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 (Grraphics)

Link do grupo de pesquisa no CNPq

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

Linha de Pesquisa

Visão computacional

1. 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.

Através de dispositivos eletrônicos e eletrodos, é possível captar sinais provenientes de nervos motores. Essa técnica é conhecida como miografia. Quando são usados eletrodos sobre a pele, é conhecida como miografia de superfície.

Pelo fato de haver músculo e pele entre o eletrodo e os neurônios de interesse, os sinais obtidos através da miografia de superfície contêm uma grande quantidade de ruído. Através de filtros, é possível eliminar parte do ruído. Técnicas de aprendizado computacional podem ser usadas para classificar os sinais obtidos, vinculando-os a movimentos pré-estabelecidos. Uma vez treinado o classificador, o sinal poderá ser usado para controlar próteses mecânicas.

2. Interface myo_cap

2.1 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 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 defnido. Os parâmetros de exibição afetam apenas a visualização do sinal na tela do computador.

O programa salva a série temporal em um vetor na memória. O mesmo vetor é usado quando a série é lida de um arquivo. Ao selecionar **Show capture**, o programa deve exibir os dados capturados independente se foram lidos do arquivo ou direto da placa.

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 cojunto de funções tiva.getMaxY e tiva.getMinY onde Y é um certo parâmetro de captura.

2.2. 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 usário define os parâmetros de captura.

2.3. Parâmetros de captura

Os parâmetros de captura são enviados à placa TIVA e são salvos no arquivo CSV. 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 L 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 falor 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.

2.4. 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.

Minimum Voltage (voltMin, float): indica a voltagem correspondente à intensidade 0.

Maximum Voltage (voltMax, float): indica a voltagem correspondente à intensidade L, onde

L é o tamanho do intervalo dinâmico, dado por 2^k, e k é o número de bits por amostra.

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

Horizontal Tick (horizTick in seconds, float): intervalo entre as linhas verticais 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.

2.5. Menus

Menu File

Load capture: carrega arquivo de captura CSV previamente salvo.
Save capture: salva dados capturados em arquivo CSV.

Menu capture

Start capture: inicia a captura. Desabilita a alteração dos parâmetros de captura.
Stop capture: para a captura. Habilita a alteração dos parâmetros de captura.
Show capture: exibe os dados previamente capturados.

Menu settings

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

Capture settings: abre diálogo para edição dos parâmetros de captura: Bits per sample, Sample rate, Channels per board, Number of boards.
Display settings: abre diálogo para edição dos parâmetros de exibição: Swipe, Zero, Amplitude, Vertical tick, Horizontal tick e Channels.
Communication settings: abre diálogo para edição dos parâmetros de comunicação com a placa, ou seja, número da porta serial, tamanho do pacote de dados e modo packed/unpacked.

2.6. Parâmetros de comunicação

Os parâmetros de comunicação estabelecem como será a comunicação com a porta serial.

Número da porta serial

Baud rate: o baud rate padrão é característica específica da placa. A classe que implementa os parâmetros específicos da placa deve conter um baud rate funcional para a comunicação inicial e possível seleção de novo baud rate.

Tamanho do pacote de dados: estabelece o tamanho do pacote que transporta os dados de captura da placa para a porta serial.

Modo packed/unpacked: seleciona o modo de comunicação dos dados. Deve haver flags constantes na classe dos parâmetros especificos da placa, MODE_PACKED_AVAILABLE e MODE_UNPACKED_AVAILABLE com valores diferentes de zero caso os modos respectivos estejam disponíveis. O modo packed é selecionável caso MODE_PACKED_AVAILABLE seja diferente de zero.

2.7. Arquivo de saída CSV

Armazena os dados da captura, uma amostra por linha e colunas separadas por ponto e vírgula. Nas primeiras linhas armazena metadados com as configurações da captura, tanto de EMG quanto dos gestos. As linhas com metadados começam com um único sinal de #.

Linhas com comentários começam com dois sinais de # e são ignoradas.

