SISTEMA INTEGRADO PARA DESENVOLVIMENTO E EXECUÇÃO EM TEMPO REAL DE ALGORITMOS DE PROTEÇÃO DE SISTEMAS ELÉTRICOS

Renato Machado Monaro*
monaro@usp.br

Raphael Philipe Mendes da Silva*
rapphil@usp.br

José Carlos de Melo Vieira Júnior*
jcarlos@sc.usp.br

Denis Vinicius Coury*
coury@sc.usp.br

*Laboratório de Sistemas de Energia Elétrica-LSEE
Escola de Engenharia de São Carlos-EESC
Universidade de São Paulo-USP
São Carlos, São Paulo, Brasil

ABSTRACT

Integrated system for development and real time execution of protection algorithm in electrical systems
This article presents the development of an integrated system of hardware and software, whose objective is to serve as a platform for developing and executing new power system protection algorithms on a real-time environment. The proposed system aims to reduce development time of new protection functions, enabling real-time analysis in opposition of only computer simulations, as can be typically found in technical literature. In this work, generator protection functions were implemented and tested, and the results were compared with a commercial relay, showing the feasibility and dependability of the proposed system.

KEYWORDS: Protection, Generators, PC104, Embedded Systems, Real Time Execution, RTAI, Linux, Comedi

RESUMO

O artigo apresenta o desenvolvimento de um sistema integrado de *software* e *hardware* cujo objetivo é servir de plataforma para desenvolver e executar em tempo real algoritmos de proteção aplicados a sistemas elétricos. O objetivo deste sistema é reduzir o tempo de desenvolvimento de novas funções de proteção, possibilitando a execução de testes em tempo real e não por meio de simulações computacionais, como é tipicamente encontrado na literatura. Neste trabalho as funções de proteção de gerador que foram implementadas e testadas são descritas, e os resultados obtidos comparados com aqueles extraídos através de testes em um relé comercial, evidenciando a viabilidade do sistema desenvolvido.

PALAVRAS-CHAVE: Proteção, Geradores, PC104, Sistemas Embarcados, Execução em Tempo Real, RTAI, Linux, Comedi

1 INTRODUÇÃO

O aumento da complexidade dos Sistemas Elétricos de Potência exige o refinamento da proteção de seus componentes de modo a garantir a confiabilidade e estabilidade dos mesmos, beneficiando a operação desses sistemas como um todo. Desta maneira os relés de proteção assumem cada vez mais importância, devido à sua capacidade de isolar faltas com precisão e rapidez (Coury et al., 2007).

A busca da melhoria da proteção leva naturalmente ao desenvolvimento de novas técnicas de proteção e/ou ao aper-

Artigo submetido em 13/04/2011 (Id.: 1318)
Revisado em 01/08/2011, 22/08/2011
Aceito sob recomendação do Editor Associado Prof. Antonio Carlos Zambroni de Souza

feiçoamento daquelas existentes. O aumento do interesse nessa área é constatado pelo crescente número de publicações que abordam esses tópicos. Como exemplo, tem-se Sidhu *et. al.* que, em suas sínteses bibliográficas entre 2004 e 2007, mostram um grande número de artigos relacionados à área, confirmando assim o interesse mencionado (Sidhu et al., 2006; Sidhu et al., 2007; Sidhu et al., 2008; Sidhu et al., 2010). No entanto, a maioria dessas propostas não discute a aplicabilidade das mesmas em campo, ficando restritas a simulações computacionais. Neste contexto, torna-se relevante o desenvolvimento de esquemas que permitam a implementação em *hardware* e *software* de novos algoritmos de proteção, com a finalidade de validar seu desempenho e averiguar sua aplicabilidade em casos práticos. Contudo, o desenvolvimento de sistemas para esta finalidade normalmente requer um profundo conhecimento de suas arquiteturas, as quais são específicas e particulares da família de microprocessadores utilizada. Como consequência, isso pode levar a um longo tempo de desenvolvimento, tornando desestimulante esse tipo implementação. Adicionalmente, o uso exclusivo do ambiente de simulação computacional pode levar à proposição de algoritmos de proteção suficientemente complexos, de modo a inviabilizar sua implementação em situações práticas.

Por outro lado, existem trabalhos que abordam o desenvolvimento de dispositivos de proteção dispondo de componentes específicos de *hardware*. McLaren et al. (1994) propõem a utilização de um *hardware* genérico para a implementação de relés de proteção. A utilização de módulos genéricos de *hardware* possibilita a extensão e expansão das capacidades deste conforme as necessidades. Essa propriedade possibilita que um engenheiro de proteção com algum conhecimento de *hardware* projete funções de proteção específicas a suas necessidades. O sistema projetado é composto por um elemento *Digital Signal Processor* (DSP) e um computador industrial 386. Em certo ponto do desenvolvimento, foi necessária a utilização de código de máquina de baixo nível (*Assembly*) para aumentar o desempenho do sistema proposto, dificultando sua utilização por engenheiros de proteção.

O trabalho de Ruan and Lin (2005) apresenta o teste da utilização de um novo modelo de DSPs de baixo custo para a implementação de funções de proteção. A comparação com outros dispositivos foi feita através da implementação de ferramentas de extração fasorial utilizando a Transformada de Fourier (TF) e o método dos mínimos quadrados. Constatou-se a superioridade do novo modelo de DSPs perante os utilizados tradicionalmente para implementação de relés de proteção.

O uso de *Field Programmable Analogic Array*s (FPAAs) em conjunto com DSPs para a implementação de relés de distância com característica *mho* é proposto por Dadash Zadeh et al. (2009). Tal abordagem possibilita a obtenção das características de alta velocidade de atuação de dispositivos de estado sólido em conjunto com o monitoramento e comunicação dos dispositivos digitais. Além disso, deve-se ressaltar a flexibilidade obtida pela utilização de elementos analógicos programáveis, pois possibilita a mudança dos circuitos eletrônicos diretamente no dispositivo. Foram necessárias muitas unidades FPAAs para o desenvolvimento da plataforma devido à baixa densidade de elementos analógicos presentes em cada unidade.

Os trabalhos supracitados utilizam essencialmente DSPs para executar as operações necessárias para a proteção de Sistema Elétrico de Potências (SEPs). No entanto, implementar uma função de processamento digital de sinal como um *software* em um DSP frequentemente requer esforços consideráveis de projeto e verificação (Stranneby, 2001).

Com intuito de facilitar e diminuir o tempo de desenvolvimento deste processo, este artigo propõe um conjunto de *hardware* e *software* projetado para servir de plataforma para desenvolvimento e execução em tempo real de novos algoritmos de proteção. O sistema proposto é composto por uma placa de processamento ligada a uma placa de aquisição de dados que seguem o padrão PC104 (PC/104 Consortium, 2009) e pelo emprego de um Sistema Operacional (SO) baseado no Linux. Logo, por utilizar placas de computadores e um SO de uso comum, os procedimentos de desenvolvimento e testes de novas funções de proteção são facilitados, pois pode-se empregar linguagens de programação de alto nível largamente utilizadas em microcomputadores convencionais, tais como C e C++. Estas características se destacam como as vantagens do sistema proposto em relação àqueles baseados em DSPs e FPAAs mencionados anteriormente. Para verificar a aplicabilidade e desempenho do sistema proposto, foram implementados algoritmos de proteção de geradores síncronos, a saber: proteção diferencial, proteção de sobrecorrente de fase instantânea e proteção de sobrecorrente de neutro temporizada. Um SEP foi modelado usando o *Alternative Transients Program* (ATP), e sobre ele foram simulados diversos casos de faltas. As simulações serviram para prover dados de falta ao sistema proposto, por meio de uma caixa de geração de sinais elétricos. Com isso, os algoritmos de proteção foram avaliados executando-se testes de desempenho. Adicionalmente, os resultados foram comparados com a resposta de um relé comercial, com as mesmas funções de proteção, mesmos ajustes e para a mesma situação de falta.

2 A PLATAFORMA PC104

O PC104 (ou PC/104) é um padrão de computador embarcado controlado pelo PC/104 Consortium (2009) que define um padrão de formato e de barramento. PC/104 é destinado a

aplicações de computação embarcada que dependam de uma aquisição de dados confiáveis em ambientes extremos.

