

myo_cap_gui – interface gráfica para digitalização de sinal miográfico

Palavras-chave

miografia, conversão A/D, neurociência

Área de conhecimento

Ciências exatas e da terra - Ciência da Computação - Sistemas de Computação – Processamento Gráfico (Ggraphics)

Link do grupo de pesquisa no CNPq

<http://dgp.cnpq.br/dgp/espelhogrupo/8003760460683702>

Linha de Pesquisa

Visão computacional

Introdução/Justificativa

Inclua na justificativa os benefícios esperados no processo ensino-aprendizagem dos alunos de graduação e/ou pós-graduação vinculados ao projeto. Explicite também o retorno para os cursos de graduação e/ou pós-graduação e para os professores da UFS em geral.

Objetivos

O projeto tem como objetivo criar um sistema de software, mais especificamente uma interface gráfica, e firmware que permita que um computador se comunique com a placa TIVA, exiba na tela e salve em arquivo os sinais capturados pela placa. A Figura 1 mostra como aproximadamente deve ser a tela principal da interface gráfica. O sistema deve ser escrito em inglês, já que o trabalho deve ser publicado em veículo internacional no futuro.

A placa TIVA possui 12 conversores A/D, portanto, pode capturar 12 sinais a cada leitura. Será conectada a placas feitas sob medida para capturar sinais miográficos. Cada placa poderá possuir até 12 canais e os sinais podem ser multiplexados caso haja necessidade.

A interface gráfica permitirá a escolha de parâmetros de captura e de exibição. Os parâmetros de captura devem ser transmitidos à placa TIVA por meio de protocolo a ser definido. Os parâmetros de exibição afetam apenas a visualização do sinal na tela do computador.

A série temporal é impressa na saída padrão. O programa deve ser chamado através de um script bash que redireciona a saída para o arquivo especificado pelo primeiro parâmetro.

É interessante que o programa salve a série temporal em um vetor na memória, pois no futuro pode ser necessário implementar outras funcionalidades mais flexíveis de visualização, exibição e edição do sinal. Para acelerar a alocação e a execução, o vetor deve crescer em blocos ao invés de uma posição por vez. O programa deve ter controle do

número de posições usadas do vetor.

A interface gráfica não conhece a configuração do hardware. Deve ser genérica de modo que funcione com qualquer configuração. O usuário, por sua vez, deve ser capaz de indicar a configuração correta do hardware, por exemplo, número de canais a serem lidos e número de placas a serem multiplexadas. A placa TIVA recebe a solicitação e retorna erro ou ok conforme sua capacidade.

Futuramente pode-se definir um conjunto de funções para consultar a placa quanto a suas capacidades e configuração, por exemplo, um conjunto de funções tiva.getMaxY e tiva.getMinY onde Y é um certo parâmetro de captura.

Arquitetura

O programa será escrito em python 2.7 para que seja compatível com as bibliotecas de captura de movimento do LeapMotion. Deve haver ao menos 3 arquivos python:

tiva.py: contém as funções de comunicação com a placa tiva.

win_main.py: contém a implementação da janela principal.

win_settings.py: implementa a caixa de diálogo onde o usuário define os parâmetros de captura.

Parâmetros de captura

Os parâmetros de captura são enviados à placa TIVA e são salvos no arquivo de log. São definidos no início da captura e, para serem modificados, a captura deve ser interrompida.

As funções definidas abaixo retornam um par (0, "") em caso de sucesso. Retornam um inteiro diferente de zero e uma mensagem em caso de erro. A mensagem é enviada pelo firmware da placa TIVA, permitindo ao usuário saber o que a placa suporta, e corrigindo os parâmetros de acordo.

Os inteiros representam códigos de erro previamente estabelecidos. A mensagem deve ser informativa, explicitando o que saiu errado. Por exemplo: caso um parâmetro esteja fora dos valores suportados, indicar na mensagem qual é o intervalo como em:

```
tiva.setBps: Error: BPS value = 16 outside supported interval  
[8..12].
```

Bits per sample (k in bps, integer): indica o número de bits por amostra. O intervalo dinâmico é função de k, dado por 2^k . Por exemplo, 8 bps correspondem a um intervalo dinâmico de 256 valores. Deve haver uma função tiva.setBps(k) que envia o valor de bps desejado. Note que mesmo capturando em 12 bits, mandará apenas os 8 mais significativos se esse for o valor de k. O firmware retornará erro caso k seja menor que 8 ou maior que 12.

Sample rate (r in Hz): indica a taxa de amostragem em amostras por segundo. Deve haver uma função tiva.setSampleRate(rate) que envia o valor de bps desejado. O firmware

retornará erro caso r esteja fora do intervalo permitido.

Channels per board (c, integer): indica o número de canais a serem lidos por placa. A placa não é capaz de detectar quantos canais estão disponíveis, portanto, serão lidos os sinais dos c primeiros conversores A/D da TIVA. Deve haver uma função tiva.setNChannels(c) que define quantos canais devem ser lidos. O firmware retornará erro caso c seja menor que 1 ou maior que 12.

Number of boards (n, integer): indica o número de placas a TIVA deve multiplexar. O valor padrão é 1. Deve haver uma função tiva.setNBoards(n) que define quantos canais devem ser lidos. O firmware retornará erro caso n seja menor que 1 ou maior que 8, já que usaremos 3 vias de seleção. Não precisa necessariamente ser potência de 2. Podemos multiplexar 3 placas, por exemplo.

Note que, caso o hardware mude no futuro, ou caso usemos mais de um hardware alternativo, o firmware deve ser adaptado e não a interface gráfica. A interface deve ser o mais genérica possível.

