Iniciação Científica - FAPESP
Relatório Científico Final

Paulo Matias

23 de dezembro de 2009

1 Introdução

Este relatório descreve os trabalhos desenvolvidos no âmbito do projeto de iniciação científica intitulado "Instrumentação eletrônica para neurobiofísica: Implementação de módulos de monitoramento e visualização de dados neurais em tempo real durante experimentos *in vivo* em pesquisas no DipteraLab", discutindo as atividades realizadas e os resultados obtidos.

O projeto tem por objetivo gerar meios para facilitar a execução de experimentos realizados com moscas, visando o estudo da codificação neural do sistema visual das mesmas. Trata-se da implementação de uma interface gráfica capaz de visualizar informações relacionadas aos dados experimentais adquiridos - tais como taxas de disparo dos neurônios, histogramas, e gráficos do tipo "raster" - à medida que os dados são coletados durante o experimento. Também será estudada a possibilidade de reconhecer em tempo real uma sequência pré-estabelecida na resposta neural da mosca, viabilizando o início de experiências com realimentação, tal como a alteração do estímulo visual em função da resposta neural.

No restante desta introdução, contextualizaremos o trabalho, a fim de justificar algumas decisões tomadas no projeto.

1.1 Características básicas da codificação neural

As respostas do neurônio H1 da mosca e de outros neurônios sensores são, em grande parte, codificadas nos intervalos de tempo entre ocorrência de spikes, principalmente no caso de estímulos dinâmicos[1]. Portanto, é importante que as medidas dos instantes de ocorrência dos spikes sejam precisas[2].

1.2 Visão geral do aparato experimental

A Figura 1 mostra uma visão geral do aparato experimental disponível. O frontend analógico amplifica, filtra e discrimina o sinal, entregando pulsos nos instantes de ocorrência de spikes em cada um dos neurônios. Esses pulsos são passados a um hardware digital simples e barato, composto de lógica TTL, que

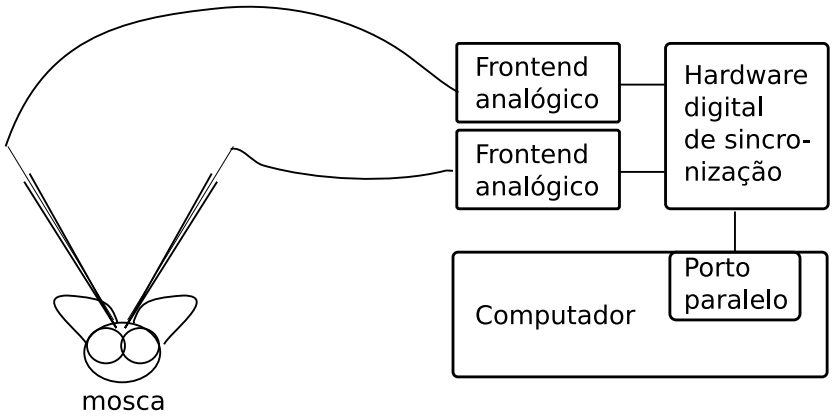


Figura 1: Visão geral do aparato experimental desenvolvido e utilizado no DipteraLab.

sincroniza pulsos de um ou mais canais e se comunica com um computador por meio do porto paralelo.

Sempre que ocorrem um ou mais pulsos, o computador é avisado por meio da interrupção do porto paralelo. Dessa forma, uma rotina de interrupção é executada pelo computador para armazenar em quais canais houve pulso naquele instante e a medida do instante de tempo em si (timestamp) de ocorrência daqueles pulsos.

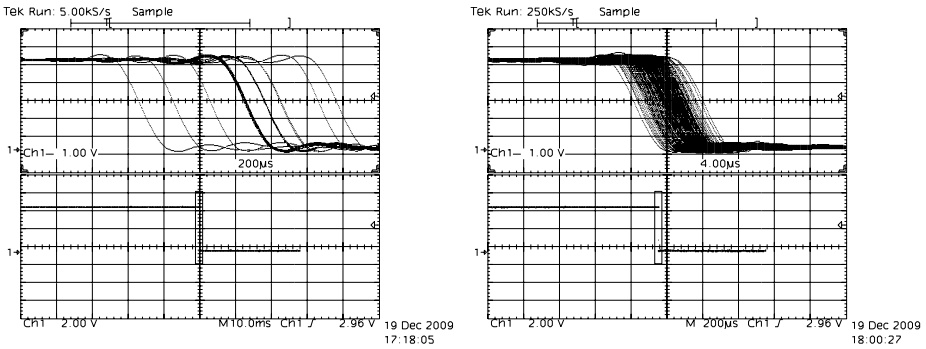
Quando a rotina de interrupção inicia, ela desce o sinal de STROBE do porto paralelo, indicando para o hardware externo que a rotina está em execução e que novos dados não podem ser recebidos ainda pelo computador. Quando a rotina termina, o sinal de STROBE é levantado novamente, para indicar o término da rotina. Se ocorrerem pulsos enquanto o sinal de STROBE estiver baixo, o hardware externo aguarda para entregá-los quando o sinal de STROBE subir novamente.