Abaixo está um exemplo de arquivo de saída.

```
## File generated by myo_cap software
## Available from github.com/ddantas/myo_cap
## Timestamp: 2018-05-23_20-57-32
##
## EMG capture settings
##
# sampleRate: 2000
# channelsPerBoard: 4
# nBoards: 1
# bitsPerSample: 12
##
## Data
t; ch0; ch1; ch2; ch3
0.0000; 2048; 2048; 2048; 2048
0.0005; 2047; 2048; 2049; 2048
0.0010; 2047; 2048; 2049; 2048
0.0015; 2048; 2047; 2048; 2049
...
```

2.8. Arquivo de configurações

Usa o mesmo formato dos metadados do arquivo de saída. Armazena as configurações de exibição e emulação de EMG.

```
## File generated by leap_cap software
## Available from github.com/ddantas/leap_cap
## Timestamp: 2018-05-23_20-57-32
##
```

```
## EMG capture settings
##
# sampleRate: 2000
# channelsPerBoard: 4
# nBoards: 1
# bitsPerSample: 12
##
## EMG display settings
##
# swipeSamples: 2000
# vMin: -3.3
# vMax: 3.3
# vertTick: 1.0
# horizTick: 200
# showChannels: 4
```

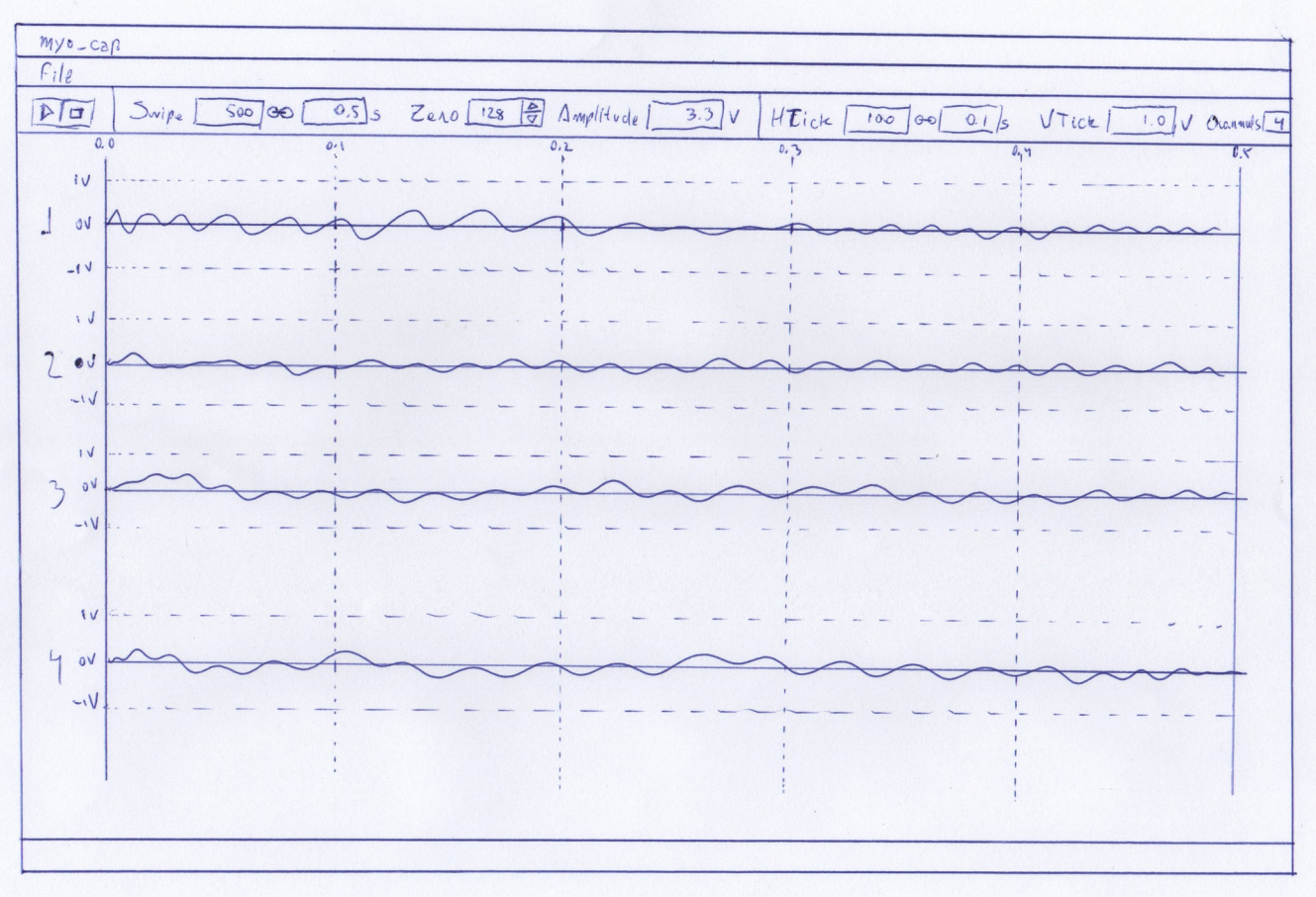

Figura 1: aspecto da interface gráfica.