O formato PC104 foi originalmente concebido pela empresa Ampro em 1987 e mais tarde padronizado pelo *PC/104 Consortium* em 1992. Uma tentativa de padronização correspondente ao PC104 pelo IEEE foi feita com o IEEE P996.1, mas nunca foi ratificada.

Conforme mencionado anteriormente, a plataforma PC104 permite a implementação de computação embarcada em ambientes industriais, devido às seguintes características: formato reduzido, permitindo assim menor área de exposição a eventuais danos; ausência de partes móveis, visto que não há disco rígido e sim um dispositivo de armazenamento de estado sólido (*Solid State Disk*); sistema de refrigeração somente com dissipadores (*fanless*).

Além dessas características que provêm robustez à plataforma, esta permite o uso de sistemas operacionais convencionais, flexibilizando seu uso. Também é possível a utilização de SOs de tempo real para suprir a necessidade de aplicações de controle que têm requisitos de tempo. Tais características fazem da plataforma PC104 uma excelente base de desenvolvimento para aplicações industriais, pois une robustez com flexibilidade. Assim, pode-se utilizar o conjunto descrito como uma plataforma de desenvolvimento de diferentes funções de proteção integradas, emulando o funcionamento de um relé de proteção multifuncional, sem a dependência de conhecimentos profundos da arquitetura do *hardware*.

3 SISTEMAS OPERACIONAIS DE TEMPO REAL

Um SO é um *software* (programa) ou conjunto de *softwares* cuja função é gerenciar os recursos de *hardware* disponíveis. O SO pode ser visto como um intermediário entre outros *softwares* e os componentes físicos (*hardware*), além de servir de interface entre a máquina e o usuário final, permitindo assim seu controle (Silberschatz et al., 2004).

SOs podem estar presentes em sistemas embarcados, que são largamente utilizados para controle e monitoramento de aplicações de risco, como aviões, carros, plantas industriais, relés de proteção de SEPs, etc. Tais aplicações são caracterizadas como sistemas de tempo real, em que o tempo de resposta e tratamento a um evento deve ser pré-definido e obedecido, e caso não seja, pode resultar em severas consequências, como falhas. Assim, tais sistemas dependem de um monitoramento/controle correto e feito em tempo pré-definido (Laplante, 2004). Sistemas embarcados podem tirar vantagem dos recursos oferecidos por um SO para o gerenciamento de um sistema de tempo real.

Um SO de tempo real deve atender os programas em execução (processos) de forma a atingir os requisitos de tempo necessários para a aplicação a qual serve. A diferença entre um SO de tempo real e um de propósito geral está basicamente no seu escalonador de processos. Este é responsável por gerenciar o tempo de uso do processador para cada processo, alternando entre estes. O escalonador de um SO de tempo real precisa ser preemptivo, ou seja, os processos têm prioridades uns sobre os outros. Isso permite que um processo de baixa prioridade seja interrompido por um com alta prioridade, alcançando assim os tempos desejados.

3.1 *Real-Time Application Interface* (RTAI)

O Linux é um sistema operacional de código aberto, sob licença *General Public License* (GPL), sendo utilizado em diversos campos, de computadores pessoais a supercomputadores. A grande flexibilidade provida pelo código aberto e a existência de documentação abundante, faz desta uma alternativa atraente para a implementação de novas aplicações.

No entanto, o Linux foi construído como um SO multiusuário de propósito geral e essas características geralmente entram em conflito com as características de um SO de tempo real. Um SO de propósito geral tenta maximizar o desempenho médio do sistema ao custo de um descontrole sobre o tempo de execução de cada processo. Por outro lado, um SO de tempo real tenta colocar um limite superior sobre o tempo de execução ao custo de diminuir o desempenho médio. Existem diferenças estruturais que não permitem ao Linux operar como um SO de tempo real, dentre elas: chamadas de sistemas de kernel não preemptivas, paginação, algoritmo do escalonador de processos, reordenação de requisições e execução em lotes.

Como alternativa aos SOs de tempo real proprietários e aproveitando as vantagens do uso do Linux, surge o RTAI. O RTAI é uma extensão que faz o Linux operar em tempo real, permitindo assim o desenvolvimento de aplicações com restrições de tempo. Ele funciona baseado no modelo de abstração de interrupções, em que modificações fazem com que o Linux seja executado como uma tarefa de baixa prioridade de um pequeno kernel de tempo real. As funções de tempo real são tratadas como de alta prioridade neste pequeno kernel. O restante das aplicações, como gráficos, geração de relatórios, etc, são tratadas pelo Linux. A arquitetura do RTAI pode ser observada na Figura 1.

Como pode ser observado, a arquitetura utilizada pelo RTAI possui uma camada intermediária entre o *hardware* e o kernel do SO Linux, o kernel de tempo real. Este kernel tem como característica a utilização de um escalonador de tarefas preemptivo. Este tem controle sobre os tempos de execu-

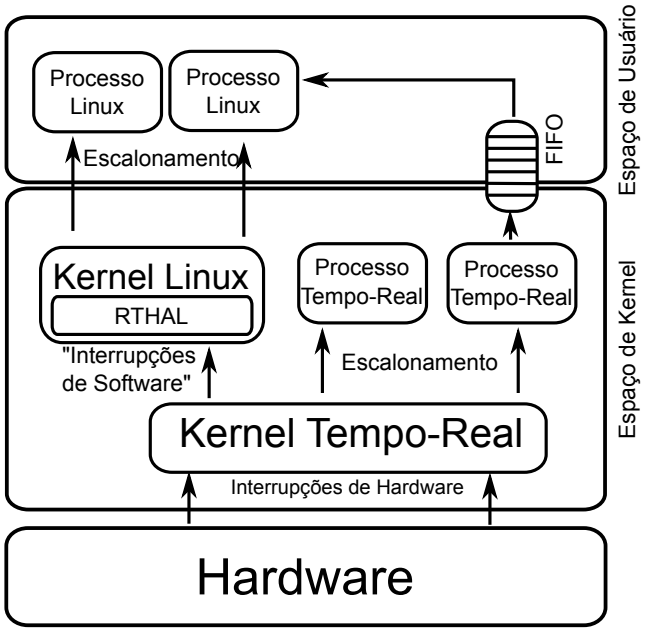

Figura 1: Arquitetura do RTAI

ção de cada processo e consegue aplicar prioridades entre os mesmos. Deste modo, as interrupções do *hardware* são gerenciadas e o tempo de processamento das tarefas de tempo real é dividido. Além disso, abaixo do kernel do Linux é inserida uma camada de abstração de *hardware*, *Real Time Hardware Abstraction Layer* (RTHAL). A RTHAL é controlada por interrupções de *software* provenientes do kernel de tempo real, fazendo com que os processos do Linux sejam gerenciados com baixa prioridade. Para compartilhar informações entre os processos de tempo real e os processos do Linux, pode-se utilizar uma estrutura de dados como a *First In First Out* (FIFO), possibilitando a comunicação entre códigos em execução com diferentes níveis de privilégio.

4 COMEDI

O Comedi (Schleef et al., 2005) é o conjunto padrão de *drivers* e bibliotecas para aquisição de dados do Linux. Iniciado em 1996 por David Schleef, o Comedi busca suportar diversos fabricantes e modelos de placas através de uma interface comum, a *Application Programming Interface* (API), criando assim um compromisso entre modularidade e complexidade.

O Comedi é separado em duas partes: o Comedi em si, um pacote de *drivers* que são carregados em espaço de kernel; o *comedilib*, que atua como interface entre usuários e os *drivers*. Programas utilizando o Comedi podem ser escritos em linguagem C ou C++.

O Comedi está organizado em canais, sub-dispositivos e dispositivos. Um canal é o nível mais baixo de medição e controle. Existem diversos tipos de canais que possibilitam: leitura e escrita de sinais analógicos e digitais, contadores de geradores de pulso e frequência e leitura de *encoders*.

Os canais de um mesmo tipo são agrupados em um conjunto comum, chamado sub-dispositivo. Os sub-dispositivos são agrupados em um dispositivo. A Figura 2 traz uma ilustração desta organização.

