Universidade Federal de Campina Grande
Centro de Engenharia Elétrica e Informática
Unidade Acadêmica de Engenharia Elétrica
Curso de Graduação em Engenharia Elétrica

Instrumento Virtual para Computador Pessoal

Cícero Einstein do Nascimento Santos

Campina Grande (PB)
2009

Cícero Einstein do Nascimento Santos

Instrumento Virtual para Computador Pessoal

Projeto apresentado ao Curso de Graduação em Engenharia Elétrica da Universidade Federal de Campina Grande, como requisito para conclusão da disciplina Projeto em Engenharia Elétrica.

Orientador: Prof. Alexandre Cunha Oliveira

Campina Grande (PB)

2009

Cícero Einstein do Nascimento Santos

Instrumento Virtual para Computador Pessoal

Projeto apresentado ao Curso de Graduação em Engenharia Elétrica da Universidade Federal de Campina Grande, como requisito para conclusão da disciplina Projeto em Engenharia Elétrica.

_____

Cícero Einstein do Nascimento Santos

Orientando

_____

Prof. Alexandre Cunha Oliveira, D. Sc.

Orientador

Dedico aos amigos que torceram
por mim e me acompanharam nesta
etapa final.

Agradecimentos

A Deus, pelo dom da sabedoria e pelas bênçãos alcançadas durante esta caminhada.

Ao prof. Alexandre, pela orientação deste trabalho e pelas contribuições na consolidação dos conceitos adquiridos.

Aos meus amigos Isaac e Ciclindo, por tornarem minha estadia em Campina Grande agradável e divertida, pela confiança, pelos conselhos, pela conversa fiada e séria, e principalmente pela amizade.

Aos meus companheiros de curso, Allan, Bybyanna, Diego, Fábio, Marcus e João, por me acompanharem nesta caminhada.

Aos meus companheiros do Iphotobot, pela amizade e pelas reuniões não culturais ou científicas.

Aos meus mais recentes companheiros de apartamento, Gustavo, Rafael (Baiano), Thibério, que me acolheram e acompanharam nesta etapa final, pelas conversas da madruga e pelas partidas de CS.

Aos meus pais e irmãos (aqui incluo Tivan), pelo apoio, pelos sacrifícios e pelo amor a mim dedicado.

E por fim, a Juliana, por todo amor e carinho a mim dedicado, pelos incentivos e por chorar as minhas mágoas e sorrir minhas vitórias comigo.

Procure ser um homem de valor,
ao invés de procurar ser um
homem de sucesso.

Albert Einstein

Resumo

O presente trabalho propõe a utilização de um computador pessoal em conjunto com um sistema de aquisição de dados para projetar um instrumento virtual. Os passos apresentados compreendem a especificação dos dispositivos e equipamentos utilizados, assim como a configuração do sistema operacional de forma a torná-lo determinístico em relação a interrupções temporizadas e periódicas.

Palavras-chave: Instrumento Virtual, Linux, Recompilação de *Kernel*, Determinismo Temporal, Aquisição de Dados.

Sumário

1. Introdução

Uma aplicação que usa múltiplos instrumentos pode facilmente envolver vários processos, cada um dedicado a uma tarefa específica. Além disso, estes processos múltiplos podem operar em paralelo, proporcionando um incremento no desempenho geral. Porém, este incremento pode tornar a tarefa onerosa, vista a necessidade de aquisição de vários instrumentos.

Instrumentos tradicionais como: voltímetros, osciloscópios, monitores de grandezas naturais ou elétricas e controladores, entre outros, podem ser substituídos por um sistema formado basicamente por um conversor Analógico/Digital (A/D) e um conjunto de rotinas desenvolvidas em um computador pessoal.

O objetivo de uma instrumentação virtual é utilizar um computador qualquer para imitar um instrumento real com controles e mostradores dedicados, mas com a versatilidade oferecida por um *software* [1].

Um instrumento virtual pode ser definido como um sistema formado por sensores, *hardware* e *software*, organizados para criar um instrumento flexível e sofisticado de monitoramento e controle [2].

Instrumentos virtuais são largamente utilizados no ensino de medidas e suas teorias [3] [4] e em atividades laboratoriais [4] [5] [6] [7]. Além disto, pode ser observada sua inserção em atividades especializadas, tais como aplicações médicas [8].

Instrumentos virtuais são mais efetivos quando utilizados com um ambiente computacional adequado. A maior desvantagem de se utilizar um computador pessoal (PC) para desenvolver instrumentos virtuais é que ele possui apenas um microprocessador central.

Ao final deste trabalho pretende-se apresentar uma investigação sobre a implementação de uma ferramenta de medição e análise de sinais elétricos, baseada na utilização de uma placa de aquisição de dados e um computador pessoal.

2. Objetivo Geral

- Investigar a implementação de um instrumento virtual baseado em uma placa de aquisição de dados e um computador pessoal.

3. Objetivos Específicos

- Apresentar o sistema de *hardware*;
- Investigar os requisitos do sistema operacional;
- Apresentar teste de um protótipo de instrumento virtual.

4. Sistema de *Hardware*

4.1. Computador Pessoal

Foi utilizado um computador pessoal (PC) com:

- Processador AMD K6-2(3D), 400 MHz;
- 256 MB de memória RAM;
- HD 20GB;
- Dispositivo de leitura óptico (CD-R);
- Dispositivo de leitura flexível (3,5” - 1,44MB).

4.2. Placa de Aquisição de Dados

Foi utilizada uma placa de aquisição de dados (Figura 4.1) com interface para barramento *Industry Standard Architecture* (ISA), e composta de:

- Circuito de Seleção;
- Circuito de Interface Paralela Programável (PPI);
- Circuito de Temporização;
- Circuito de Conversores Analógico/Digital (A/D);
- Circuito de Conversores Digital/Analógico (D/A).

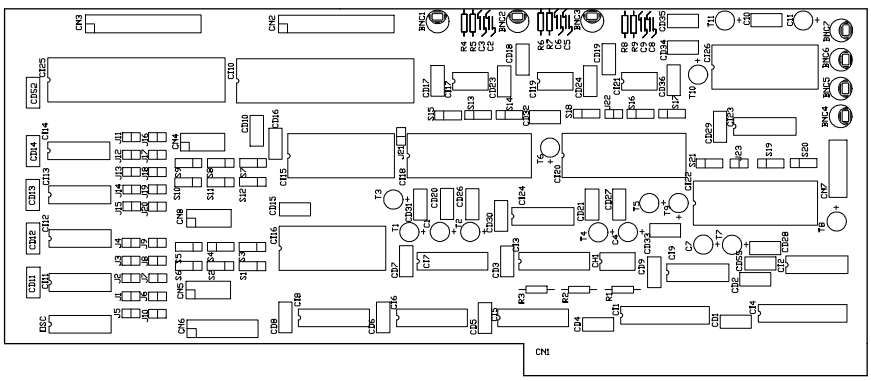

Figura 4. 1 - Placa de Aquisição de Dados.

4.2.1. Circuito de Seleção

O circuito de seleção determina qual dispositivo será utilizado a cada instante (apenas um dispositivo por vez tem acesso ao barramento de dados). Especifica também, quais as operações efetuadas pela PPI, circuitos temporizadores, A/D's e D/A's.

O circuito de seleção também é responsável pela programação do endereço base (EB) da placa, tornando possível sua identificação pelo PC. Tal endereço é definido a partir de chaves tipo *Dual In-line Package Switch* (DIP Switch) identificadas na Figura 4.1 por (CH1).

Todos os endereços apresentados na Tabela 4.1 encontram-se em hexadecimal e foram escolhidos de forma a não haver conflitos com o funcionamento de outros sistemas de *hardware* no PC.

Tabela 4. 1 – Estado das chaves para seleção do endereço base.

Endereço Base (EB)	Chave 1	Chave 2	Chave 3	Chave 4*
0100H	ON	OFF	ON	XX
0200H	ON	ON	OFF	XX
0300H**	ON	OFF	OFF	XX

* A Chave 4 encontra-se desconectada, logo pode assumir qualquer estado.
** Endereço base utilizado no projeto.

4.2.2. Circuito de PPI

A PPI é o meio existente na placa de aquisição para efetuar comunicação digital de forma paralela. A placa comporta duas interfaces desse tipo, implementadas por circuitos integrados (CI) 82C55A do fabricante *Harris Semiconductor*. Cada dispositivo PPI, possui 3 portas configuráveis como entrada, saída ou controle, via *software*. A configuração é feita através da gravação de uma palavra de controle no registrador de controle da PPI. O diagrama funcional da PPI é representado pelo diagrama de blocos da Figura 4.2.

O *Data Bus Buffer* é utilizado para transmissão de dados durante a execução de uma instrução de leitura ou escrita na PPI. Palavras de controle e de estado também são transferidas por este módulo.