Quando é realizado experimento com apenas um neurônio, essa característica do equipamento de aquisição não acarreta em perda de precisão das medidas, pois o período de refração do neurônio é de 2 ms, e a rotina pode facilmente executar em bem menos de 2 ms. Entretanto, quando é realizado experimento com mais de um neurônio, o tempo de execução da rotina deve ser incluído no erro experimental das medidas dos timestamps, pois é possível que dois neurônios diferentes disparem em instantes de tempo muito próximos.

Por fim, uma fonte de erro experimental que é sempre importante nas medidas dos timestamps é a variância do instante de execução da rotina de interrupção (jitter).

1.3 Uso de sistemas operacionais de tempo real

Para garantir as características de baixo jitter, com latência pequena e determinística, que são essenciais para uma boa medida dos instantes de ocorrência



(a) Sinal gerado com Linux (b) Sinal gerado com Linux-RTAI

Figura 2: Comparação entre sinais de onda quadrada gerados com um sistema operacional comum (Linux) e com um sistema operacional de tempo real (Linux-RTAI).

de spikes, torna-se necessário o uso de um sistema operacional de tempo real no computador que realiza a aquisição dos dados.

Para ilustrar a diferença entre um sistema operacional comum e um sistema operacional de tempo real, apresentamos a Figura 2, que mostra medidas realizadas com um osciloscópio de um sinal de onda quadrada gerado no porto paralelo de um computador por um programa executado em Linux comum e por um programa executado no Linux-RTAI. Enquanto o sinal gerado com o Linux comum apresentou jitter da ordem de 2 ms, o sinal gerado com o Linux-RTAI apresentou jitter da ordem de apenas 8 μs.

2 Módulos de aquisição em tempo real

2.1 RTLinuxFree

Parte do projeto de iniciação científica consistia no estudo e adaptação dos módulos já existentes de aquisição em tempo real. Esses módulos eram desenvolvidos para o sistema operacional RTLinuxFree, baseado na antiga série 2.4 do kernel Linux. A distribuição usada como base foi um livecd do Knoppix com kernel RTLinux pré-instalado.

Não foi necessária a modificação do código dos módulos em si para a construção da interface gráfica executando sob o RTLinux, mas foi necessária a atualização de todo o sistema do Knoppix para a distribuição Debian *etch*, última versão do Debian a suportar a série 2.4, a fim de possibilitar a instalação das bibliotecas gráficas necessárias.

Como o processo de atualização e customização do Knoppix foi trabalhoso, desenvolvemos um CD-ROM com nossa versão modificada do sistema, para facilitar a instalação do sistema pronto para receber a interface gráfica em novas

máquinas.

Foram realizadas tentativas de utilizar uma versão mais nova do RTLinuxFree, para os primeiros kernels da série 2.6. Entretanto, essa nova versão do RTLinuxFree apresentou mais problemas com detecção de hardware que a versão antiga. Foi constatado que ela não traria vantagens, por funcionar apenas com os kernels mais antigos da série 2.6.

2.2 Xenomai

O Xenomai é outro sistema operacional de tempo real baseado em Linux que, ao contrário do RTLinuxFree, é desenvolvido ativamente até hoje. Como o sistema baseado em RTLinuxFree apresentava problemas para funcionar em máquinas mais modernas, por utilizar uma versão antiga do kernel Linux, com ausência de vários drivers, optamos por experimentar o Xenomai.

Foram desenvolvidas versões dos módulos de tempo real para o Xenomai. Entretanto, o Xenomai possui um modelo de comunicação das tarefas de tempo real com o espaço de usuário muito diferente do RTLinuxFree. Enquanto no RTLinuxFree eram utilizadas FIFOs nas quais os dados se comportavam como um fluxo contínuo de informações, o Xenomai encapsulava todas as comunicações em pacotes discretos de dados.