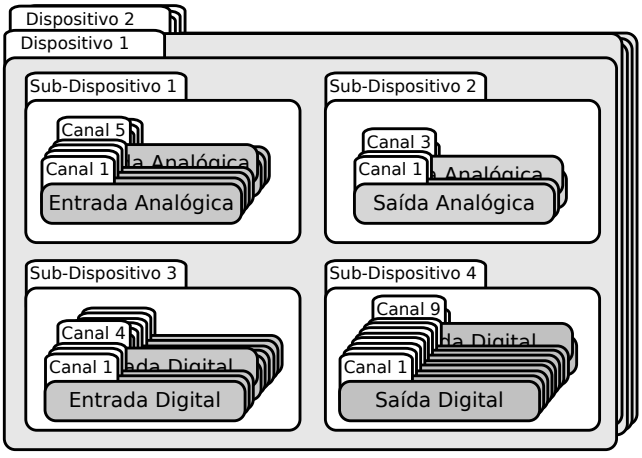

Figura 2: Organização do Comedi

Através de um conjunto de funções disponíveis no *comedilib* é possível acessar os vários dispositivos de uma placa de aquisição de dados.

5 COMPONENTES DO SISTEMA PROPOSTO

Nesta seção serão apresentados os equipamentos e *softwares* que compõem o sistema integrado de desenvolvimento e execução de algoritmos de proteção. Todo o desenvolvimento do *hardware* foi concentrado na plataforma PC104, batizada de *Cubo PC104*. O diagrama de blocos da Figura 3 mostra uma visão geral das principais partes do *hardware* e suas inter-relações. Esse arranjo foi inspirado na arquitetura encontrada em relés digitais comerciais, que é mostrada de forma sintética na Figura 4 (Coury et al., 2007).

5.1 O Cubo PC104

Para avaliar a viabilidade do emprego da plataforma PC/104 nos testes de novos algoritmos de proteção em tempo real, escolheu-se o seguinte conjunto de placas que conta com uma placa de processamento de dados, uma placa de aquisição de dados e uma fonte de energia. A placa de processamento de dados escolhida tem como características:

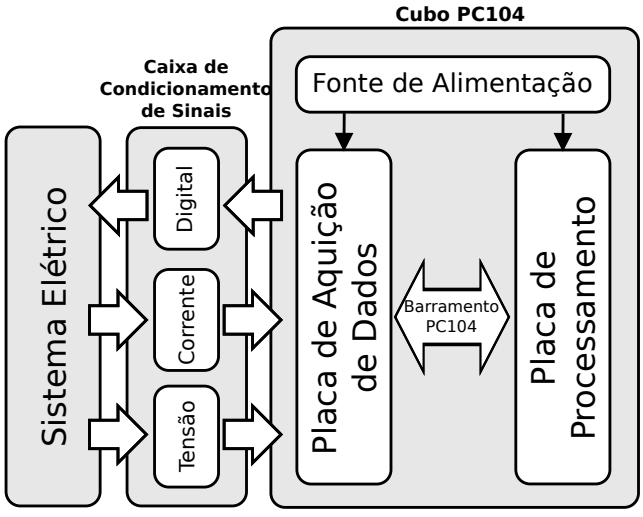

Figura 3: Visão do Conjunto

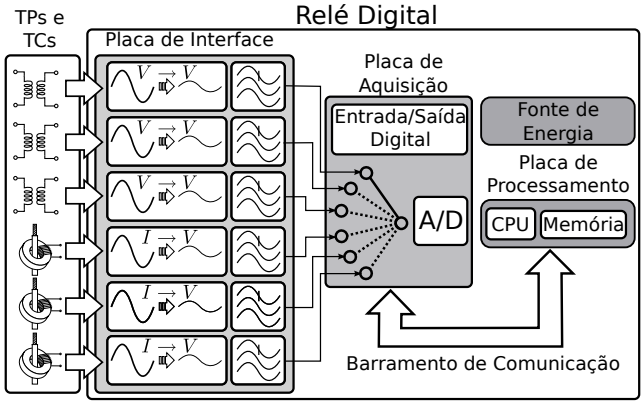

Figura 4: Principais Subsistemas de um Relé Digital.

- Processador AMD Geode LX800;
- 512MB de Memóra RAM DDR333;
- 8GB de Armazenamento em Cartão *Compact Flash* (CF);
- 1 Placa de rede *Ethernet* 10/100 Mb/s;
- 4 Portas USB 2.0;
- 2 Portas Seriais (RS232);
- 1 Porta Paralela.

A escolha da placa de aquisição de dados baseou-se no número de canais de entrada analógica, na taxa de amostragem máxima e a na disponibilidade do *driver* desta placa no Comedi. A placa escolhida tem como principais características:

- Resolução de 16 *bits*;
- Taxa de amostragem máxima de 100 kS/s;
- 16 canais de entrada analógica em modo comum ou 8 em modo diferencial;
- 8 canais de entrada/saída digital;
- *Buffer* de 512 amostras.

A Figura 5 mostra uma foto do *Cubo PC104* com suas dimensões físicas.

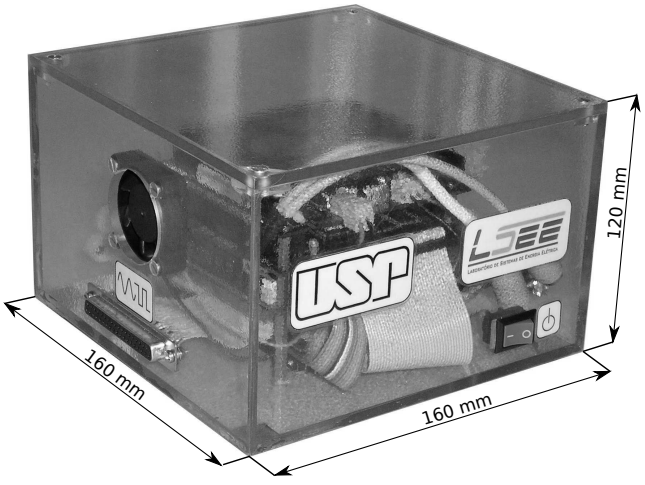

Figura 5: Cubo PC104

5.2 Caixa de Condicionamento de Sinais

A placa de aquisição de dados necessita que os sinais elétricos de entrada, neste caso valores instantâneos de tensões e de correntes, sejam fornecidos em níveis compatíveis aos valores para os quais ela foi projetada. Para tanto, foi desenvolvida uma Placa de Circuito Impresso (PCI) para o condicionamento das correntes de entrada e outra para as tensões de entrada.

A PCI responsável pelo condicionamento de tensão foi projetada para receber sinais de tensão de até $\pm 380 V_{pico}$ e fornecer em sua saída uma tensão de $\pm 10 V_{pico}$, mais apropriada para a placa de aquisição de dados. Um divisor resistivo foi empregado para rebaixar a tensão do sinal de entrada. Um amplificador isolado de barreira ótica foi escolhido para realizar a isolação elétrica entre a plataforma PC104 e o sistema elétrico. Após o amplificador isolado, um amplificador operacional realiza o ajuste de ganho e *offset*. Por fim, o sinal é submetido a um filtro ativo *Anti-Aliasing* do tipo *Butterworth* de segunda ordem com frequência de corte de $2,0 kHz$, conforme mostra a Figura 6.

O condicionamento do sinal de corrente é realizado por uma PCI que converte a corrente de entrada em tensão na sua saída, visto que as placas de aquisição de dados operam, em sua maioria, apenas com tensão. Esta PCI foi projetada para receber sinais de corrente de até $\pm 30 A_{pico}$ e fornecer em sua saída uma tensão de $\pm 10 V_{pico}$. Um sensor de efeito *Hall* foi

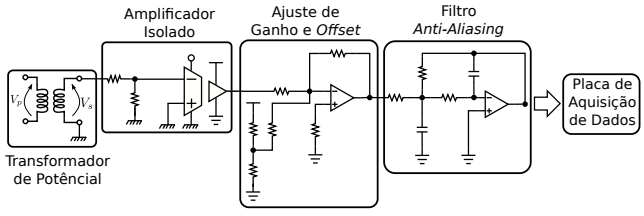

Figura 6: Circuito de Condicionamento de Tensão

utilizado para realizar a conversão do sinal de corrente em tensão, ao mesmo tempo que isola eletricamente o circuito de condicionamento do sistema elétrico. Assim como no circuito de condicionamento de tensão, um amplificador operacional realiza o ajuste de ganho e *offset* e um filtro ativo é utilizado de modo a evitar o efeito *Aliasing*, conforme mostra a Figura 7.