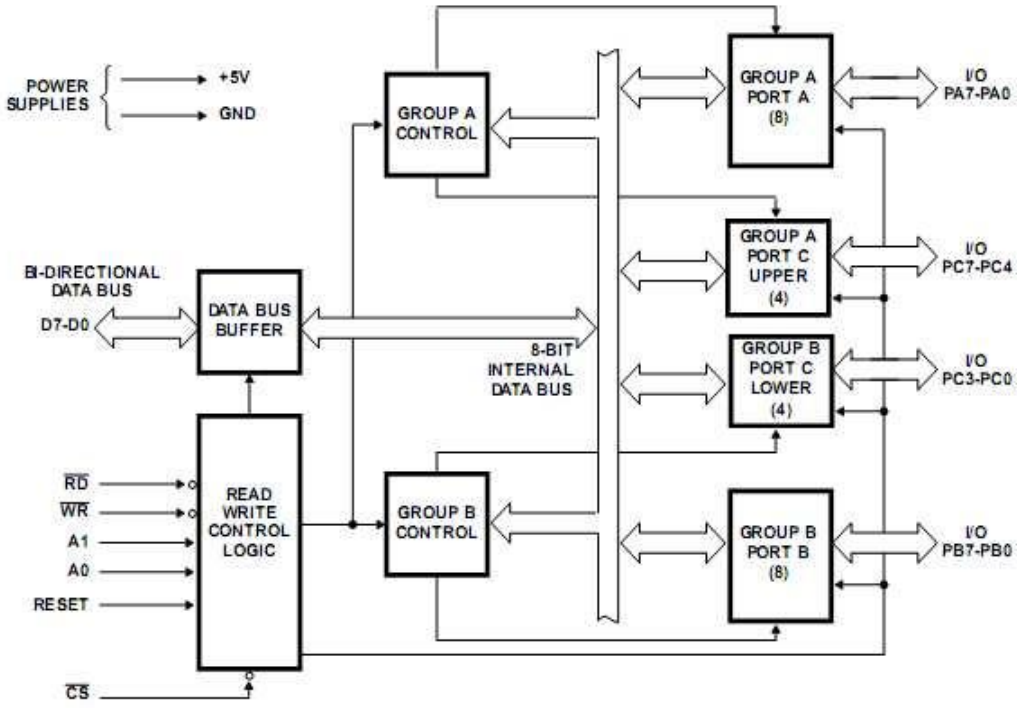

Figura 4. 2 – Diagrama blocos do 82C55A [9].

O bloco *Read/Write Control Logic* administra a transferência interna e externa de dados e palavras de controle ou estado. Este módulo envia as instruções aos blocos de controle *Goup A Control* e *Group B Control*, Seu funcionamento é definido a partir dos pinos CS, RD, WR, A0 e A1, da seguinte forma:

- CS – Habilita a comunicação entre a PPI e a Unidade Central de Processamento (CPU);
- RD – Habilita a leitura de dados a partir da PPI;
- WR – Habilita a escrita de dados na PPI;
- A0 e A1 – Seleciona entre as portas e o registrador de controle;
- Reset – Inicializa o registrador de controle para 9Bh e as portas A, B e C são configuradas como entrada.

Os blocos *Goup A Control* e *Group B Control* são responsáveis pelo controle de acesso às portas A, B e C. Estes blocos aceitam comandos do *Read/Write Control Logic*, recebem palavras de controle do barramento de dados e enviam comandos para as portas associadas. O Grupo A é composto pelas portas A e C (quatro bits mais significativos) e o Grupo B pelas portas B e C (quatro bits menos significativos).

O controle das portas pode ser feito através da escrita de uma palavra de controle de 8 bits, no formato D_7-D_6-D_5-D_4-D_3-D_2-D_1-D_0, onde:

- D_7 - Deve possuir sempre o nível lógico “1”;
- D_6 e D_5 – Definem o modo de operação do Grupo A;
- D_4 e D_3 – Definem as portas do Grupo A como entrada ou saída;
- D_2 – Define o modo de operação do Grupo B;
- D_1 e D_0 – Definem as portas do Grupo A como entrada ou saída.

Há três modos de operação: *Mode 0*, *Mode 1* e *Mode 2*. O *Mode 0* pode ser usado em todas as portas, e configura-as como leitura ou escrita, de forma que a transmissão dos dados é realizada sem que uma rotina de controle (*hand shaking*) seja executada. No *Mode 1*, as portas A e B podem ser configuradas como leitura ou escrita, sendo que os pinos da porta C são utilizados para troca do sinal de *hand shaking*. Por fim, o *Mode 2* pode ser usado apenas na porta A, que é configurada como barramento bi-direcional sendo utilizados 5 pinos da porta C para o protocolo de *hand shaking*.

No PC, os acessos ao registrador de controle e às portas para cada PPI da placa são feitos através dos endereços listados na Tabela 4.2.

Tabela 4. 2 - Endereços dos registradores dos dispositivos PPI.

Dispositivo	Registrador Controle	Registrador Porta A	Registrador Porta B	Registrador Porta C
PPI 1	EB+0013H	EB+0010H	EB+0011H	EB+0012H
PPI 2	EB+0017H	EB+0014H	EB+0015H	EB+0016H

4.2.3. Circuito de Temporização

O circuito de temporização é composto por um oscilador *Transistor-Transistor Logic* (TTL) de 10 MHz, um circuito divisor de freqüência e, como componentes principais, dois temporizadores 82C54 do fabricante *Intel*. O diagrama de blocos da Figura 4.3 representa o diagrama funcional do 82C54.

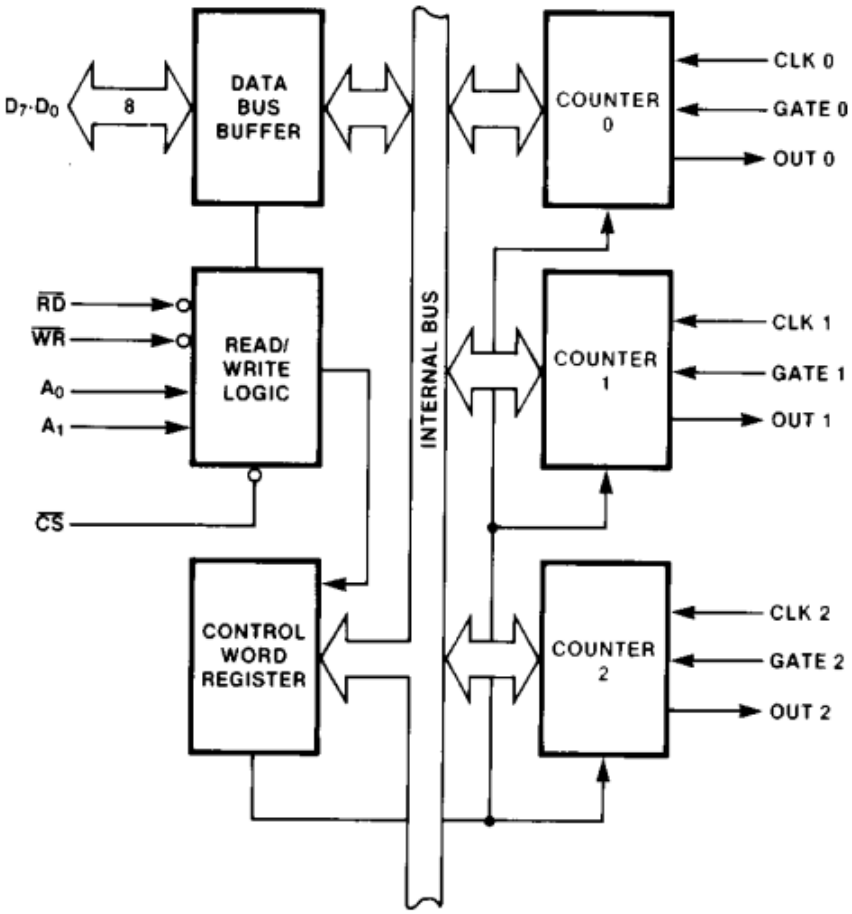

Figura 4. 3- Diagrama de blocos do 82C54 [10].

A interface de dados entre o 82C54 e o barramento do sistema é realizada pelo bloco *Data Bus Buffer*.

O bloco *Read/Write Logic* pode ser acessado através do barramento do sistema e gera sinais de controle para os demais blocos. As operações de escrita/leitura são configuradas a partir dos pinos A0, A1, RD, WR e CS, da forma:

- A0 e A1 - Selecionam um dentre os contadores ou o Registrador de Controle;
- RD - Habilidade a operação de leitura do item selecionado;
- WR - Habilita a operação de escrita no item selecionando;
- CS - Habilita a comunicação entre o barramento e o 82C54.

Todos os contadores operam de forma independente, são decrescentes e possuem 16 bits de resolução, podendo operar em diferentes modos. Para definir as operações dos

contadores podem ser utilizadas palavras de controle gravadas no registrador de controle (*Control Word Register*).

A palavra de controle é da forma D_7-D_6-D_5-D_4-D_3-D_2-D_1-D_0, onde:

- D_7 e D_6 – Definem o contador utilizado;
- D_5 e D_4 – Definem as operações de escrita/leitura
- D_3, D_2 e D_1 – Definem o modo de operação dos contadores;
- D_0 – Definem o modo de contagem entre binária pura ou binária codificada em decimal (BCD).

Há 6 modos de operação dos contadores: *Mode 0*, *Mode 1*, *Mode 2*, *Mode 3*, *Mode 4* e *Mode 5*. O *Mode 0* (Figura 4.4) executa uma contagem simples, iniciada após a escrita de uma nova contagem seguida de um pulso de relógio (*CLK*). Então, o pino *OUT* assume o nível lógico "0" até o final da contagem, quando assume o nível lógico "1", permanecendo com este nível até que seja iniciada uma nova contagem.

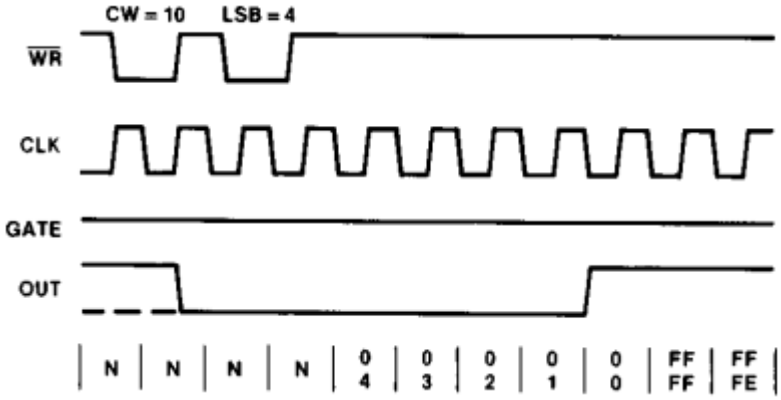

Figura 4. 4 - *Mode* 0 [10].

No *Mode 1* (Figura 4.5), após a escrita de uma nova contagem, o pino *OUT* é iniciado com o nível lógico "1", com o qual permanece até que uma borda ascendente seja detectada no pino *GATE*. Então, a contagem é iniciada após o próximo pulso de *CLK*, fazendo *OUT* assumir o nível lógico "0", com o qual permanece até o final da contagem, que é sinalizada com um nível lógico "1", mantido até a próxima borda positiva do sinal de *GATE*.