Parâmetros de exibição

Os parâmetros de exibição não são enviados à placa TIVA. Podem ser modificados a qualquer momento e não interferem na captura dos sinais.

Swipe (t in seconds, float. w in samples, integer): indica o tempo/número de amostras em uma varredura horizontal da tela. O valor padrão é 1s. É dado por duas caixas de texto na tela: uma com o tempo em segundos (t) e outra com o número de amostras (w). Obviamente, w = t*r. O valor de um muda automaticamente com a mudança do outro.

Zero (z, integer): indica a posição do eixo horizontal no intervalo dinâmico. O eixo horizontal aparece como uma linha sólida no grid. Por padrão é L/2, onde L = 2^k e k é o número de bits per sample. Deve pertencer ao intervalo [0..L-1].

Amplitude (A in Volts, float): indica a quanto corresponde o intervalo dinâmico L. O valor padrão é 3.3V.

Vertical Tick (vTick in Volts, float): intervalo entre as marcações pontilhadas do grid. O valor padrão é 1.0V

Horizontal Tick (hTick in seconds, float): intervalo entre as marcações pontilhadas do grid.

Channels (z, integer): permite a exibição de mais canais na tela. O número padrão é 4. Caso não caibam todos os canais sendo capturados, deve haver um scroll bar.

Funcionalidades

Start Capture: inicia a captura, imprimindo na saída padrão os parâmetros de captura e em seguida a série temporal capturada. Desabilita a alteração dos parâmetros de captura.

Stop Capture: para a captura e a impressão da série temporal. Habilita a alteração dos parâmetros de captura.

Menu File

Load: carrega arquivo com os parâmetros de captura e exibição.

Save: salva arquivo com os parâmetros de captura e exibição.

Capture Settings: abre diálogo para edição dos parâmetros de captura.

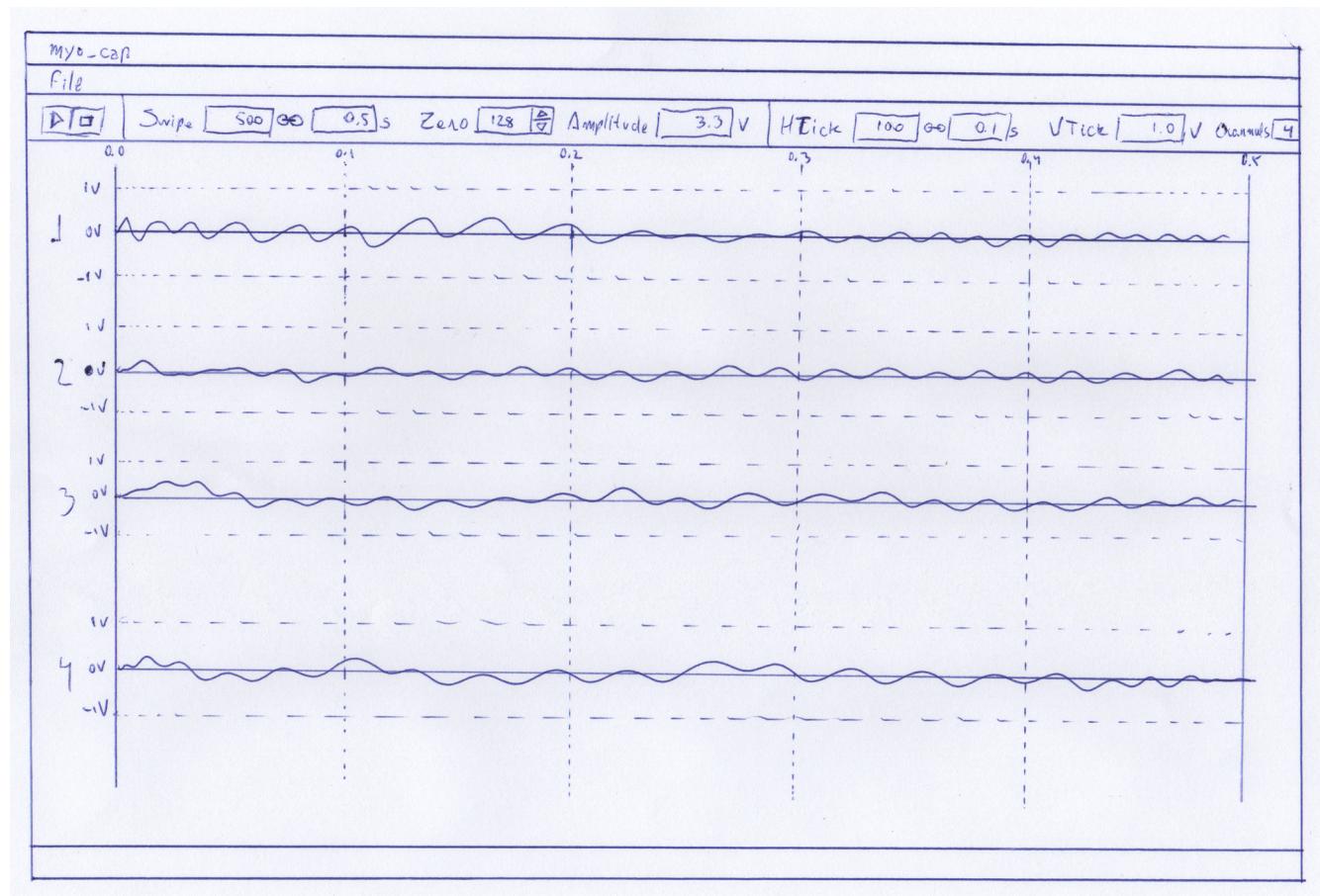


Figura 1: aspecto da interface gráfica.