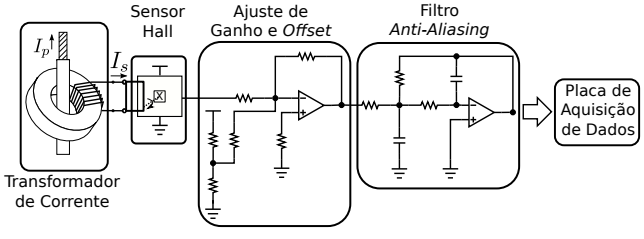

Figura 7: Circuito de Condicionamento de Corrente

5.3 Conjunto de *softwares* utilizados

Nesta subseção será descrito o conjunto de *softwares* escolhido para o desenvolvimento da aplicação proposta. O SO escolhido é o Linux, através da distribuição Debian GNU/Linux ou simplesmente Debian, versão 5.0 (Lenny). O Debian GNU/Linux é uma criação do projeto Debian, uma associação de indivíduos que tem por causa comum criar um SO livre, coerente e completo, prezando principalmente pela estabilidade (Ferrari, 2007). Estes fatos fizeram dele a escolha para o desenvolvimento deste projeto.

Como mencionado anteriormente, o RTAI é um *patch* (modificações no código-fonte) aplicado sobre o *Kernel* do Linux para que este possa suportar aplicações de tempo-real.

Por fim, será também utilizado o Comedi, cujas características principais foram descritas anteriormente.

As versões dos *softwares* citados são exibidas a seguir:

- Debian GNU/Linux 5.0 (Lenny) com kernel 2.6.28.7;
- RTAI 3.7;
- Comedi 0.7.76.

6 AVALIAÇÃO DE DESEMPENHO DO *CUBO PC104*

Nesta seção serão apresentados os procedimentos realizados para avaliar o desempenho do *Cubo PC104* em testes embarcados em tempo real. As análises foram conduzidas por meio da implementação dos algoritmos das funções de proteção diferencial, sobrecorrente de fase com atuação instantânea e sobrecorrente de neutro com atuação temporizada.

Foram observados a precisão da taxa de amostragem e o tempo de processamento das etapas de aquisição de dados, medição e proteção. Os testes foram conduzidos considerando duas taxas de amostragem: 8 e 16 amostras por ciclo de $60Hz$. Simulações de transitórios eletromagnéticos em um SEP modelado no *Alternative Transients Program* (ATP) (ATP, 1987) forneceram dados sobre a operação de um Gerador Síncrono (GS) submetido a faltas internas e faltas externas. Os resultados da atuação das funções de proteção foram comparados aos resultados obtidos de um relé comercial submetido aos mesmos testes.

6.1 O Sistema Elétrico Analisado

A Figura 8 mostra o diagrama unifilar do sistema elétrico utilizado nas simulações para fornecer dados para o teste dos algoritmos de proteção e avaliação do desempenho da plataforma PC104.

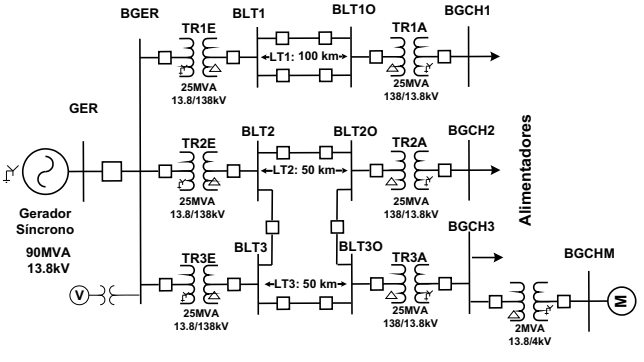

Figura 8: Representação do SEP analisado.

O sistema elétrico é composto por um GS de $13,8kV$ e potência aparente de $90MVA$, transformadores elevadores com relações de $13,8/138kV$ e potência aparente de $25MVA$, linhas de transmissão com extensões variando entre 80 e $150km$, transformadores abaixadores similares aos elevadores e cargas caracterizadas por fator de potência de $0,92$ indutivo e potência aparente variando entre 5 e $25MVA$. Este sistema foi implementado no Laboratório de Sistemas de Energia Elétrica (LSEE) e empregado com

sucesso em outros trabalhos (Barbosa et al., 2007; Barbosa et al., 2008).

6.2 Funções de Proteção Teste

Para realizar os testes de desempenho do *Cubo PC104* foram implementadas algumas funções de proteção, visando a proteção do GS do sistema elétrico discutido na seção 6.1. Para tanto foram implementadas as funções de proteção diferencial do estator (87G) e de sobrecorrente de neutro (51GN) (C37-102, 1996; Kindermann, 2006). Adicionalmente incluiu-se a proteção contra sobrecorrente instantânea de fase (50) (Std-242, 2001). Os ajustes utilizados nas funções de proteção estão disponíveis no Anexo Apêndice A. A Figura 9 mostra as proteções implementadas e a lógica de atuação utilizada. As proteções diferencial e de sobrecorrente de neutro ativam o sinal de disparo (*trip*) e a função de sobrecorrente instantânea de fase ativa o sinal de alarme.

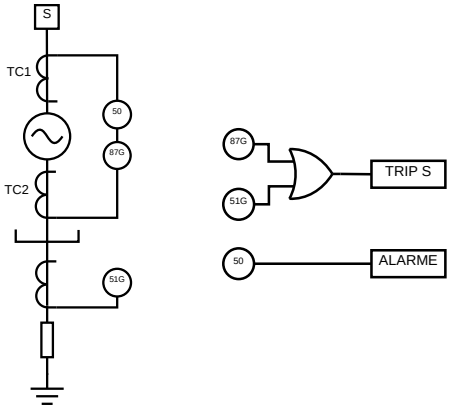

Figura 9: Funções de Proteção e Lógica de Atuação.

Foram utilizadas funções disponíveis na API do RTAI para tornar os algoritmos de proteção um processo de tempo real (*rt_task_init_schmod*, *rt_make_hard_real_time*). Isso garante a máxima prioridade deste sobre todos os processos do SO. O intervalo de amostragem foi controlado por uma função da API do RTAI dedicada à geração de interrupções periódicas (*rt_task_make_periodic*). As aquisições de dados foram realizadas através de funções do *comedilib* que acessam a placa de aquisição de dados, realizam a leitura do canal solicitado e retornam o valor medido devidamente convertido para a grandeza de medição (*comedi_data_read_delayed*, *comedi_to_phys*).

A Figura 10 mostra o fluxo de execução dos algoritmos implementados. O bloco de aquisição refere-se à leitura dos dados da placa de aquisição de dados de todos os canais utilizados. A partir dos dados provenientes da aquisição são calculadas as grandezas necessárias aos algoritmos de proteção, tais como os fasores das corrente para a proteção diferencial e os valores eficazes para as funções de sobrecorrente. Em seguida as funções de proteção são executadas. Por fim, aguarda-se a interrupção de tempo para iniciar uma nova execução (*rt_task_wait_period*).

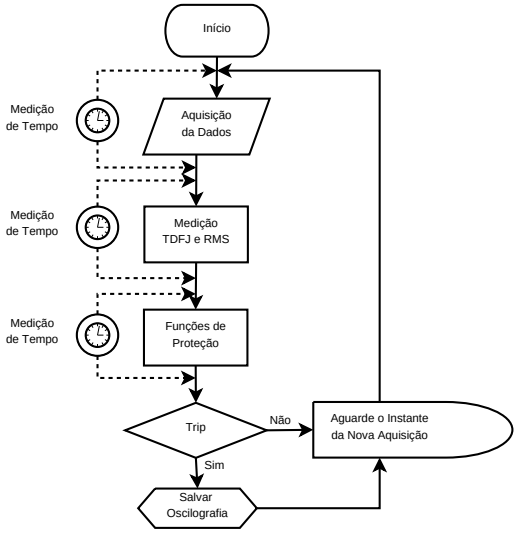

Figura 10: Fluxo de Execução dos Algoritmos de Proteção.