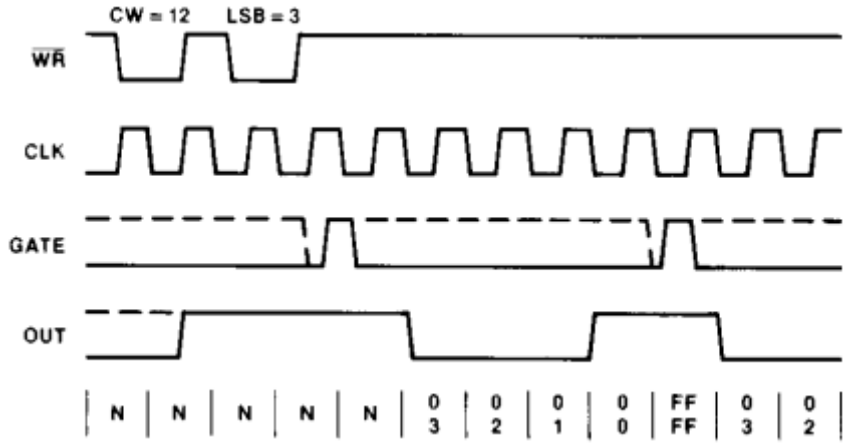

Figura 4. 5 - *Mode* 1 [10].

O *Mode 2* (Figura 4.6) é tipicamente utilizado para gerar interrupções de *Real Time Clock* (RTC). O pino *OUT* é iniciado com o nível lógico "1". Após um pulso de *CLK* ocorrido em seqüência à escrita, a contagem N é iniciada, sendo atribuído o nível lógico "0" ao pino *OUT* a cada N pulsos de *CLK*, no qual permanece pelo período de um pulso de *CLK*. Este processo repete-se periodicamente.

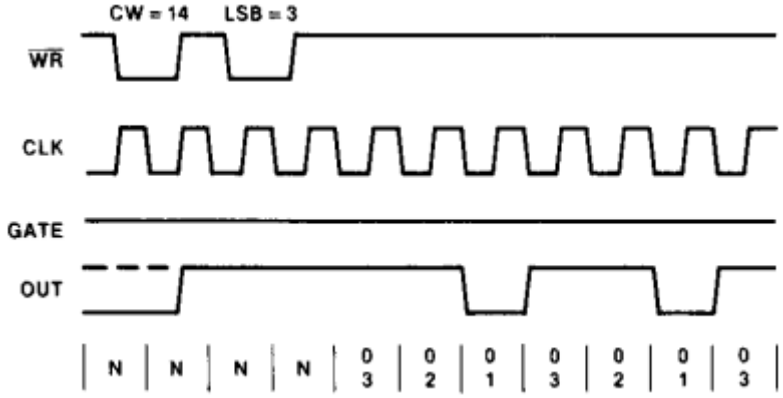

Figura 4. 6 - *Mode* 2 [10].

O *Mode 3* (Figura 4.7) tem seu funcionamento semelhante ao *Mode 2*, exceto pelo fato de *OUT* permanecer no nível lógico "0" durante metade da contagem N. Resultando numa onda quadrada de período igual a N.

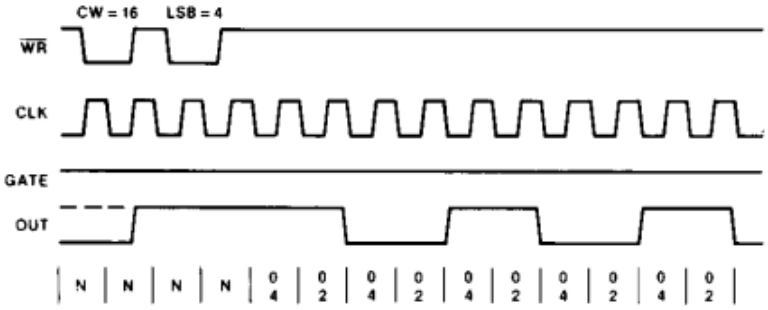

Figura 4. 7 - *Mode* 3 [10].

No *Mode 4* (Figura 4.8) o pino OUT é iniciado com o nível lógico “1”, e assume o nível lógico “0” durante um pulso de CLK ao final da contagem. O diferencial é que a contagem é disparada apenas pela escrita de uma nova contagem, sem que seja necessário aguardar bordas de subida/descida ou pulsos de *CLK*.

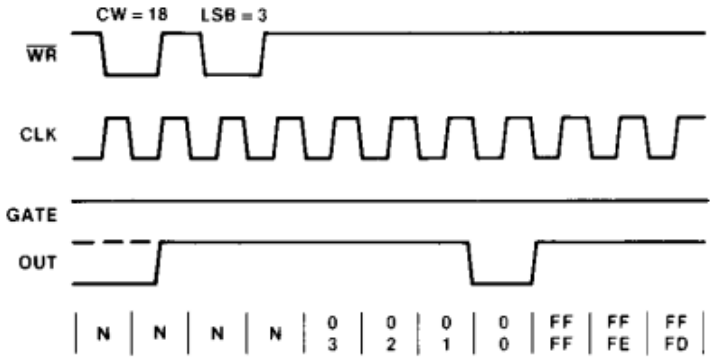

Figura 4. 8 - *Mode* 4 [10].

No *Mode 5* (Figura 4.9) o pino *OUT* é inicializado com o nível lógico “1”. Após uma borda de subida no sinal de *GATE* seguida de um pulso de *CLK*, a contagem é iniciada e ao seu final *OUT* assume o nível lógico “0” durante um pulso de (*CLK*).

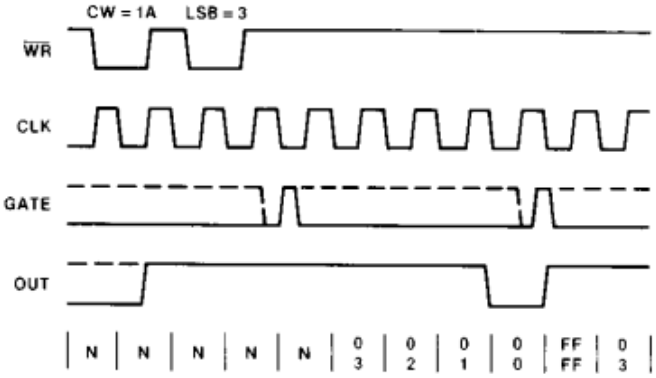

Figura 4. 9 - *Mode* 5 [10].

O acesso ao registrador de controle, bem como aos registradores de contagem é feito mediante os endereços apresentados na Tabela 4.3.

Tabela 4. 3 - Endereços de acesso aos registradores dos temporizadores Intel.

Dispositivo	Registrador Controle	Registrador Contador 1	Registrador Contador 2	Registrador Contador 3
Temporizador 1	EB+001BH	EB+0018H	EB+0019H	EB+001AH
Temporizador 2	EB+001FH	EB+001CH	EB+001DH	EB+001EH

Para ativar os sinais de *Clock* e *Gate* do circuito de temporização, há 4 contadores ligados em cascata, sendo possível dividir a freqüência do oscilador por 5, 10, 50, 100, 500, 1000, 5000 e 10000. Nas Tabelas 4.4 a 4.7 são listadas as configurações dos *Strap's* para os sinais de *Clock* e *Gate* do circuito de temporização.

Tabela 4. 4 - Configuração para frequência de *Clock*.

Temporizador 1					
Freqüência	J1	J2	J3	J4	J5
2 MHz	☒	☐	☐	☐	☐
1 MHz	☐	☒	☐	☐	☐
200 KHz	☐	☐	☒	☐	☐
100 KHz	☐	☐	☐	☒	☐
10 MHz*	☐	☐	☐	☐	☒
Temporizador 2					
Freqüência	J6	J7	J8	J9	J10
2 MHz	☒	☐	☐	☐	☐
1 MHz	☐	☒	☐	☐	☐
200 KHz	☐	☐	☒	☐	☐
100 KHz	☐	☐	☐	☒	☐
10 MHz*	☐	☐	☐	☐	☒

* Configuração atual da placa.
☒ Contatos fechados
☐ Contatos abertos

Tabela 4. 5 - Configuração para frequência de *Gate*.

Temporizador 1					
Freqüência	J11	J12	J13	J14	J15
1 KHz	☒	☐	☐	☐	☐
2 KHz	☐	☒	☐	☐	☐
10 KHz	☐	☐	☒	☐	☐
20 KHz*	☐	☐	☐	☒	☐
VCC	☐	☐	☐	☐	☒

Temporizador 2					
Freqüência	J16	J17	J18	J194	J20
1 KHz	☒	☐	☐	☐	☐
2 KHz	☐	☒	☐	☐	☐
10 KHz	☐	☐	☒	☐	☐
20 KHz*	☐	☐	☐	☒	☐
VCC	☐	☐	☐	☐	☒

* Configuração atual da placa.
☒ Contatos fechados
☐ Contatos abertos

Tabela 4. 6 - Configuração para sinal de *clock*.

Temporizador 1			
Tipo de sinal	Contador 0	Contador 1	Contador 2
	S3	S2	S1
Interno*	1-2 ☒	1-2 ☒	1-2 ☒
Externo	2-3 ☒	2-3 ☒	2-3 ☒
Temporizador 2			
Tipo de sinal	Contador 0	Contador 1	Contador 2
	S10	S11	S12
Interno*	1-2 ☒	1-2 ☒	1-2 ☒
Externo	2-3 ☒	2-3 ☒	2-3 ☒

* Configuração atual da placa.
☒ Contatos fechados
☐ Contatos abertos

Tabela 4. 7 - Configuração para sinal de *Gate*.