3. Interface leap_cap

3.1. Objetivos

Este projeto tem como objetivo criar um sistema de software, mais especificamente uma interface gráfica, que permita a captura simultânea de sinais de eletromiografia (EMG) e gestos feitos pela mão de um usuário do sistema.

A captura de sinais de EMG será feita através de um dispositivo de captura especializado. O dispositivo é composto por uma TivaWare, responsável pela conversão A/D, placas de captura, multiplexação, cabos e eletrodos, dispostos sobre a superfície do braço do usuário. Alternativamente, o sistema pode usar como entrada um arquivo CSV com os sinais de EMG, permitindo o teste do sistema sem a presença do dispositivo.

A captura de gestos poderá ser feita de duas maneiras diferentes. A mais simples será através de um teclado, capaz de indicar quais dedos estão pressionando uma tecla. Outra maneira mais sofisticada será através do LeapMotion, um dispositivo capaz de capturar com boa precisão o quanto está flexionado cada um dos dedos da mão do usuário.

O dispositivo e respectivo software de captura estão em desenvolvimento e disponíveis no github. Já existem protótipos com um e quatro canais funcionais. Existem também um protótipo do software capaz de capturar e salvar em arquivo os sinais de EMG. As especificações do hardware e código fonte do software estão no link abaixo:

https://github.com/ddantas/myo_cap

O aluno responsável por esse projeto deve, na medida do possível, aproveitar o código fonte já disponível, evitando refazer o que é responsabilidade de outros alunos, mais especificamente, código relacionado à captura e exibição de sinais de EMG: escolha de parâmetros de captura de sinais de EMG, comunicação com o dispositivo de EMG, escolha de parâmetros de exibição de sinais de EMG.

Deve portanto focar em criar as funções de captura de gestos pelo teclado e LeapMotion, e na implementação da interface gráfica que exibirá os gestos a serem realizados pelo usuário.

3.2. 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:

leap.py: contém as funções de comunicação com o LeapMotion.

win_main.py: contém a implementação da janela principal.

win_gesture_settings.py: implementa a caixa de diálogo onde o usário define os parâmetros de captura.

3.3. Parâmetros de captura de sinais de EMG

Tanto a interface gráfica para definir os parâmetros de captura quanto as funções que enviam os parâmetros para a placa devem ser aproveitadas do sistema de captura de EMG.

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

Para mais detalhes, ver Seção **2.3**.

3.4. Parâmetros de exibição de sinais de EMG

A interface gráfica para exibir e definir os parâmetros de exibição de sinais de EMG devem ser aproveitadas do sistema de captura de EMG.

Para mais detalhes, ver Seção **2.4**.

3.5. Parâmetros de captura de gestos

Os parâmetros de captura de gestos são salvos, juntamente com os parâmetros de captura de sinal de EMG, no arquivo CSV.

Device: LeapMotion ou Keyboard.

Routine: nome de arquivo contendo a rotina de captura, ou seja, a sequência de gestos. O formato é definido na Seção **3.7. Rotina de captura**.

Hand: especificação da mão a ser capturada. Valores possíveis: {L, R} para esquerda ou direita.

3.6. Menus

Menu File

Load capture: carrega arquivo de captura CSV previamente salvo.
Save capture: salva dados capturados em arquivo CSV.
Load EMG signal: carrega dados de EMG para emular o uso do dispositivo de captura de sinais de EMG, permitindo testar o sistema mesmo sem o dispositivo.

Menu Capture

Start capture: inicia a captura. Desabilita a alteração dos parâmetros de captura.
Stop capture: interrompe a captura. Habilita a alteração dos parâmetros de captura.
Show capture: exibe dados de gestos e EMG previamente capturados.