Um *buffer* circular foi implementado para armazenar um conjunto de medidas, pois para calcular os valores eficazes e os fasores é necessário uma quantidade de amostras que represente pelo menos um ciclo completo dos sinais de corrente. A Figura 11 ilustra o princípio de funcionamento do *buffer* circular. Por convenção a amostra mais nova é armazenada na posição 1. Quando uma nova amostra é obtida, a amostra que ocupava a posição 1 ocupará a posição 2, e a amostra que ocupava a última posição é descartada.

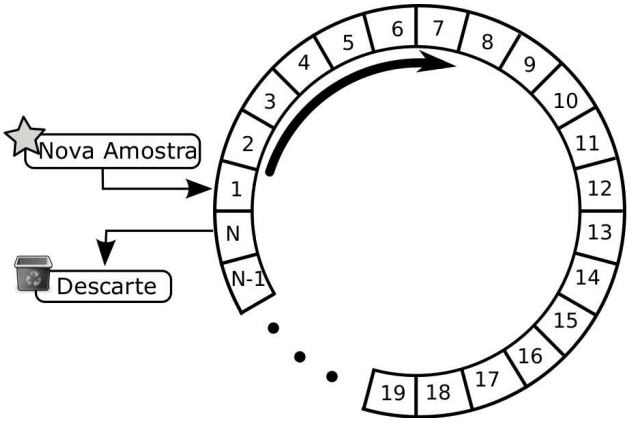

Figura 11: Buffer Circular.

Foram necessários sete *buffers* circulares para as funções de proteção utilizadas: três para as correntes próximas aos terminais do gerador, três para as correntes próximas ao neutro

e um para a corrente de neutro do gerador. Essas funções de proteção foram programadas em linguagem C++, sendo os respectivos algoritmos apresentados a seguir. Além disso, foram implementados os algoritmos para cálculo do valor eficaz e dos fasores das tensões e das correntes, também apresentados em seguida.

O Algoritmo 1 mostra como foi implementado o cálculo dos valores eficazes para as funções de sobrecorrente, no qual $i(n)$ é a corrente instantânea de uma das fases ou do neutro, na n-ésima posição do *buffer* circular e N é o número de amostras por ciclo. O cálculo do fasor para a função diferencial está disponível no Algoritmo 2, no qual optou-se pela Transformada Discreta de Fourier Janelada (TDFJ) de ciclo completo com deslocamento de uma amostra.

Algoritmo 1 Cálculo do Valor Eficaz

$I \leftarrow 0$
$n \leftarrow 1$
for $n \leq N$ **do**
  $I \leftarrow I + i(n)^2$
  $n \leftarrow n + 1$
end for
return $\sqrt{\frac{I}{N}}$

Algoritmo 2 Cálculo do Fasor

$\vec{I} \leftarrow 0 + \jmath 0$
$n \leftarrow 1$
for $n \leq N$ **do**
  $\vec{I} \leftarrow \vec{I} + i(n)e^{\frac{2\pi}{N}\jmath n}$
  $n \leftarrow n + 1$
end for
return $\frac{\sqrt{2}}{N}\vec{I}$

As proteções de sobrecorrente verificam se o valor eficaz da corrente monitorada (I) é inferior ao valor máximo permitido (I_{pickup}). O Algoritmo 3 mostra como essa proteção foi programada. A proteção diferencial realiza a comparação dos fasores das correntes de entrada e de saída de cada fase. O pseudo-código da proteção diferencial está disponível no Algoritmo 4, no qual I_1 e I_2 indicam as correntes de entrada e de saída do elemento protegido, I_o é a corrente de operação, I_r é a corrente de restrição.

Algoritmo 3 Proteção de Sobrecorrente

if $I(k) \geq I_{pickup}$ **then**
  return $true$
else
  return $false$
end if

6.3 Teste do Conjunto

Utilizando o *Cubo PC104*, os algoritmos de proteção discutidos na seção 6.2 foram embarcados e posteriormente testados com auxílio de uma caixa de geração de sinais elétricos, que é um sistema de alta potência que pode gerar sinais

Algoritmo 4 Proteção Diferencial

$I_o \leftarrow \left|\vec{I_1} - \vec{I_2}\right|$
$I_r \leftarrow \left|\vec{I_1} + \vec{I_2}\right|$
if (I_r, I_o) acima da curva de operação **then**
  return $true$
else
  return $false$
end if

elétricos a partir de arquivos de simulações. Essa caixa pode ser utilizada para testar dos antigos relés eletromecânicos aos modernos relés equipados com microprocessadores.

Alimentando a caixa de geração de sinais elétricos com as oscilografias obtidas das simulações de faltas no sistema elétrico da Figura 8, pretendeu-se avaliar a confiabilidade e desempenho dos algoritmos em tempo real de execução, mostrando assim seu potencial de aplicação em campo. O diagrama apresentado na Figura 12 descreve a sequência dos eventos necessários para a realização dos testes das funções de proteção e da avaliação global do sistema proposto.

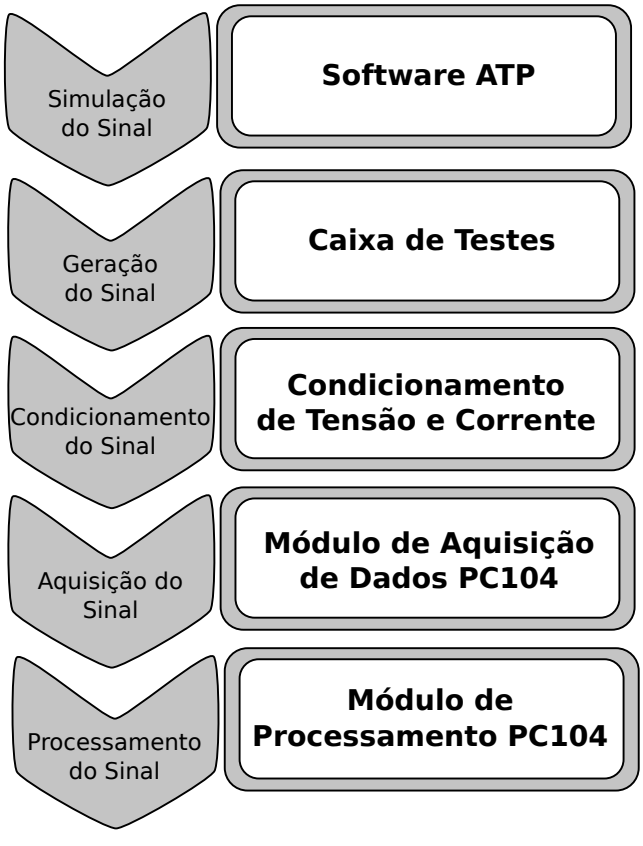

Figura 12: Estrutura geral de testes na plataforma PC104

6.4 Teste do Relé Comercial

A título de comparação de desempenho das funções de proteção implementadas, um relé comercial foi ensaiado. No relé em questão foram habilitadas as funções de proteção discutidas na seção 6.2 e os parâmetros de ajuste foram os mesmos utilizados no teste da plataforma.

A Figura 13 mostra a estrutura geral dos testes usando o relé comercial.

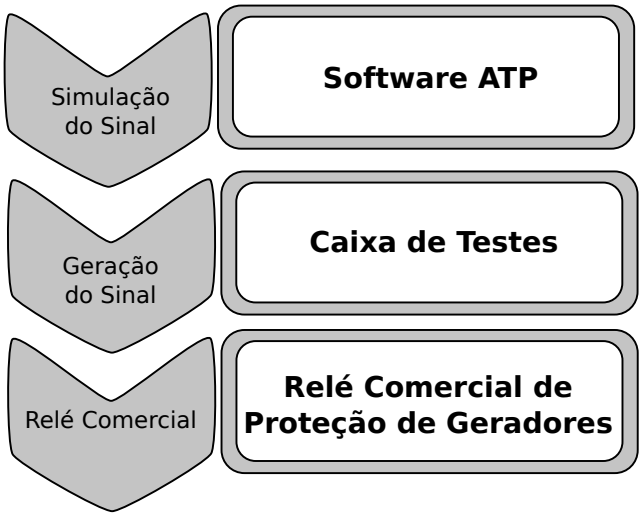

Figura 13: Estrutura geral de testes no relé comercial

7 RESULTADOS DA IMPLEMENTAÇÃO DO *CUBO PC104*

Nos testes realizados no *Cubo PC104* foram avaliados a correta atuação das funções de proteção, o tempo de execução de cada etapa do algoritmo e a precisão da taxa de amostragem.