Temporizador 1			
Tipo de sinal	Contador 0	Contador 1	Contador 2
	S6	S5	S4
Interno*	1-2 ☒	1-2 ☒	1-2 ☒
Externo	2-3 ☒	2-3 ☒	2-3 ☒
Temporizador 2			
Tipo de sinal	Contador 0	Contador 1	Contador 2
	S7	S8	S9
Interno*	1-2 ☒	1-2 ☒	1-2 ☒
Externo	2-3 ☒	2-3 ☒	2-3 ☒

* Configuração atual da placa.
☒ Contatos fechados
☐ Contatos abertos

A entrada dos sinais externos, tanto para *Clock*, quanto para *Gate*, é efetuada através dos conectores CN4 e CN5, respectivamente, sendo a relação entre pinos do conector e o respectivo sinal descrita na Tabela 4.8.

Tabela 4. 8 - Relação entre pinos e sinal de entrada.

Temporizador 1			
Contador	Pino	Conector *Clock*	Conector *Gate*
contador 0	pino 1	CN4	CN5
contador 1	pino 2	CN4	CN5
contador 2	pino 3	CN4	CN5
Temporizador 2			
Contador	Pino	Conector *Clock*	Conector *Gate*
contador 0	pino 5	CN4	CN5
contador 1	pino 7	CN4	CN5
contador 2	pino 9	CN4	CN5
GND			
Contador	Pino	Conector *Clock*	Conector *Gate*
GND	pino 10	CN4	CN5

A saída dos temporizadores é tomada no conector CN8, cuja relação com os pinos dos temporizadores é descrita na Tabela 4.9.

Tabela 4. 9 - Relação entre pinos e sinal de saída.

Temporizador 1	
Pino	Saída
pino 1	contador 0
pino 3	contador 1
pino 5	contador 2
pino 7	GND
Temporizador 2	
Pino	Saída
pino 2	contador 0
pino 4	contador 1
pino 6	contador 2
pino 8	GND

4.2.4. Circuito Conversor A/D

O circuito de conversão A/D é formado por *buffer*, filtro ativo passa-baixa de 2ª ordem e o conversor A/D AD1674JN do fabricante *Analog Devices*.

O filtro ativo apresenta uma freqüência de corte determinada pela expressão:

$$f_c = \frac{1}{2\pi\sqrt{RaRbCaCb}}$$

Onde Ra, Rb, Ca e Cb são componentes identificados na placa segundo a Tabela 4.10.

Tabela 4. 10 - Componentes do filtro ativo.

Conversor	Ra	Rb	Ca	Cb
1	R4	R5	C2	C3
2	R6	R7	C5	C6
3	R6	R9	C9	C8

O AD1674 é um conversor A/D com 12 bits de resolução, taxa de conversão de 10μs. Seu diagrama funcional do conversor A/D pode ser visualizado na Figura 4.10.

Quando um comando de conversão é definido o *Sample-and-Hold Amplifier* (SHA) é configurado para modo de espera, o temporizador é habilitado e o Registrador de Aproximações Sucessivas (SAR) é reiniciado. Uma vez iniciado um ciclo de conversão, ele não pode ser interrompido. Quando o SAR indica o final da conversão, o bloco de controle (*Control*) desabilita o relógio, configura o SHA para o modo de amostragem, o que permite a aquisição dos 12 bits alvo da conversão.

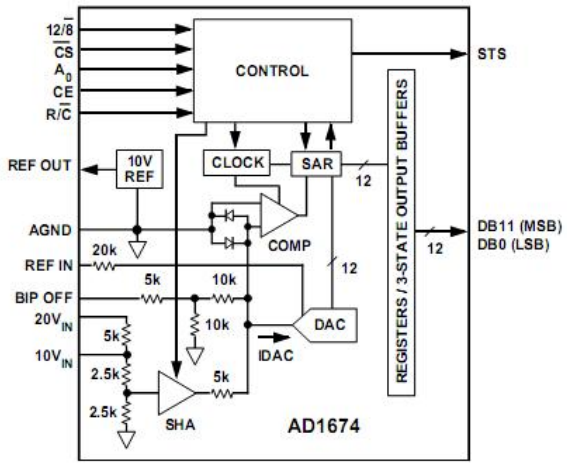

Figura 4. 10 - Diagrama funciona do AD1674 [11].

A operação do conversor A/D é gerenciada através dos pinos CE, CS, R/C, 12/8 e A_0, da forma seguinte:

CE e CS – Habilita o conversor;

R/C – Define as operações de leitura ou conversão;

12/8 – Define o modo de transferência de dados;

A_0 – Define os modos de conversão;

O conversor permite a conversão de sinais bipolares ou unipolares com até 20 V de pico-a-pico (Vpp). Estas configurações são realizadas com ajustes de hardware, feitos na placa através de *Strap's*. A Tabela 4.11 descreve as configurações possíveis.

Tabela 4. 11 - Configuração dos conversores A/D.

A/D	Transferência de dados		Nível do Sinal		Tipo de Sinal	
1	*Strap* S15		*Strap* S14		*Strap* S13	
	8 *bits*	12 *bits*	10 Vpp	20 Vpp	Unipolar	Bipolar
	1-2 ☒*	2-3 ☒	1-2 ☒*	2-3 ☒	1-2 ☒	2-3 ☒*
2	*Strap* S18		*Strap* S17		*Strap* S16	
	8 *bits*	12 *bits*	10 Vpp	20 Vpp	Unipolar	Bipolar
	1-2 ☒*	2-3 ☒	1-2 ☒*	2-3 ☒	1-2 ☒	2-3 ☒*
3	*Strap* S21		*Strap* S20		*Strap* S19	
	8 *bits*	12 *bits*	10 Vpp	20 Vpp	Unipolar	Bipolar
	1-2 ☒*	2-3 ☒	1-2 ☒*	2-3 ☒	1-2 ☒	2-3 ☒*

* Configuração atual da placa.
☒ Contatos fechados
☐ Contatos abertos

A placa foi projetada de tal forma que os conversores A/D possam operar em conjunto, isto é, a amostragem e conversão realizadas, ocorrem simultaneamente e de forma paralela. Isto permite que 3 sinais distintos possam ser amostrados e convertidos no mesmo instante.

O grupo de conversores possue endereços de leitura de dados individuais e um endereço acessado quando se deseja iniciar uma conversão. Na Tabela 4.12 são listados os endereços de leitura de cada conversor, assim como o endereço de início de conversão.

Tabela 4. 12 - Endereços dos conversores A/D.

Conversor	Byte Menos Significativo	Byte Mais Significativo
1	EB+0020H	EB+0021H
2	EB+0024H	EB+0025H
3	EB+0028H	EB+0029H
Endereço de Início de Conversão		
EB+0021H		

A leitura ocorre de forma que o dado lido terá o seguinte formato:

$$D_{11}\text{-}D_{10}\text{-}D_{9}\text{-}D_{8}\text{-}D_{7}\text{-}D_{6}\text{-}D_{5}\text{-}D_{4}\text{-}D_{3}\text{-}D_{2}\text{-}D_{1}\text{-}D_{0}\text{-X-X-X-X}$$

Os 4 bits menos significativos lidos são bits espúrios que devem ser eliminados da palavra de dados. O procedimento é um deslocamento lógico da palavra de dados à direita buscando uma palavra de dados da forma:

$$0\text{-}0\text{-}0\text{-}0\text{-}D_{11}\text{-}D_{10}\text{-}D_9\text{-}D_8\text{-}D_7\text{-}D_6\text{-}D_5\text{-}D_4\text{-}D_3\text{-}D_2\text{-}D_1\text{-}D_0$$

Um processo de conversão tem início quando o PC escreve qualquer dado no endereço de início de conversão.

Os sinais analógicos são obtidos através de cabos coaxiais, os quais se interligam ao meio externo, por meio de conectores BNC. Na Tabela 4.13 são discriminados os conectores e o respectivo circuito de conversão A/D ao qual está ligado.

Tabela 4. 13 - Relação entre conector BNC e circuito conversor A/D.

Conversor	Conector
1	BNC1
2	BNC2
3	BNC3

4.2.5. Circuito Conversor D/A

A placa de aquisição possui um conversor D/A de 12 bits, com 4 canais de saída independentemente programáveis. O circuito não requer nenhuma programação por hardware. O conversor AD75004, do fabricante *Analog Devices*, está configurado para gerar tensões na faixa de ± 5.0 V, e seu diagrama funcional pode ser visualizado na Figura 4.11.

A operação do conversor é administrada através dos pinos CS, WR, A3, A2, A1 e A0, onde:

CS – Habilita a utilização do dispositivo;

WR – Habilita as operações de escrita no dispositivo;

A3 e A2 – Definem a operação a ser realizada;

A1 e A0 – Definem o canal de saída analógica.

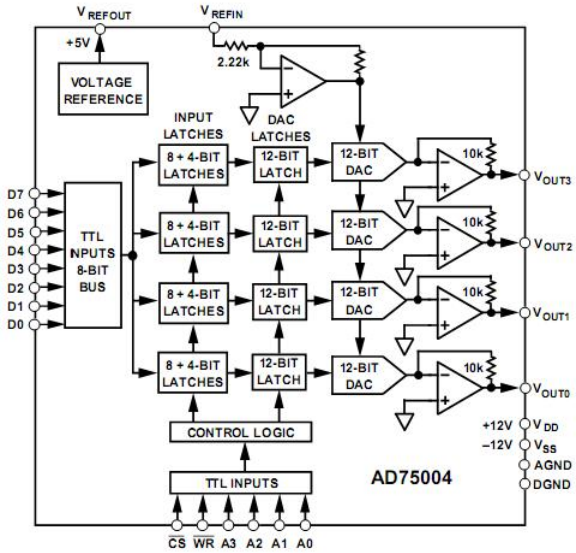

Figura 4. 11 - Diagrama funcional do AD75004 [12].

Para realizar uma conversão é necessário cumprir os seguintes procedimentos:

1. Gravar uma palavra-código de 12 bits no registrador de um dos canais do conversor (não há uma seqüência definida, podendo vir primeiro os 4 bits mais significativos e em seguida os 8 bits menos significativos ou vice-versa);
2. Atualizar o canal para representar o novo valor digital a converter.