Essa característica do Xenomai acarretou em uma sobrecarga inaceitável no sistema de aquisição durante a transmissão dos dados de estímulo. A interface gráfica em espaço de usuário ficava praticamente travada ao tentar enviar os dados de estímulo para os módulos de tempo real.

2.3 RTAI

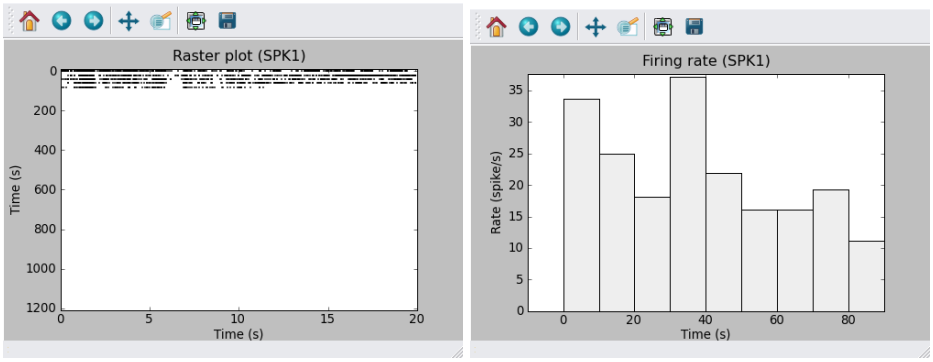
Como os testes com o Xenomai não foram satisfatórios, passamos a experimentar outro sistema operacional de tempo real baseado em Linux e desenvolvido ativamente - o RTAI.

O RTAI possui um modelo de comunicação com FIFOs praticamente idêntico ao modelo que era utilizado pelo RTLinuxFree, sendo isento do problema que havia sido observado com o uso do Xenomai.

Foram desenvolvidos módulos de tempo real para o RTAI, que se comportaram de maneira bastante satisfatória. Dessa forma, concluímos com duas versões funcionais do sistema de aquisição, uma baseada no RTLinuxFree e outra baseada no RTAI.

3 Interface gráfica

Foi desenvolvida uma interface gráfica modular, que permitisse a visualização de diversas informações úteis em tempo real durante a realização do experimento. Foi escolhida a linguagem de programação Python com as bibliotecas matplotlib e scipy, que fornecem um ambiente de programação bastante similar ao MATLAB, familiar aos integrantes do DipteraLab.



(a) Gráfico raster (b) Taxa de disparo

Figura 3: Algumas das janelas da interface gráfica sendo atualizadas ao longo de um experimento com mosca.

Foram desenvolvidos módulos para visualização de gráficos do tipo raster (Figura 3a) em paralelo com o gráfico do estímulo (de forma a verificar facilmente se a resposta do neurônio está coerente com o estímulo fornecido), para visualização de taxas de disparo (Figura 3b) e para visualização da contagem de pulsos espúrios, ou seja, pulsos que indicariam um intervalo entre-spikes menor que 2 ms para um mesmo neurônio, que não seriam possíveis devido ao tempo de refração do neurônio, podendo assim indicar erros no ajuste do frontend analógico.

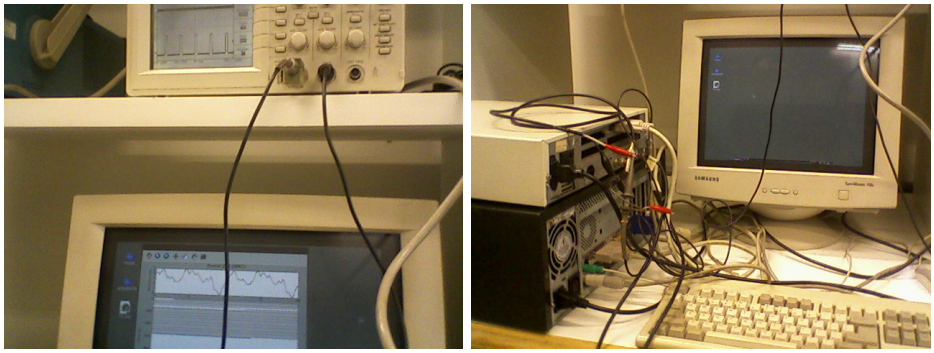
O programa foi organizado de forma a facilitar a adição de novos módulos de visualização que venham a ser desenvolvidos futuramente.