Neste trabalho duas taxas de amostragem, 8 e 16 amostras por ciclo de $60Hz$, foram utilizadas nos testes realizados.

7.1 Tempo de Processamento

Um dos critérios de avaliação do desempenho do sistema integrado proposto neste trabalho é a capacidade de processamento, que compreende o tempo necessário para realizar a aquisição dos sinais de entrada, pré-processa-los e por fim executar as funções de proteção. Ressalta-se que o tempo disponível para a execução de todas essas tarefas é o intervalo de tempo entre duas amostragens consecutivas. Caso esse critério não seja atendido, ocorrerá a perda de informação com prejuízo à confiabilidade da função de proteção.

A Figura 14 mostra execução das etapas do algoritmo ao longo do tempo, na qual $T(k)$ denota o tempo em que ocorrem as interrupções periódicas, $T_a(k)$, $T_m(k)$ e $T_p(k)$ indicam o fim das etapas de aquisição, medição e proteção, respectivamente.

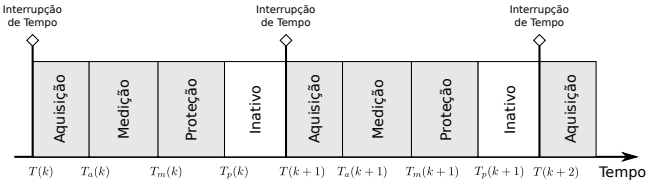

Figura 14: Tempo de Execução dos Algoritmos de Proteção.

A avaliação do tempo de execução busca evidenciar a utilização do tempo de processamento disponível de cada bloco do algoritmo em execução. Para calcular o tempo de execução da etapa de aquisição (T_{acq}), medição (T_{mea}) e proteção (T_{pro}) foram utilizadas as Equações 1, 2 e 3, nessa mesma ordem, nas quais k indica a k-ésima execução do algoritmo. Para o cálculo do tempo de inatividade (T_{idl}) utilizou-se a Equação 4. É importante salientar que T_{idl} deve ser maior que zero, pois é neste intervalo de tempo que os demais processos de baixa prioridade do SO são executados. Além disso, o tempo de inatividade nulo indica que a placa de processamento não consegue executar a tarefa solicitada em tempo real. Ressalta-se que o tempo gasto pelas instruções que realizam as medições de tempo foi desconsiderado.

$$T_{acq}(k) = T_a(k) - T(k) \quad (1)$$

$$T_{mea}(k) = T_m(k) - T_a(k) \quad (2)$$

$$T_{pro}(k) = T_p(k) - T_m(k) \quad (3)$$

$$T_{idl}(k) = T(k+1) - T_p(k) \quad (4)$$

Para realizar essa avaliação, foram medidos os tempos de execução das etapas do algoritmo durante o intervalo de um minuto. Neste período, a plataforma PC104 foi alimentada com oscilografias de situações de operação do sistema elétrico em regime permanente e em condição de falta. Este procedimento buscou identificar alguma mudança significativa nos tempos de execução das etapas em diversas situações de operação.

As Tabelas 1, 2 e 3 mostram, respectivamente, os tempos de execução das etapas de aquisição, medição e proteção, para o intervalo observado. Constata-se que os tempos de execução se mantiveram praticamente constantes e com baixa variação.

Os gráficos das Figuras 15(a) e 15(b) buscam destacar o tempo gasto na execução de cada etapa do algoritmo frente ao tempo total disponível para o processamento. Observa-se que o tempo de execução das funções de proteção utilizadas foi bem inferior ao tempo decorrido entre amostras consecutivas. Este desempenho positivo possibilita avaliar funções de proteções mais avançadas, que exijam a execução de cálculos mais complexos.

Algo a ser notado é que os períodos médios de tempo gastos na aquisição de dados e na execução da proteção foram praticamente iguais, para as duas taxas de amostragens utilizadas. No entanto, o percentual de processamento destacado para essas atividades dobrou, como se observa nas Figuras 15(a) e 15(b). Isso ocorre porque o tempo total disponível quando se utiliza a taxa de 16 amostras por ciclo é metade do tempo disponível para a taxa de 8 amostras por ciclo.

Outra constatação importante é que o tempo dedicado à medição quadruplicou percentualmente, quando se consideram 16 amostras por ciclo. Como a taxa de amostragem duplicou, o tempo disponível ao processamento entre duas amostras consecutivas foi reduzido à metade. Além disso, dada a nova taxa de amostragem, o número de amostras que representam um ciclo de forma de onda também aumentou. Deste modo, o número de operações aritméticas para o cálculo dos fasores e dos valores eficazes dobrou. Com referência aos Algoritmos 1 e 2, a variável N passa de 8 para 16. Portanto, neste caso, o aumento do número de operações aritméticas aliada à diminuição do tempo disponível para tais cálculos elevou a participação percentual da etapa de medição no total do tempo suficiente para o processamento de todas as etapas. Contudo, ainda há tempo disponível para a execução das instruções não prioritárias, conforme mostrado na Figura 15(b).

Tabela 1: Tempo Execução da Aquisição

Amostras por Ciclo	8	16
Médio (μs)	159,892	159,912
Variância (μs^2)	0,222	0,277
Máximo (μs)	177,625	174,458
Mínimo (μs)	158,514	158,359

Tabela 2: Tempo Execução da Medição

Amostras por Ciclo	8	16
Médio (μs)	94,252	173,485
Variância (μs^2)	0,0696	0,174
Máximo (μs)	103,125	182,812
Mínimo (μs)	93,655	172,256

Tabela 3: Tempo Execução da Proteção

Amostras por Ciclo	8	16
Médio (μs)	25,586	25,476
Variância (μs^2)	0,065	0,069
Máximo (μs)	35,061	34,728
Mínimo (μs)	25,032	24,956

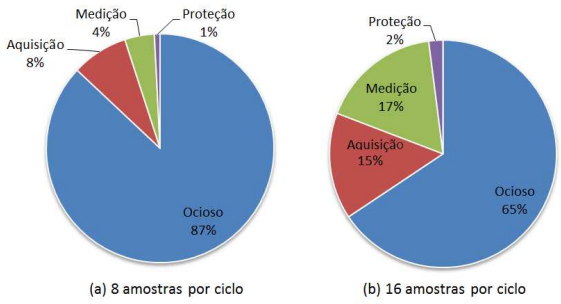

Figura 15: Tempo de Execução.

7.2 Precisão da Taxa de Amostragem

Outro critério para avaliar o desempenho do sistema proposto foi a precisão da taxa de amostragem dos sinais de entrada. Uma vez que a taxa de amostragem é controlada por *software* é importante verificar sua precisão e desvio máximo. Esta preocupação justifica-se pelo fato que, se a taxa de amostragem não for constante, devido à latência, os sinais de entrada não serão amostrados coerentemente. Isso leva a erros de medição e por consequência à atuação errônea das funções de proteção, como exemplo, o espalhamento espectral na TF (Baghzouz et al., 1998). Para a medição do intervalo de amostragem, utilizou-se a Fórmula 5. Para tal, a taxa de amostragem foi medida em um intervalo de um minuto de execução do algoritmo de proteção. Durante esse intervalo, a plataforma PC104 foi alimentada com oscilografias de situações de operação do sistema elétrico em regime permanente e em condições de falta. Este procedimento visou evidenciar eventuais variações na taxa de amostragem devido à mudança no regime de operação do sistema elétrico.

$$T_{smp}(k) = T(k+1) - T(k) \tag{5}$$

A Tabela 4 mostra os resultados obtidos para as duas taxas de amostragem investigadas, 8 e 16 amostras por ciclo de $60Hz$, ou seja, $2083,33\mu s$ e $1041,66\mu s$ respectivamente. Observa-se que o erro médio foi extremamente baixo, assim como, o erro máximo. A partir do histograma da Figura 16, que mostra a dispersão do erro da taxa de amostragem, pode-se observar que 90% das amostras têm erro inferior a $1,25\mu s$ $(0,06\%)$ para a taxa de amostragem de 8 amostras por ciclo. Com auxílio do histograma da Figura 17, que mostra a dispersão do erro da taxa de amostragem para o caso que se usou a taxa de 16 amostras por ciclo, constata-se que 90% das amostras têm erro menor que $1,23\mu s$ $(0,11\%)$.