Os endereços para gravação das palavras-código são apresentados na Tabela 4.14.

Tabela 4. 14 - Endereços para conversão D/A.

Canal	8 Bits Menos Significativos	4 Bits Mais Significativos	Atualizar Canal
0	EB + 0002H	EB + 0001H	EB + 0002H
1	EB + 0004H	EB + 0005H	EB + 0006H
2	EB + 0008H	EB + 0009H	EB + 000AH
3	EB + 000CH	EB + 000DH	EB + 000EH

O conversor D/A trabalha com números binários com sinal. Assim, os valores digitais entre 000H e 7FFH, são considerados os números positivos +0.0 V e +5.0V, respectivamente. Os valores de 800H a FFFH representam os números binários negativos -0.0V e -5.0V, respectivamente.

5. Sistema Operacional

Um sistema operacional é um *software* ou um conjunto de *softwares* cuja função é mediar as interações entre um computador e o usuário [13].

Para a finalidade buscada no instrumento, o sistema operacional a ser utilizado deve oferecer uma interface ao sistema de *hardware* (Placa de Aquisição), a qual é realizada através da escrita/leitura de palavras código em endereços de registradores específicos.

Outra característica importante é uma capacidade de temporização de tarefas de forma determinística e numa alta resolução. Isto porque, a aquisição de dados deverá ser realizada a intervalos de tempo fixos e de curta duração (da ordem de microssegundos) para que a construção da forma de onda do sinal amostrado seja representativa do sinal real.

Um sistema operacional que pode atender principalmente o requisito de determinismo no uso de temporizadores é conhecido com Sistema Operacional de Tempo Real (ou RTOS - *Real Time Operating System*). Este tipo de sistema operacional é destinado à execução de múltiplas tarefas, onde o tempo de resposta a um evento (externo ou interno) é pré-definido [14]. Esse tempo de resposta é chamado de prazo da tarefa. A perda de um prazo, isto é, o não cumprimento de uma tarefa dentro do prazo esperado, caracteriza uma falha do sistema.

Aplicações determinísticas geralmente fazem interações entre tarefas críticas e todas consomem uma quantidade mensurável de tempo de processamento. Assim, aplicações determinísticas são avaliadas não somente pela velocidade, mas pela confiabilidade nas respostas às entradas e fornecimento de saídas com um tempo médio muito pequeno [15].

Outro aspecto importante do RTOS é a previsibilidade. O sistema é considerado previsível quando podemos antecipar seu comportamento independentemente de falhas, sobrecargas e variações de *hardware* [16].

Em instrumentos virtuais, o determinismo refere-se ao sincronismo entre a aquisição de dados e o processamento de sinais [17]. De forma que se tornam possíveis analises que dependem de uma resolução de tempo fixa, tal como *Fastest Fourier Transform* (FFT).

Em razão da popularidade e da complexidade adicionada ao *Microsoft Windows*®, muitos usuários o adotam como seu sistema operacional de uso pessoal. Porém, em suas versões comuns não oferece suporte a tarefas necessárias a um RTOS. Isto porque ele é um sistema operacional projetado para um propósito generalista de fácil utilização por todo tipo de usuário com um sistema interativo para *Desktops* ou Servidores [18].

As principais deficiências encontradas no *Microsoft Windows*® ao utilizá-lo em tarefas de tempo real são: um grande conjunto de prioridades, não-determinismo em decisões temporizadas e inversões de prioridades, particularmente em interrupções de processos [18].

Entretanto, as limitações do *Microsoft Windows*® podem ser contornadas com o uso de um conjunto de *softwares* (conhecidos com extensões) que agregam novas características ao núcleo do sistema. Um exemplo de extensão é o *software* RTX da empresa IntervalZero [19]. Na Tabela 5.1 é possível avaliar a comparação entre as performances do *Microsoft Windows*® em determinadas tarefas.

Tabela 5. 1 - Performance metrics for Windows XP with RTX 5.1 running on a 800 MHz Pentium III processor with an ACPI compliant chipset [18].

Operation	Windows XP	Windows CE 3.0	RTX 5.1
SetEvent (no thread switch): min/max in µs	1.04 / 5000+	1.49 / 7.20	0.29 / 2.71
SetEvent > WFSO: min/max in µs	1.38 / 5000+	2.46 / 10.7	0.60 / 2.96
ReleaseMutex > WFSO: min/max in µs	1.49 / 5000+	3.51 / 10.5	0.70 / 3.26
ReleaseSemaphore >WFSO: min/max in µs	1.39 / 5000+	3.00 / 9.40	0.61 / 3.43
Yield: min/max in µs	1.11 / 5000+	1.32 / 8.34	0.33 / 3.37
Thread priority change: min/max in µs	1.31 / 5000+	1.41 / 8.96	0.56 / 3.81
Interrupt service thread dispatch: min/max in µs	4.3 / 5000+	4.3 / 26	2.0 / 19
Win32-to-RTSS SetEvent call: min in µs	NA	NA	14

NA – Não avaliado.

A dificuldade de acesso a estas extensões está no custo envolvido, o que inviabiliza sua popularização no meio acadêmico, sendo mais atraente dentro da iniciativa industrial.

Contudo, no mundo do *software* livre, o sistema operacional GNU/Linux pode ser utilizado como sistema base para obtenção de um RTOS. Apesar de comumente as distribuições não apresentem esta característica, há distribuições sendo desenvolvidas de forma a atender esse tipo de demanda. Este é o caso do OSADL *Project* desenvolvido pelo *Open Source Automation Development Lab* [20].

Outras diferentes maneiras de transformar uma distribuição comum do Linux num RTOS estão associadas ao uso de extensões para o *kernel* na forma de módulos carregáveis em tempo de execução, como é o caso do *Real-Time Aplication Interface* (RTAI) [21], assim como na utilização *patches* distribuídos por desenvolvedores Linux, os quais realizam modificações no próprio *kernel*, para suporte *Real-Time* [22].

O uso dos *patches* distribuídos por desenvolvedores Linux se mostrou mais atraente, tendo em vista o custo zero de aquisição, além da complexidade oferecida por este sistema, sendo portanto, o utilizado neste trabalho. A distribuição escolhida foi *Slackware 12.2*, pela sua compatibilidade com o PC utilizado.

6. Configuração do Sistema Operacional

Para início de investigação foi necessário configurar o sistema operacional de forma a atender os requisitos de determinismo no tempo. Isto é feito com a recompilação do *kernel* Linux. Para isso, é preciso um código fonte da versão selecionada e do *patch* que adiciona características de um sistema RTOS. Estes arquivos estão disponíveis em http://www.kernel.org.

A compilação doe um *kernel* Linux requer um grande conhecimento do *hardware* envolvido. Isto pode ser vantajoso, tendo em vista que pode ser criado um sistema operacional sob medida para a uma determinada finalidade. Mais informações sobre como recompilar um *kernel* no Linux podem ser encontradas em [23]. Os códigos fonte usados foram:

- *linux-2.6.29.4.tar.bz2*;
- *patch-2.6.29.4-rt19.bz2*.

Durante a compilação é preciso ter certeza de que alguns itens de configuração estão selecionados de forma correta no menu *Processor type and features*. Neles você deve escolher a opção de adicionar suporte a temporizadores de alta resolução (Figura 6.1), escolher a família de processadores adequada (Figura 6.2) e adicionar a característica de tempo real ao seu sistema operacional (Figura 6.3).

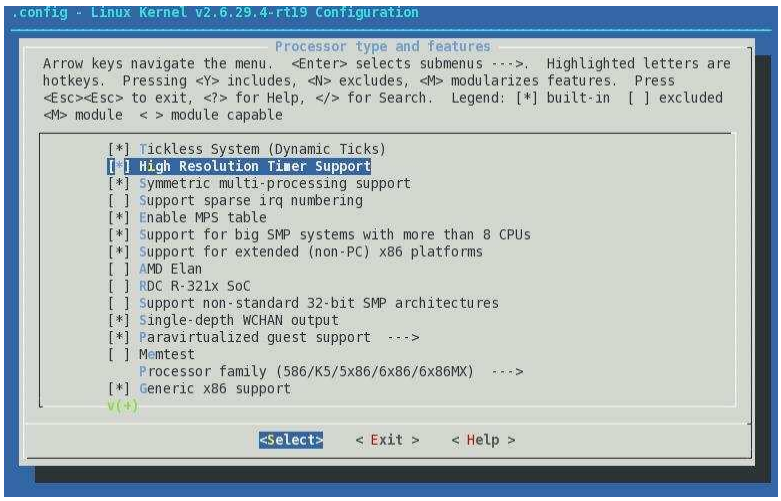

Figura 6. 1 - *High Resolution Timer Support* (Suporte a Temporizadores de Alta Resolução).

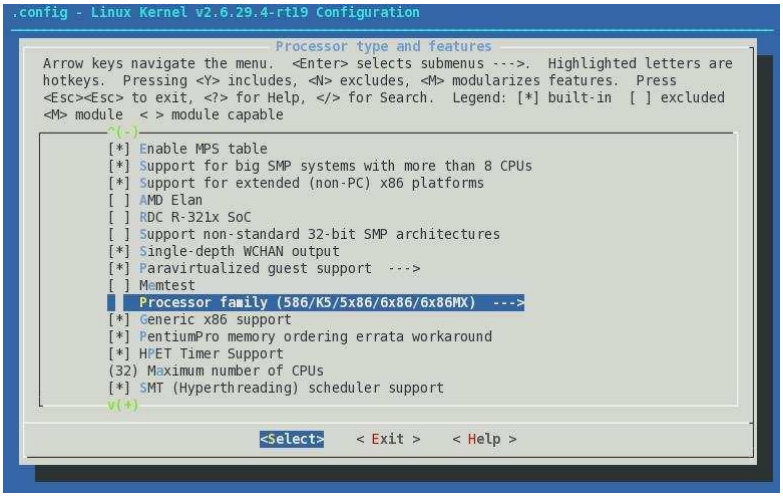

Figura 6. 2 - *Processor family* (Família do Processador).