A interface gráfica disponibiliza também uma janela para a execução de comandos em Python, onde fica disponível um vetor contendo todos os dados capturados no experimento até então. Dessa forma, podem ser realizados cálculos rápidos sem precisar esperar o término do experimento. Os resultados obtidos podem ser visualizados tanto de forma numérica como na forma de gráficos renderizados pela matplotlib. Entretanto, essa janela de comandos só funciona no sistema baseado em RTAI, que utiliza uma versão recente do kernel Linux e suporta recursos de threads que não funcionavam adequadamente na versão utilizada pelos sistemas baseados em RTLinuxFree.

4 Medidas de desempenho

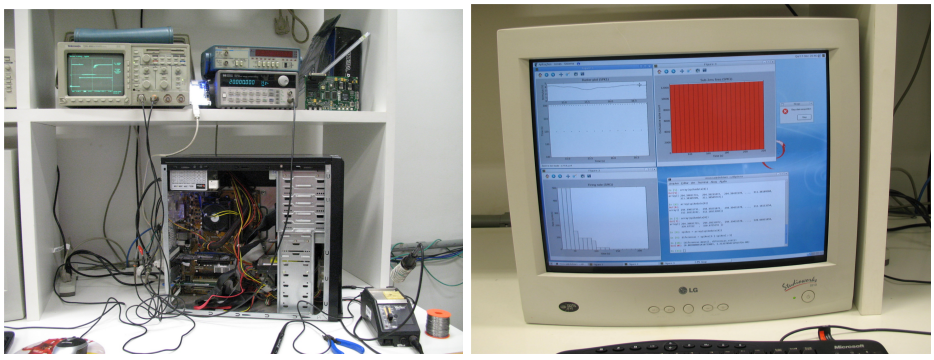
Foram realizadas diversas medidas de desempenho em bancada com os sistemas desenvolvidos. Algumas fotos dos equipamentos de bancada são apresentadas nas Figuras 4 e 5.

Por meio do monitoramento do sinal de STROBE e do sinal de interrupção do porto paralelo em um osciloscópio, é possível medir a latência de atendimento à interrupção (distância entre a borda positiva do sinal do pino de interrupção



(a) (b)

Figura 4: Testes de bancada do sistema com RTLinuxFree.



(a) (b)

Figura 5: Testes de bancada do sistema com RTAI.

e a borda negativa do sinal de STROBE), o tempo de duração da rotina de interrupção (largura da região de sinal baixo do STROBE) e o jitter total da operação (espalhamento da borda positiva do sinal de STROBE).

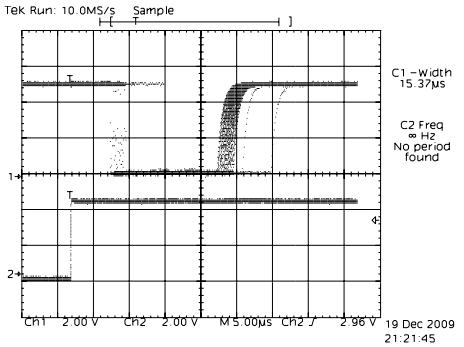
Apresentaremos a seguir medidas desses parâmetros em diversas configurações do sistema. Em todos os gráficos de sinal de osciloscópio que serão apresentados, o Canal 1 representa o sinal de STROBE e o Canal 2 representa o sinal do pino de interrupção do porto paralelo.

O pino de interrupção é alimentado com uma onda quadrada de 500Hz (pior-caso, devido ao tempo de refração dos neurônios) produzida por um gerador HP 33120A. Os pinos de dados são todos mantidos altos, em particular o primeiro, que indica que um novo dado de estímulo deve ser devolvido pelo computador, para assegurar que a rotina tome o maior tempo possível.

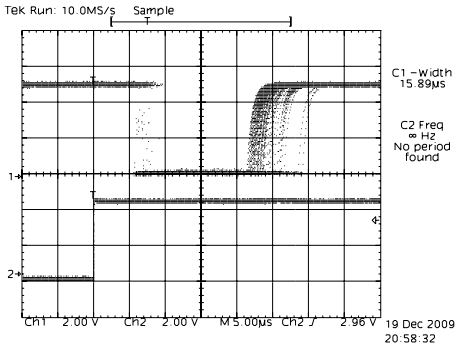
As medidas para cada configuração do sistema são repetidas para o console (versão sem interface gráfica e sem servidor X executando), para a versão com interface gráfica (GUI) em utilização normal, e para a versão com interface gráfica sofrendo imposição de carga sobre o sistema (*ping flood* na rede Ethernet e quatro janelas *glxgears*).