Menu Settings

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

EMG capture settings: abre diálogo para edição dos parâmetros de captura de EMG: Bits per sample, Sample rate, Channels per board, Number of boards.
EMG display settings: abre o diálogo para edição dos parâmetros de exibição do sinal de EMG: Swipe, Zero, Amplitude, Vertical tick, Horizontal tick e Channels.
EMG communication settings: abre diálogo para edição dos parâmetros de comunicação com a placa, ou seja, número da porta serial, tamanho do pacote de dados e modo packed/unpacked.
EMG emulation: Menu com *checkbox*. Caso habilitado, é necessário carregar um arquivo de sinal de EMG. Exibe diálogo com mensagem: “Please load EMG signal by acessing menu File > Load EMG signal”
Gesture capture settings: abre diálogo para edição dos parâmetros de captura de gestos: Device, Routine, Hand side.

3.7. Rotina de captura

A rotina de captura especifica quais gestos serão capturados e por quanto tempo. É um arquivo CSV com duas colunas: a primeira indica o gesto a ser feito e a segunda o tempo em segundos. Linhas em branco ou iniciadas com o caractere # são ignoradas.

Os gestos são compostos por duas palavras unidas por *underscore* (_). A primeira indica os dedos a serem movimentados e a segunda indica o movimento.

A primeira palavra pode ser uma das seis a seguir: {hand} ou {1th, 2in, 3md, 4an, 5mn}, correspondentes aos cinco dedos, polegar, indicador, médio, anular e mínimo.

A segunda palavra pode ser uma das quatro a seguir: {open, close} ou {flex, curl, flex_curl}, correspondentes a todos os dedos esticados, todos os dedos flexionados, primeira articulação flexionada, segunda e terceira articulações flexionadas e três articulações flexionadas.

Todas as combinações possíveis são dadas pela união dos produtos cartesianos entre os conjuntos {hand} X {open, close} e {1th, 2in, 3md, 4an, 5mn} X {flex, curl, flex_curl}, totalizando 17 gestos.

Abaixo está um exemplo de arquivo de rotina de captura.

```
hand_open;2
hand_close;2
hand_open;2
hand_close;2

## 1 - thumb
```

```
1th_flex;2
hand_open;2
1th_flex;2
hand_open;2
1th_flex_curl;2
hand_open;2
1th_flex_curl;2
hand_open;2
```

3.8. Arquivo de saída CSV

Armazena os dados da captura, uma amostra por linha e colunas separadas por ponto e vírgula. Nas primeiras linhas armazena metadados com as configurações da captura, tanto de EMG quanto dos gestos. As linhas com metadados começam com um único sinal de #.

Linhas com comentários começam com dois sinais de # e são ignoradas.

Abaixo está um exemplo de arquivo de saída.

```
## File generated by leap_cap software
## Available from github.com/ddantas/leap_cap
## Timestamp: 2018-05-23_20-57-32
##
## EMG capture settings
##
# sampleRate: 2000
# channelsPerBoard: 4
# nBoards: 1
# bitsPerSample: 12
##
## Gesture capture settings
##
# device: keyboard
# routine: default.csv
# hand: right
##
## Data
th_flex; th_curl; in_flex; in_curl; md_flex, md_curl; an_flex; an_curl; mn_flex; mn_curl; ch0; ch1; ch2; ch3
0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 2048; 2048; 2048; 2048
0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 2048; 2048; 2048; 2048
0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 2048; 2048; 2048; 2048
100; 100; 0; 0; 0; 0; 0; 0; 0; 0; 2048; 2048; 2048; 2048
...
```

3.9. Arquivo de configurações

Usa o mesmo formato dos metadados do arquivo de saída. Armazena as configurações de exibição e emulação de EMG.

```
## File generated by leap_cap software
## Available from github.com/ddantas/leap_cap
## Timestamp: 2018-05-23_20-57-32
##
## EMG capture settings
##
# sampleRate: 2000
# channelsPerBoard: 4
# nBoards: 1
# bitsPerSample: 12
##
## EMG display settings
##
# swipeSamples: 2000
# vMin: -3.3
# vMax: 3.3
# vertTick: 1.0
# horizTick: 200
# showChannels: 4
##
## EMG emulation settings
##
# emulationFlag: true
# emulationData: 2018-05-04_14-39-15.csv
##
## EMG communication settings
##
# comPort: 3
# packetSize: 500
##
## Gesture capture settings
##
# device: keyboard
# routine: default.csv
# hand: right
```

4. Classe `textfile`

A classe textfile será usada para salvar e carregar os arquivos de configurações e de captura. Permite que várias *threads* salvem dados em um único arquivo. Sua interface é dada pelas funções a seguir.