Tabela 4: Precisão da Taxa de Amostragem

Amostras por Ciclo	8	16
Erro Absoluto Médio (μs)	0,607	0,543
Erro Relativo Médio ($\%$)	0,058	0,052
Variância (μs^2)	0,295	0,266
Erro Absoluto Máximo (μs)	11,882	9,817
Erro Relativo Máximo ($\%$)	1,140	0,942

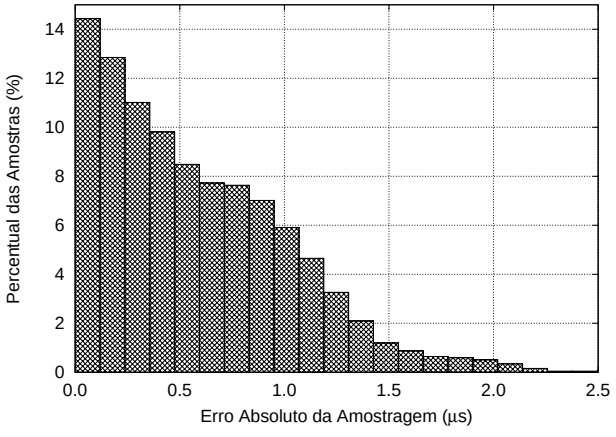

Figura 16: Dispersão do Erro de Amostragem (8 Amostras por Ciclo).

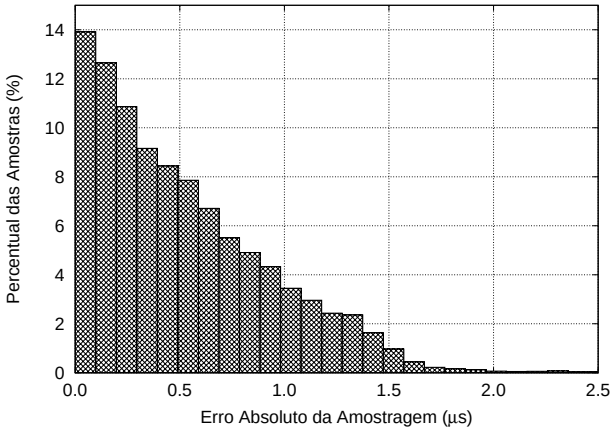

Figura 17: Dispersão do Erro de Amostragem (16 Amostras por Ciclo).

7.3 Desempenho das Funções de Proteção

Os gráficos das Figuras 18 e 19 mostram, respectivamente, a resposta das funções de proteção implementadas no sistema proposto e do relé comercial. O distúrbio simulado foi uma falta entre a fase C do gerador e a terra dentro da área de proteção da função de proteção diferencial, conforme mostra a Figura 9. Os gráficos das Figuras 20 e 21 mostram o desempenho das funções de proteção dos dois sistemas (sistema proposto e o relé comercial) para uma situação de falta entre a fase A e a fase B do gerador dentro da área de proteção da função de proteção diferencial. O sub-índice 1 indica as correntes dos Transformadores de Corrente (TCs) próximos aos terminais dos geradores e o sub-índice 2 discrimina as correntes dos Transformador de Correntes (TCs) próximos ao neutro do gerador. Por fim, a corrente de neutro é assinalada por IN. A parte inferior dos gráficos representa as saídas lógicas de todas das funções de proteção e da lógica de atuação implementada, sendo que a linha cheia indica o estado ativo. Como o tempo de atuação da função de proteção de sobrecorrente de neutro é muito longo, optou-se por indicar somente o instante da partida desta função nos gráficos analisados. Pela análise comparativa dos gráficos conclui-se que as funções de proteção, embarcadas no PC104, atuaram corretamente.

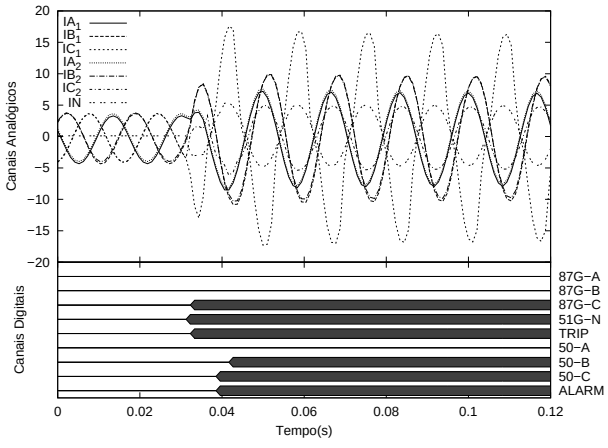

Figura 18: Resposta Cubo PC104 (Falta C-Terra)

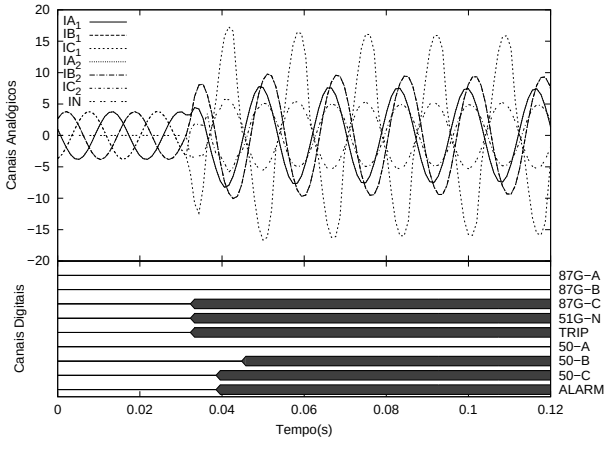

Figura 19: Resposta Relé Comercial (Falta C-Terra)

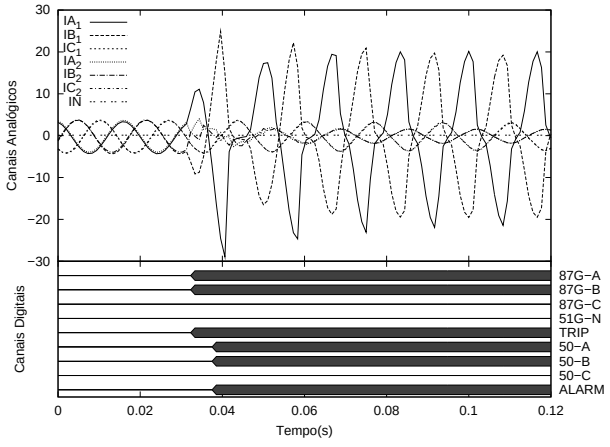

Figura 20: Resposta Cubo PC104 (Falta A-B)

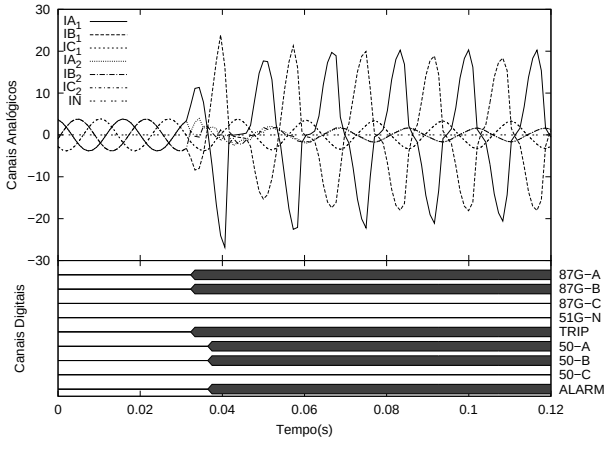

Figura 21: Resposta Relé Comercial (Falta A-B)

Uma segunda análise foi feita no que diz respeito ao tempo de atuação. As Tabelas 5 e 6 mostram os tempos de atuação das funções de proteção embarcadas no PC104 e as funções ativadas no relé comercial para as situações de faltas indicadas anteriormente. Com base nestas, confirma-se o correto funcionamento dos algoritmos implementados. Pequenas diferenças no tempo atuação podem ser observadas, e isso se deve, provavelmente, aos algoritmos específicos de cálculo dos fasores ou ao número de amostras que o relé comercial emprega para confirmar o evento de proteção. Enfatiza-se que essas informações não estavam disponíveis durante a execução dos ensaios em laboratório.