4.1 RTLinuxFree

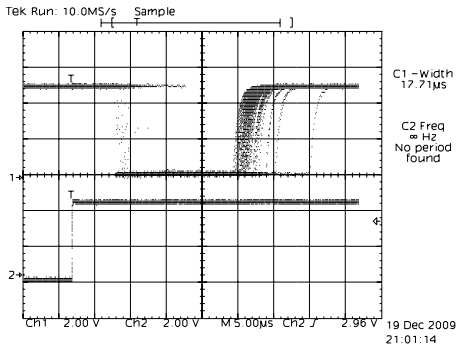
As medidas a seguir foram realizadas no sistema com RTLinuxFree em uma máquina Pentium III Coppermine com 512MB de RAM.



Console	
Latência	6μs
Duração	15μs
Jitter	8μs



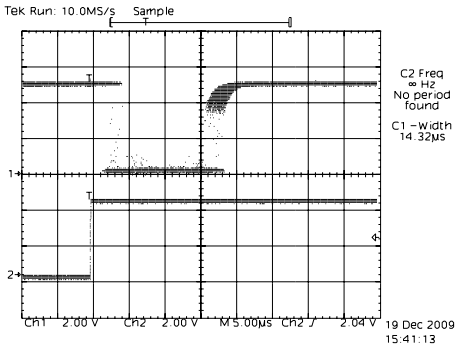
GUI	
Latência	7μs
Duração	16μs
Jitter	8μs



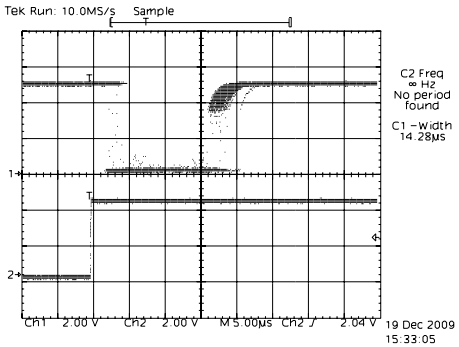
GUI + carga	
Latência	7μs
Duração	18μs
Jitter	11μs

4.2 RTAI 32 bits

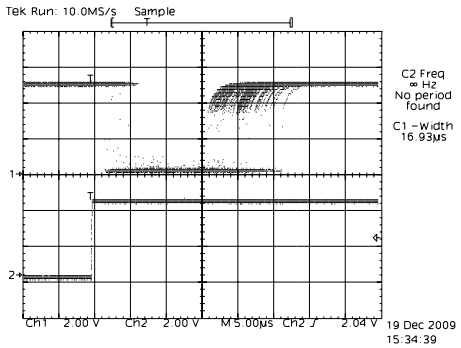
As medidas a seguir foram realizadas no sistema com RTAI em uma máquina Athlon 64 X2 com 2GB de RAM, com sistema operacional de 32 bits.



Console	
Latência	3μs
Duração	14μs
Jitter	3μs



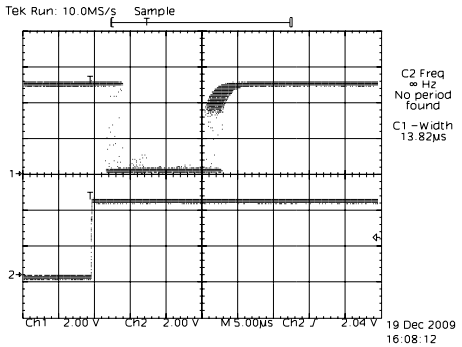
GUI	
Latência	3μs
Duração	14μs
Jitter	5μs



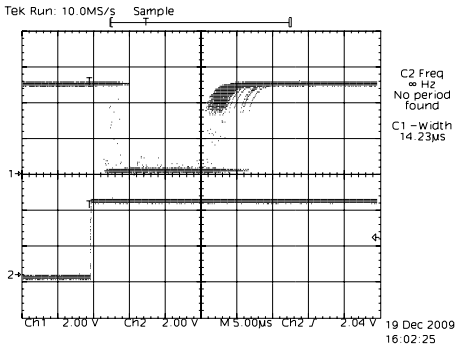
GUI + carga	
Latência	3μs
Duração	17μs
Jitter	10μs