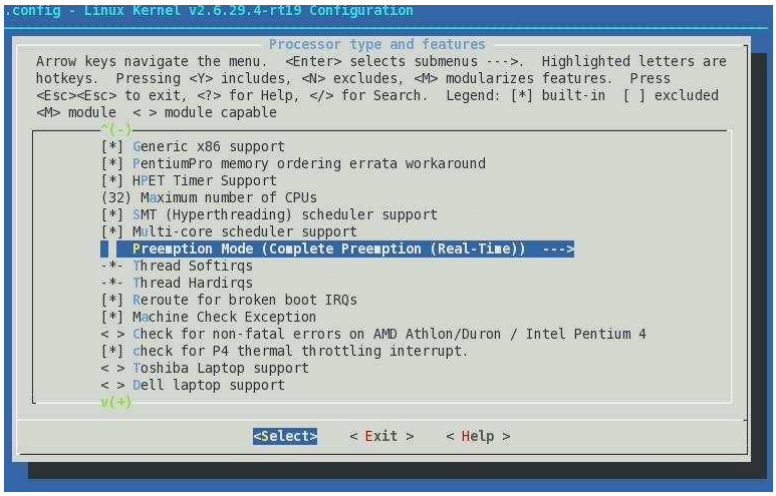

Figura 6. 3 - *Preemption Mode* (Modo de Preempção).

Tendo feito isso e prosseguindo com a compilação, o sistema operacional deve apresentar atributos de um RTOS. As principais verificações de que o sistema foi configurado com sucesso são a versão do *kernel* (Figura 6.4) e lista de temporizadores de alta resolução disponíveis (Figuras 6.5 a 6.7).

Figura 6. 4 - Versão do *kernel* Linux.

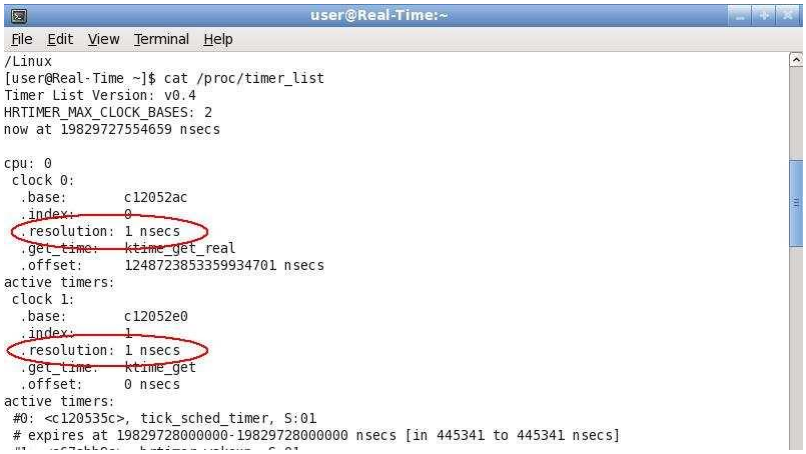

Figura 6. 5 - Resolução dos temporizadores disponíveis.

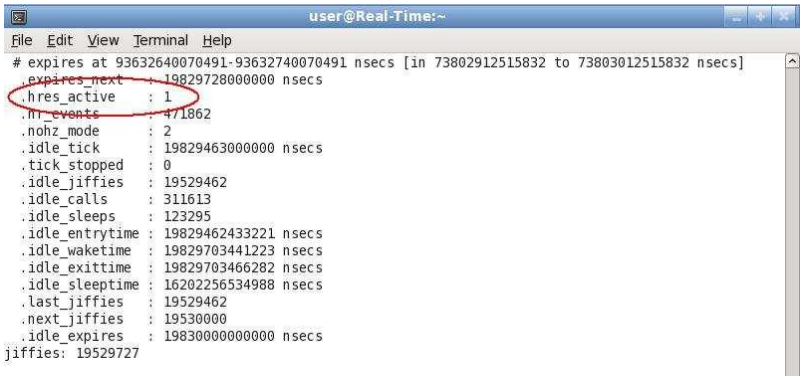

Figura 6. 6 - Temporizadores de alta resolução ativos.

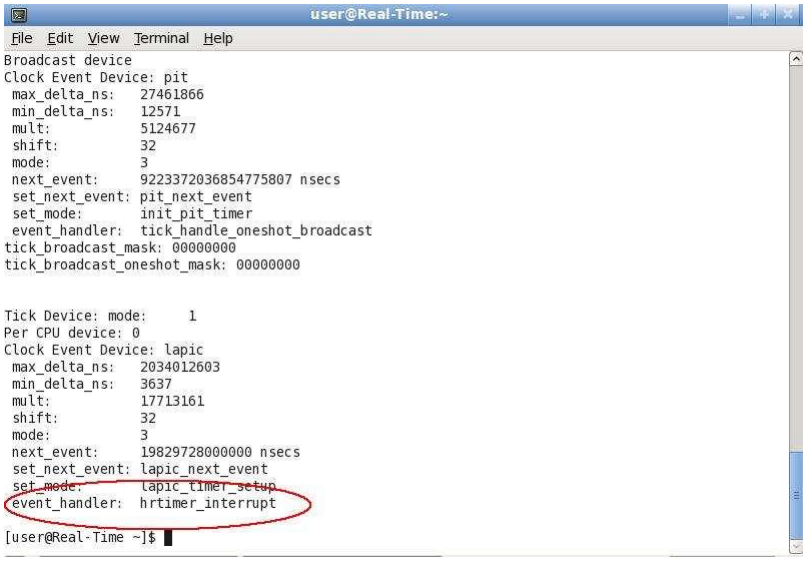

Figura 6. 7 - Temporizador de alta resolução.

Apesar de ser um projeto genérico, pode ser observado em [22] que há um numero limitado de plataformas onde o projeto foi testado com êxito. As Figuras 6.4 a 6.7 foram

geradas após a configuração de máquina com processador AMD Turion 64 X2 (TL-50), 1,6 GHz, 256 MB de memória RAM, com um *kernel* 2.6.29.4-rt19.

O mesmo procedimento realizado no PC alvo do projeto exibiu resultados que divergem do modelo esperado. Quanto à versão do *kernel*, o resultado foi obtido de forma satisfatória (Figura 6.8). Entretanto, não foi possível constatar que os temporizadores tenham o suporte a altas resoluções habilitado de maneira correta (Figura 6.9 a 6.11).

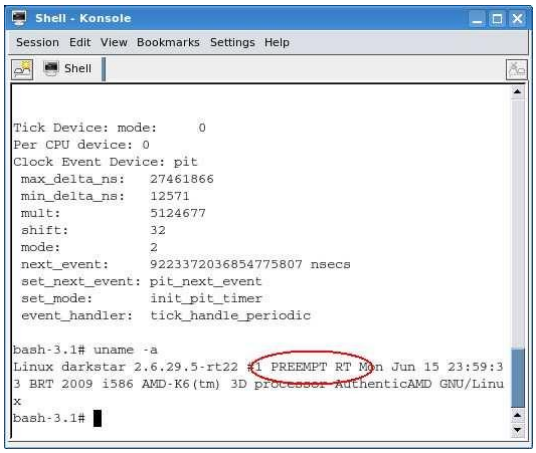

Figura 6. 8 - Versão do *kernel* para o PC alvo.

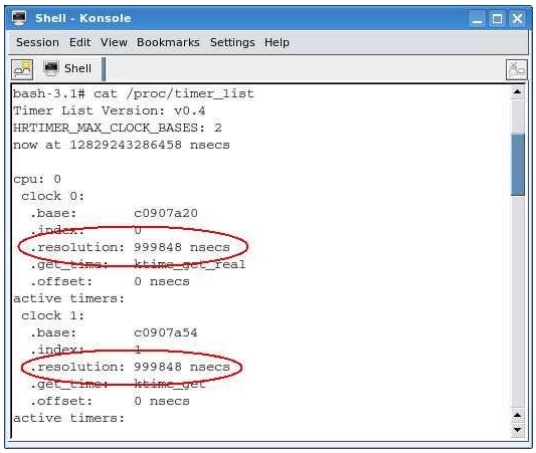

Figura 6. 9 - Resolução dos temporizadores no PC alvo.

A partir das Figuras 6.9 a 6.11 é possível supor que a compilação do *kernel* com o projeto RTOS não foi bem sucedida. Isto pode ser associado a limitações do *hardware* utilizado ou à má configuração do *kernel*.

Apesar de válida, a investigação dos problemas listados anteriormente não foi aprofundada em razão de uma limitação no prazo para conclusão do presente trabalho. Os indícios de uma má configuração do *kernel* representaram um ponto deficiente no projeto. Ela

afetou de maneira significativa a possibilidade de realizar a aquisição de dados em altas resoluções, cuja principal aplicação é a correta leitura de sinal com altas freqüências.

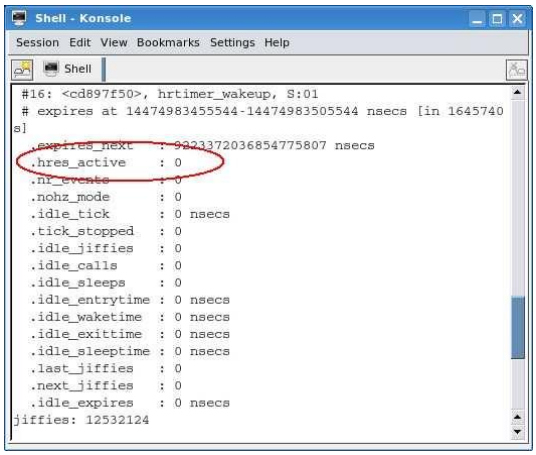

Figura 6. 10 - Temporizadores de alta resolução ativos no PC alvo.