8 CONCLUSÕES

Este artigo mostrou o desenvolvimento e avaliação de desempenho de um sistema integrado de *hardware* e *software* para desenvolver e executar, em tempo real, funções de proteção aplicadas em sistemas elétricos. Uma das principais vantagens do sistema proposto é facilitar a programação das funções de proteção, uma vez que não exige o conhecimento detalhado de sua arquitetura, tampouco de linguagens de programação de baixo nível. Além disso, a execução em tempo real e implementação em *hardware* são fundamentais para avaliar a viabilidade de implementação de novos algoritmos de proteção em condições próximas às encontradas em casos práticos.

Tabela 5: Tempos de Atuação (Falta C-Terra)

Função de Proteção	Tempo de Atuação (ms)	
	PC104	Relé Comercial
87G-A	†	†
87G-B	†	†
87G-C	2,13	2,09
50-A	†	†
50-B	11,44	14,55
50-C	8,30	8,27
51G-A	0,92	2,09

† Não houve atuação.

Tabela 6: Tempos de Atuação (Falta A-B)

Função de Proteção	Tempo de Atuação (ms)	
	PC104	Relé Comercial
87G-A	2,22	2,28
87G-B	2,22	2,28
87G-C	†	†
50-A	7,49	6,18
50-B	7,49	6,18
50-C	†	†
51G-A	†	†

† Não houve atuação.

A avaliação da precisão da taxa de amostragem mostrou que o uso do RTAI é viável em aplicações de tempo crítico como proteção. A placa de processamento foi capaz de executar os algoritmos de proteção com rapidez aceitável, o que mostra a possibilidade do uso desta para executar funções de proteção mais complexas. Conclui-se que o sistema desenvolvido neste trabalho atendeu a todos os requisitos de funcionamento examinados, destacando-se como uma importante ferramenta para auxiliar o desenvolvimento de novas técnicas de proteção, diminuindo o tempo de implementação e possibilitando testes mais completos. Ressalta-se que o emprego de funções de proteção tradicionais foi útil para validar os resultados do sistema proposto. Como continuidade deste trabalho, funções de proteção mais sofisticadas, envolvendo cálculos mais complexos e considerando técnicas de inteligência artificial serão implementadas e testadas na plataforma PC104.

REFERÊNCIAS

ATP (1987). Alternative Transients Program - Rule Book.

Baghzouz, Y., Burch, R., Capasso, A., Cavallini, A., Emanuel, A., Halpin, M., Imece, A., Ludbrook, A., Montanari, G., Olejniczak, K., Ribeiro, P., Rios-Marcuello, S., Tang, L., Thaliam, R. and Verde, P. (1998). Time-varying harmonics. I. Characterizing measured data, *IEEE Transactions on Power Delivery* **13**(3): 938–944.

Barbosa, D., da Silva, M., Bernardes, A. P., Oleskovicz, M. and Coury, D. V. (2007). Lógica Fuzzy aplicada na proteção de transformadores, *VIII Simpósio Brasileiro de Automação Inteligente*, Florianópolis.

Barbosa, D., Monaro, R., Coury, D. and Oleskovicz, M. (2008). Filtragem adaptativa para a estimação da frequência em sistemas elétricos de potência, *SBA Controle & Automação* **19**(2): 226–234.

C37-102 (1996). IEEE Guide for AC Generator Protection, *IEEE Std C37.102-1995* p. i.

Coury, D. V., Oleskovicz, M. and Giovanini, R. (2007). *Proteção digital de sistemas elétricos de potência: dos relés eletromecânicos aos microprocessados inteligentes*, 1 edn, EDUSP, São Carlos.

Dadash Zadeh, M., Sidhu, T. and Klimek, A. (2009). FPAA-based mho distance relay considering cvt transient supervision, *Generation, Transmission Distribution, IET* **3**(7): 616 –627.

Ferrari, F. A. (2007). *Curso Prático de Linux*, Universo dos Livros Ltda., São Paulo.

Kindermann, G. (2006). *Proteção de Sistemas Elétricos de Potência*, Vol. 3, UFSC-EEL-LabPlan, Santa Catarina.

Laplante, P. A. (2004). *Real-Time Systems Design and Analysis*, 3 edn, Wiley-IEEE Press.

McLaren, P., Swift, G., Neufeld, A., Zhang, Z., Dirks, E. and Haywood, R. (1994). "Open" systems relaying, *Power Delivery, IEEE Transactions on* **9**(3): 1316 –1324.

PC/104 Consortium (2009). No Title, Online.
URL: *http://www.pc104.org*

Ruan, J. and Lin, J. (2005). The application of DSP technique in the field of relay protection, *Communications, Circuits and Systems, 2005. Proceedings. 2005 International Conference on*, Vol. 2, pp. 1326–1329.

Schleef, D., Hess, F. and Bruyninckx, H. (2005). *The Control and Measurement Device Interface handbook*.

Sidhu, T. S., Bajpai, M., Burnworth, J., Darlington, A., Kasztenny, B., McLaren, P., Nagpal, M., Saha, M., Sachdev, M., Swanson, M. and Winston, P. (2006). Bibliography of Relay Literature, 2004 IEEE Committee Report, *IEEE Transactions on Power Delivery* **21**(3): 1199–1212.

Sidhu, T. S., Burnworth, J., Darlington, A., Kasztenny, B., Liao, Y., McLaren, P. G., Nagpal, M., Sachdev, M. S., Saha, M. M., Swanson, M. and Winston, P. B. (2008). Bibliography of Relay Literature, 2006 IEEE Committee Report, *IEEE Transactions on Power Delivery* **25**(1): 88–101.

Sidhu, T. S., Burnworth, J., Darlington, A., Kasztenny, B., Liao, Y., McLaren, P. G., Nagpal, M., Sachdev, M. S., Saha, M. M., Swanson, M. and Winston, P. B. (2010). Bibliography of Relay Literature, 2007 IEEE Committee Report, *IEEE Transactions on Power Delivery* **25**(1): 88–101.

Sidhu, T. S., Burnworth, J., Darlington, A., Kasztenny, B., Liao, Y., McLaren, P., Nagpal, M., Sachdev, M., Saha, M., Swanson, M. and Winston, P. (2007). Bibliography of Relay Literature, 2005 IEEE Committee Report, *IEEE Transactions on Power Delivery* **22**(2): 781–794.

Silberschatz, A., Galvin, P. B. and Gagne, G. (2004). *Operating System Concepts*, 7 edn, Wiley.

Std-242 (2001). IEEE Recommended Practice for Protection and Coordination of Industrial and Commercial Power Systems, *IEEE Std 242-2001 [IEEE Buff Book]* pp. i –740.

Stranneby, D. (2001). *Digital signal processing: DSP and applications*, Newnes.

APÊNDICE A

Ajustes das Funções de Proteção

Os ajustes das funções de proteção utilizadas no trabalho são apresentadas neste anexo, por meio das tabelas A.1, A2 e A.3.

Tabela A.1: Proteção Diferencial

Pick-Up	*Break* 1	*Slope 1*	*Break* 2	*Slope 2*
$0,2\ pu$	$1,15\ pu$	15%	$8,0\ pu$	50%

Tabela A.2: Proteção de Sobrecorrente de Fase Instantânea

Pick-Up
$1,15\ pu$

Tabela A.3: Proteção de Sobrecorrente de Neutro

Pick-Up	*Dial*	*Curve Type*
$0,4\ pu$	$4,0$	*IEEE Very Inverse*