4.3 RTAI 64 bits

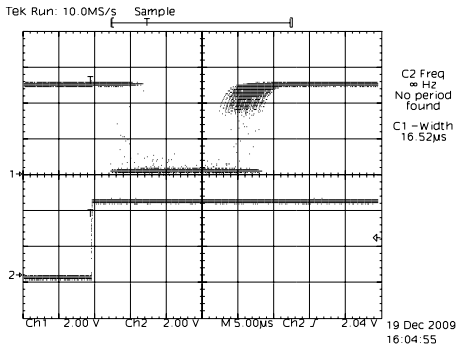
As medidas a seguir foram realizadas no sistema com RTAI em uma máquina Athlon 64 X2 com 2GB de RAM, com sistema operacional de 64 bits.



Console	
Latência	3μs
Duração	14μs
Jitter	3μs



GUI	
Latência	3μs
Duração	14μs
Jitter	6μs



GUI + carga	
Latência	3μs
Duração	17μs
Jitter	7μs

4.4 Discussão

Houve pouco ou nenhum aumento dos valores dos parâmetros medidos após a adição da interface gráfica, indicando que o acréscimo da mesma não afetará significativamente a precisão das medidas.

Mesmo impondo uma grande quantidade de carga sobre o sistema, o erro experimental das medidas de timestamp ainda seria aceitável, permanecendo sempre abaixo dos 50μs mesmo para experimentos com mais de um neurônio.

Foi constatado também que o uso do RTAI é vantajoso com relação ao uso do RTLinuxFree não somente no quesito de atualização e compatibilidade do sistema, mas também no quesito de precisão das medidas de timestamp efetuadas.

4.5 Testes com mosca

Os testes com mosca e dois sistemas de aquisição em paralelo, que haviam sido propostos no plano de trabalho, não puderam ser realizados pois, devido à quantidade incomum de chuvas na cidade de São Carlos durante os últimos meses, não foi possível capturar moscas.

Entretanto, no período em que moscas ainda podiam ser capturadas, diversas aquisições foram realizadas por integrantes do DipteraLab usando uma versão preliminar do sistema com RTLinuxFree e interface gráfica, produzindo dados experimentais consistentes.

Acima de tudo, os testes de bancada asseguram que o sistema funciona corretamente com pulsos de pior-caso gerados artificialmente, que causam maior carga de processamento que os sinais reais dos neurônios da mosca.

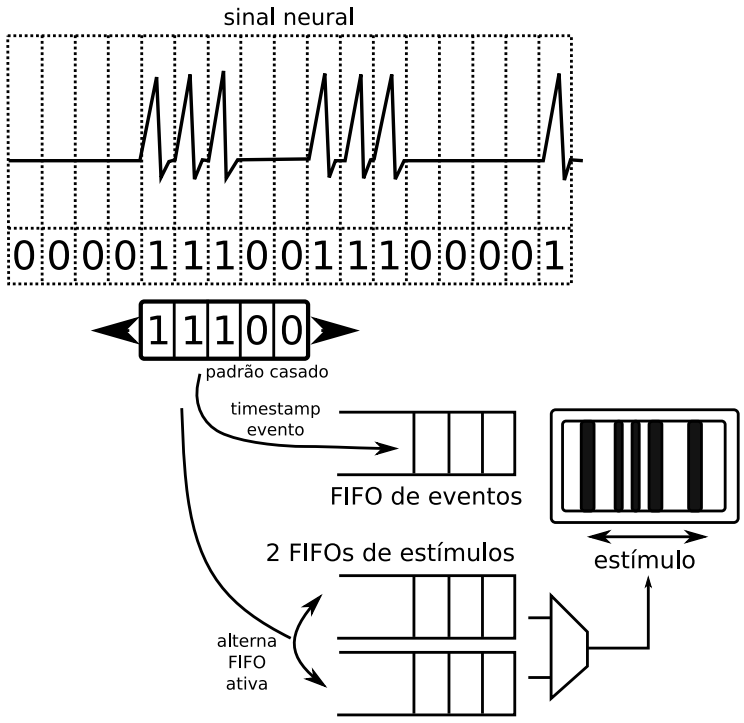


Figura 6: Arquitetura sugerida para o módulo de tempo real de um sistema com comparador de trem de spikes.