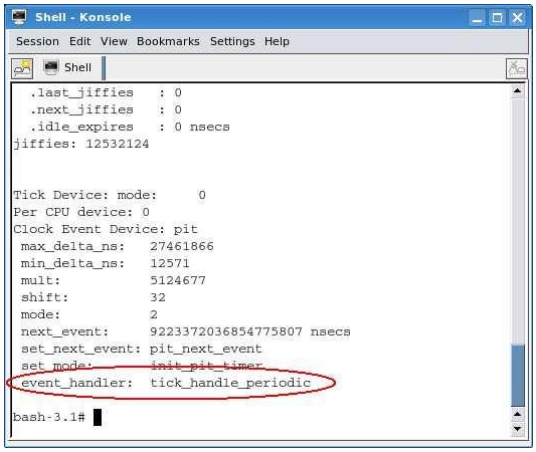

Figura 6. 11 - Temporizador do PC alvo.

Uma amostra da aquisição de um sinal no sistema obtido é visualizada na Figura 6.12. A forma de onda amostrada representa um sinal senoidal gerado a 60 Hz com amplitude de 5 V. A montagem final do sinal parece ser resultado de uma aquisição corretamente realizada. Porém, as condições para isso foram que apenas a aplicação estava em execução (além das tarefas de segundo plano) sem que nem mesmo o mouse fosse utilizado.

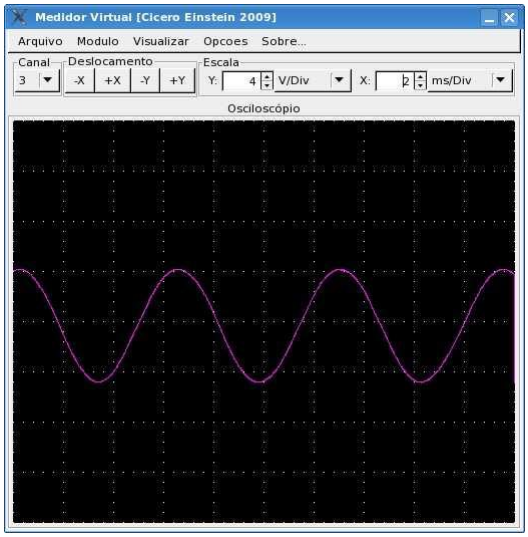

Figura 6. 12 - Amostra de utilização do sistema final.

A simples utilização do mouse requer uma interrupção com alta prioridade em diversos sistemas operacionais. No caso de um RTOS esta prioridade poderia ser negada em favor de outra tarefa. Mas no sistema obtido isso não foi comprovado, um vez que ao utilizar o mouse o sinal adquirido sofre uma grande distorção (Figura 6.13).

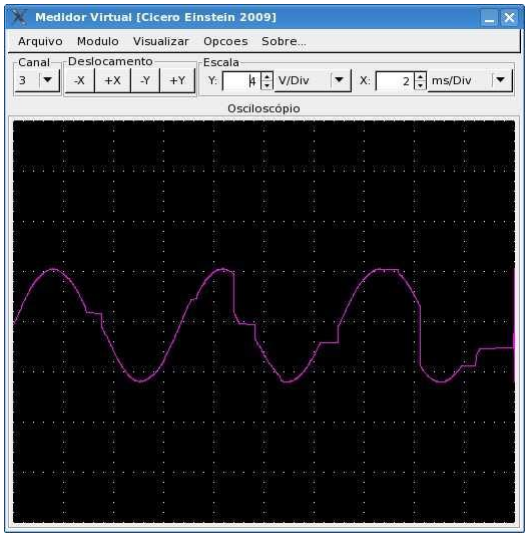

Figura 6. 13 - Amostra de utilização do sistema simultaneamente com o mouse.

Uma ferramenta para avaliação do desempenho do sistema RTOS configurado é disponível em [24]. Esta ferramenta mede a precisão na execução sucessiva de operações de espera (*Delay*), sendo possível determinar a prioridade da tarefa executada. Os resultados de quatro testes realizados no PC alvo são apresentados a seguir.

```
Teste 1:
bash-3.1# ./cyclictest -l 10000 -i 10 -p 80 -t1 -n
WARNING: High resolution timers not available
policy: fifo: loadavg: 0.20 0.08 0.23 2/96 2703

T: 0 ( 2703) P: 80 I: 10 C: 10000 Min: 600 Act: 9897294 Avg: 4948925 Max: 9897294

Teste 2:
bash-3.1# ./cyclictest -l 10000 -i 100 -p 80 -t1 -n
WARNING: High resolution timers not available
policy: fifo: loadavg: 0.29 0.11 0.24 2/96 2705

T: 0 ( 2705) P: 80 I: 100 C: 10000 Min: 174 Act: 8996945 Avg: 4498554 Max: 8996945

Teste 3:
bash-3.1# ./cyclictest -l 10000 -i 1000 -p 80 -t1 -n
WARNING: High resolution timers not available
policy: fifo: loadavg: 0.25 0.12 0.23 2/96 2707

T: 0 ( 2707) P: 80 I: 1000 C: 10000 Min: 48 Act: 327 Avg: 551 Max: 1101

Teste 4:
bash-3.1# ./cyclictest -l 10000 -i 10000 -p 80 -t1 -n
WARNING: High resolution timers not available
policy: fifo: loadavg: 1.13 0.46 0.33 2/96 2709

T: 0 ( 2709) P: 80 I: 10000 C: 10000 Min: 48 Act: 632 Avg: 572 Max: 1113
```

O parâmetro -l define o numero de ciclos executados, -i define o intervalo de tempo de cada ciclo em microssegundos, -p define o nível de prioridade da tarefa, -t define o numero de tarefas a serem executadas e -n determina que a resolução do contador utilizado seja da ordem de nanossegundos. As respostas aos testes são as latências mínima (Min), média (Avg) e máxima (Max) obtidas.

Os mesmos testes foram executados na máquina com o *kernel* 2.6.29.4-rt19 para efeito de comparação. Os resultados obtidos são expostos no quadro a seguir.

```
Teste 1:
[root@Real-Time rt-tests]# ./cyclictest -l 10000 -i 10 -p 80 -t1 -n
policy: fifo: loadavg: 0.00 0.02 0.01 2/224 2111

T: 0 ( 2111) P: 80 I: 10 C: 10000 Min: 0 Act: 18 Avg: 952 Max: 6790
```

```
Teste 2:
[root@Real-Time rt-tests]# ./cyclictest -l 10000 -i 100 -p 80 -t1 -n
policy: fifo: loadavg: 0.00 0.01 0.00 2/224 2113

T: 0 ( 2113) P: 80 I: 100 C: 10000 Min: 2 Act: 211 Avg: 742 Max: 8585

Teste 3:
[root@Real-Time rt-tests]# ./cyclictest -l 10000 -i 1000 -p 80 -t1 -n
policy: fifo: loadavg: 0.00 0.01 0.00 1/224 2115

T: 0 ( 2115) P: 80 I: 1000 C: 10000 Min: 0 Act: 27 Avg: 699 Max: 6909

Teste 4:
[root@Real-Time rt-tests]# ./cyclictest -l 10000 -i 10000 -p 80 -t1 -n
policy: fifo: loadavg: 0.02 0.01 0.00 2/224 2117

T: 0 ( 2117) P: 80 I: 10000 C: 10000 Min: 1 Act: 435 Avg: 706 Max: 12290
```

Primeiramente, é importante observar a sinalização da falta de um temporizador de alta resolução no PC alvo. Isto reforça a hipótese de uma má configuração do *kernel*. Analisando os valores médios de latência, percebe-se que no PC alvo intervalos menores que 1 ms, produzem resultados sem representatividade. Os quais se apresentam mais significativos com intervalos maiores ou iguais a 1 ms. Entretanto, fora de padrões esperados para um sistema RTOS.

A segunda máquina configurada (AMD Turion) apresenta valores estáveis em todos os testes. Isto atesta o funcionamento de temporizadores em diversas resoluções de tempo. Entretanto, os intervalos também se mantêm fora dos modelos de um sistema RTOS.

7. Instrumento Virtual

A implementação de *softwares* que necessitam de características temporais bem definidas, requer o uso de uma linguagem de programação de alto nível e que ofereça requisitos de rapidez e estabilidade.

O instrumento virtual foi implementado em linguagem C, tendo sua interface gráfica baseada na biblioteca *Portable User Interface* (IUP) [25]. Uma visão geral da interface é demonstrada na Figura 7.1.

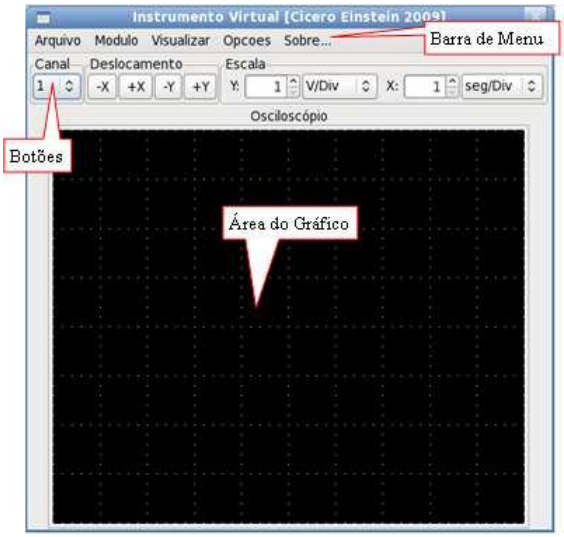

Figura 7. 1 – Visão geral da interface do Instrumento Virtual.

A interface é composta de uma Barra de Menu, um conjunto de Botões e a Área do Gráfico. No menu Arquivo está ativa apenas a opção de encerrar a aplicação, cujo submenu correspondente é nomeado Sair. Em Módulo está ativa a opção de selecionar a função Osciloscópio. Neste menu as demais funções não estão ativas em razão de sua não implementação. O menu Visualizar permite a seleção dos canais que serão traçados na Área de gráfico para um determinado módulo. Os menus Opções e Sobre se encontram inativos no momento.

Utilizando o conjunto de Botões é possível modificar o Deslocamento e as Escalas (eixos X e Y) do Canal selecionado. Cada escala é definida pelo seu valor absoluto e um fator multiplicador, que é na realidade um múltiplo da unidade aplicada a cada eixo.

