior.jpg



pano_superior.jpg