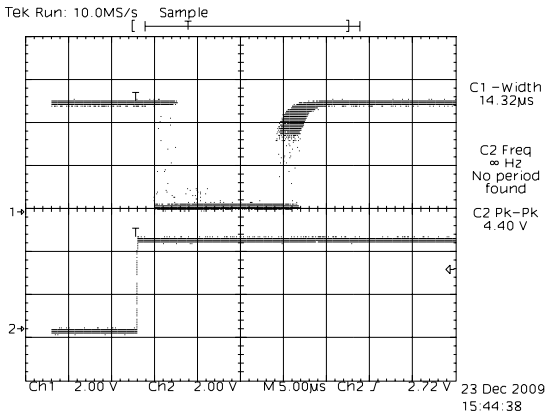
5 Comparação de trens de spikes

Esta seção pretende analisar a viabilidade de implementação de um comparador de trens de spikes, que seria utilizado em novos experimentos com realimentação.

A arquitetura inicialmente sugerida para o módulo de tempo real a ser utilizado em um sistema com suporte a esse tipo de experimento é apresentada na Figura 6.

Um padrão a ser casado com um trecho do sinal neural, fornecido no formato de bins de 2 ms ou de outra largura a ser especificada pelo usuário, é verificado em tempo real durante a aquisição. Ao ser encontrado, o estímulo é trocado e um evento é informado ao espaço de usuário por meio de uma FIFO de eventos. São utilizadas duas FIFOs de estímulos para permitir que o estímulo seja trocado quase que imediatamente, em um tempo inferior ao próximo pedido de dados de estímulo pelo hardware externo.

Essa arquitetura foi implementada em um módulo de tempo real para o RTAI, juntamente com um pequeno script em Python em espaço de usuário para testes. Programando o módulo para verificar uma sequência de 500 bins, cada um de 2 ms, obteve-se o seguinte gráfico monitorando o sinal de STROBE em um osciloscópio, com o sistema executando em console:



Os parâmetros de latência, duração e jitter medidos são os mesmos, dentro da faixa observável na figura, que para o módulo sem suporte a realimentação. Concluímos assim que as modificações não interferem significativamente na precisão de medida dos timestamps.

Dessa forma, a implementação dos experimentos com realimentação parece ser viável. Como continuidade deste trabalho, são necessários mais testes do módulo desenvolvido para validá-lo e a implementação de mais recursos, como suporte a casar mais de uma sequência de spikes diferente.

6 Conclusão

Foi desenvolvido um novo sistema de aquisição para experimentos em pesquisas no DipteraLab, com suporte a dois sistemas operacionais de tempo real distintos baseados em Linux. O novo sistema foi validado experimentalmente, indicando seu correto funcionamento e uma precisão aceitável. O trabalho desenvolvido tem sido e continuará sendo de grande utilidade para o DipteraLab.

O desenvolvimento de um sistema com suporte a realimentação foi considerado viável. Maiores testes do módulo desenvolvido durante a avaliação de viabilidade, assim como melhorias do mesmo e sua integração à interface gráfica ficam como sugestões de trabalhos futuros.

Outra sugestão para trabalhos futuros é a implementação da parte digital do hardware externo em lógica programável (FPGAs ou CPLDs), e sua melhoria para realizar a medida dos instantes de ocorrência de spikes (timestamps) em hardware, possibilitando uma precisão experimental ainda maior, que passaria a ser limitada pelo jitter do frontend analógico.

O comparador de trens de spike também é adequado para implementação em hardware, por possuir o comportamento de um *shift register*. Dessa forma, o número de bins e de padrões a serem casados passaria a ser limitado pelo número de elementos lógicos do circuito de lógica programável utilizado.

Estudos a respeito da implementação em hardware de mais partes do sistema de aquisição já foram iniciados.

Referências

[1] Rob R. de Ruyter van Steveninck, Geoffrey D. Lewen, Steven P. Strong, Roland Koberle, and William Bialek. Reproducibility and Variability in Neural Spike Trains. *Science*, 275(5307):1805–1808, 1997.

[2] I. Nemenman, G. D. Lewen, W. Bialek, and R. R. de Ruyter van Steveninck. Neural coding of a natural stimulus ensemble: Uncovering information at sub-millisecond resolution. *eprint arXiv:q-bio/0612050*, December 2006.