Para o funcionamento correto do instrumento é necessário apenas o circuito de conversores A/D da placa de aquisição de dados. A placa é associada ao barramento ISA do PC, sendo comandada via acessos aos endereços de configuração e operação descritos na subseção 4.2.4.

A ação do instrumento é executada segundo eventos gerados pelo sistema operacional e interpretados por rotinas de atendimento a cada evento. As aquisição e conversão dos dados são realizadas segundo o fluxograma mostrado na Figura 7.2.

Durante a execução do aplicativo, o processo de aquisição é executado continuamente. Para efeito de teste o *Buffer* utilizado tinha capacidade para 2000 amostras, podendo este valor ser modificado em tempo de compilação do aplicativo.

Figura 7. 2 - Fluxograma de aquisição de dados.

A taxa de atualização do gráfico foi predefinida com um período de 500 ms, podendo também ser redefinida em tempo de compilação do *software*. Esta taxa de atualização é tão precisa quanto menor for a latência entre a solicitação de uma interrupção de relógio e seu atendimento.

O intervalo de 10 us deve ser respeitado para que o conversor apresente um valor válido em seus terminais de leitura. No entanto, este intervalo será tão preciso quanto menor for a latência apresentada pela plataforma utilizada na execução do *software*.

Os três canais disponíveis na placa de aquisição podem ser amostrados e visualizados simultaneamente, sendo identificados por cores distintas. O canal 1 (BNC1) é identificado pela cor Amarela, o canal 2 (BNC2) pela cor Ciano e o canal 3 (BNC3) pela cor Magenta (Figura 7.3).

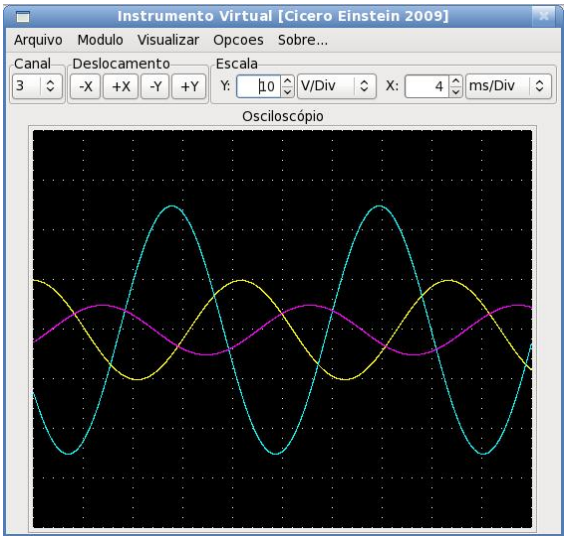

Figura 7. 3 - Exemplo de visualização dos três canais disponíveis.

8. Conclusões

A viabilidade de um sistema de medição virtual requer um conjunto de atributos de *hardware* e *software* que garantam principalmente determinismo de seus temporizadores. Dependendo da finalidade do instrumento desejado, o uso de temporizadores em alta resolução se torna peça chave para o projeto.

O conjunto de hardware utilizado requereu um sistema operacional com suporte especifico. Isto se traduziu no uso de suporte a elementos com lançamento no mercado datados de mais de 10 anos. Neste sentido o sistema operacional Linux *Slackware* foi uma escolha bem sucedida.

A configuração do *kernel* de um sistema operacional demanda o total conhecimento do hardware envolvido. Isto para que seja possível uma adequação às atribuições impostas ao sistema final. Porém, mesmo com uma reconfiguração, as características desejadas não foram obtidas.

O sistema final apresentou uma latência entre o tempo de solicitação de uma interrupção temporizada e o tempo de atendimento desta interrupção, fora dos padrões RTOS. Cuja implicação foi a inserção de insegurança na forma de onda exibida e, conseqüentemente na aferição de medidas ou informações com base na aquisição de dados realizada.

Uma possível solução para os obstáculos encontrados é um estudo aprofundado da configuração do sistema operacional escolhido para adequação ao *hardware* disponível. No caso de absoluta certeza de que o problema não reside neste ponto, a utilização de um equipamento mais recente pode, também, viabilizar o uso do PC como instrumento virtual.

Em trabalhos futuros, o aperfeiçoamento da interface deve contemplar a implementação de rotinas para salvar a imagem do gráfico, exibir medidas e atributos de cada canal amostrado, realizar operações aritméticas entre os canais, e por fim incluir novos módulos de análise tais como FFT e analise de transitórios. Os quais eram previstos inicialmente.

9. Cronograma de Atividades

As atividades compreenderão o período entre 16 de março de 2009 e 15 de julho de 2009, e serão realizadas em acordo com o cronograma a seguir.

Atividades	Período			
	16/03/09 a 15/04/09	16/04/09 a 15/05/09	16/05/09 a 15/06/09	16/06/09 a 15/07/09
Revisão Bibliográfica	X			
Implementação de Software	X	X		
Validação do Software		X		
Realização de Experimentos		X		
Análise dos Resultados			X	
Elaboração de Relatório Descritivo			X	X

Referências

[1] KATEBI, R.; JOHNSON, M. A.; WILKIE, J. **Control and Instrumentation for Wastewater Treatment Plants**. London: Springer-Verlag London Limited. 1999. p. 137;

[2] OBRENOVIC Ž.; STARCEVIC, D.; JOVANOV, E. **Virtual Instrumentation**. In: Wiley Encyclopedia of Biomedical Engineering. New York: M. Akay (Ed.). NY, USA: John Wiley & Sons Inc. 2006;

[3] RODRIGUES, C. R.. **A Instrumentação Virtual como Forma de Integração entre a Teoria e Prática no Ensino de Medidas Elétricas**. In: Congresso Brasileiro de Ensino de Engenharia, 29º; 2001, Porto Alegre. Porto Alegre: PUCRS, 2001;

[4] ARAUJO, R. B.; OLIVEIRA, L. A. G.; SOARES, R. P. O. **Real Lacos: Laboratório de Instrumentação Virtual Avançado para Ensino de Sistema de Controle**. In: World Congress on Computer Science, Engineering and Technology Education, 2006, Santos. Anais do World Congress on Computer Science, Engineering and Technology Education. Santos, 2006;

[5] BORGES, A. P. **Instrumentação Virtual Aplicada a um Laboratório com Acesso pela Internet**. 2002. Tese (Mestrado em Engenharia Elétrica) – Escola Politécnica, Universidade de São Paulo, São Paulo;

[6] BUCKMAN B. **VI-Based Introductory Electrical Engineering Laboratory Course**. In: The International Journal of Engineering Education, vol. 16, no. 3. TEMPUS Publications, 2000, pp. 212-217;

[7] CHEN, S.H., et al. **Development of Remote Laboratory Experimentation through Internet**. In: Proceedings of The IEEE Symposium on Robotics and Control. 1999. Hong Kong, 1999. p. 760-765;

[8] ZELAK, A.; LOPES, H. S. **Aplicação de Instrumentação Virtual na Análise de Sinais Biomédicos**. In: FNCTS, 1998, Curitiba, PR. Anais do 4º Fórum Nacional de Ciência e Tecnologia em Saúde, 1998. p. 249-250;

[9] Harris Semiconductor. **82C55A CMOS Programmable Peripheral Interface (Data Sheet)**. 1998;

[10] Intel ®. **82C54 CMOS Programmable Interval Timer (Data Sheet)**. 1994;

[11] Analog Devices. **12-Bit 100 kSPS A/D Converter (Data Sheet)**;

[12] Analog Devices. **Quad 12-Bit D/A Converter (Data Sheet)**.

[13] ____. **Sistema operativo**. Disponível em: http://pt.wikipedia.org/. Acesso em: 28 jul. 2009.

[14] ____. **Sistema operacional de tempo-real**. Disponível em: http://pt.wikipedia.org/. Acesso em: 09 jul. 2009;

[15] ____. **Quando Preciso de um Sistema Real-Time?** Disponível em: http://www.ni.com/. Acesso em: 11 jul. 2009;

[16] FARINES, J; FRAGA, J. S.; OLIVEIRA, R. S. **Sistemas de Tempo Real**. Florianópolis: Universidade Federal de Santa Catarina. 2000. Disponível em: http://www.das.ufsc.br/~romulo/livro-tr/. Acesso em: 15 jul. 2009;

[17] BILSKI, P.; WINIECKI, W. **Virtual Real-Time Instrumentation Using ETS Configuration**. In: XVIII IMEKO WORLD CONGRESS: Metrology for a Sustainable Development, 2006. Rio de Janeiro;

[18] CHEREPOV, M.; et al. **Hard Real-Time with Ardence RTX on Microsoft Windows XP and Windows XP Embedded**. In: Windows XP Embedded Technical Articles. Disponível em: http://msdn.microsoft.com. Acesso em: 28 jul. 2009;

[19] **IntervalZero – RTX**. Disponível em: http://www.intervalzero.com/rtx.htm. Acesso em: 29 jul. 2009;

[20] **Open Source Automation Development Lab**. Disponível em: http://www.osadl.org. Acesso em 28.jul. 2009;

[21] **RTAI - the Real-Time Application Interface for Linux from DIAPM**. Disponível em: https://www.rtai.org. Acesso em: 20 jul. 2009;

[22] **Real-Time Linux Wiki**. Disponível em: http://rt.wiki.kernel.org. Acesso em: 15 jul. 2009;

[23] LOWE, K. **Kernel Rebuild Guide**. Disponível em: http://www.digitalhermit.com/linux/Kernel-Build-HOWTO.html. Acesso em: 28 jul. 2009;

[24] ___. **Cyclictest**. Disponível em: http://rt.wiki.kernel.org/index.php/Cyclictest. Acesso em: 31 jul. 2009;

[25] **Portable User Interface**. Tecgraf - Computer Graphics Technology Group, PUC-Rio, Brasil. Disponível em: http://www.tecgraf.puc-rio.br/iup/. Acesso em: 29 jul. 2009;