`message_save`: adiciona mensagem com cada linha precedida de "## "

```
message_save(string msg)
```

`metadata_*_save`: adiciona metadado com linha precedida de "# "

```
metadata_text_save(string msg, string value)
metadata_number_save(string msg, double value, string format)
```

`metadata_load`: lê valor de metadado e retorna em formato de string.

```
metadata_load(string msg, string value)
```

`init`: inicializa log, definindo o número de colunas, formato de impressão, e nomes das colunas. Retorna um índice, começando por 1, do grupo de colunas. Pode ser chamando mais de uma vez, caso em que várias threads adicionam colunas diferentes simultaneamente.

```
format = "%f; %f; %f; %f;"
name_cols = "ch0; ch1; ch2; ch3;"
id = init(int num_cols, string format, string name_cols)
```

Caso estejamos usando duas placas de 4 canais:

```
format = "%f;%f;%f;%f;%f;%f;%f;%f"
name_cols = "ch0; ch1; ch2; ch3;ch4; ch5; ch6; ch7"
```

`log`: recebe vetor de double e grava na memória na linha correspondente do log.

```
log(int id, double values[])
```

`save`: grava log em arquivo.

```
save(filename)
```

5. Gerador de função

O gerador de função, a ser implementado na TIVA, tem como objetivo testar o sistema sem a necessidade do uso da placa de aquisição e dos eletrodos. A partir do início da captura, um *timer* é iniciado e o valor da função é calculado em função do *timestamp*, e não do índice da amostra. Isso permitirá detectar se pacotes de dados estão sendo perdidos.

A amplitude pico a pico deve ser sempre 4096, ou seja 2 elevado ao número de bits do ADC. Clicar em um item do menu adiciona um "check" e envia a função selecionada para a TIVA. Clicar novamente remove o "check" e envia uma mensagem para a Tiva selecionando o ADC.

Na TIVA, uma versão de alto nível do gerador de função deve receber o *timestamp*, período e forma de onda. A depender da forma de onda, outra função deve ser chamada. Caso o usuário tenha escolhido receber menos bits que os retornados pelo ADC, tanto o valor retornado pelo ADC quanto pelo gerador de função deve sofrer um *shift right* de tamanho adequado. A operação deve ser implementada de modo que tanto o ADC quanto o gerador de função usem o mesmo código.

5.1. Menus

Menu Function Generator

Settings: abre uma caixa de diálogo para definir a frequência da onda e tempo em segundos do stress test.

Square wave (checkable): seleciona modo *square wave*.
Sine wave (checkable): seleciona modo *sine wave*.
Sawtooth wave (checkable): seleciona modo *sawtooth wave*.

Start stress test: inicia *stress test.* O teste consiste em fazer o streamming de amostras pelo tempo selecionado em *settings.* Durante o teste, exibir uma *progress bar.* Ao final, uma caixa de diálogo com uma mensagem como a do exemplo abaixo deve ser exibida:

Test length:	30s
Capture frequency:	2000Hz
Expected samples:	60000
Received samples:	55320
Dropped samples:	4680
Drop rate:	7.80%

6. Protocolo de comunicação

O protocolo de comunicação entre a TIVA e a interface Python se divide em dois tipos de mensagens: requisições para a TIVA e mensagens de resultado.

6.1. Requisições para a TIVA

As mensagens de requisições possuem seis bytes de comprimento. Os dois primeiros bytes representam uma *instrução* e os quatro últimos um *operando.*

Instrução	Operando	Comentário
`ai`	`N/A`	Start streamming
`as`	`N/A`	Stop streamming
`sr`	`uint32`	Set sample rate in Hz
`sc`	`uint32`	Set number of channels per board
`sb`	`uint32`	Set number of boards
`ss`	`uint32`	Set number of bits per sample
`sp`	`uint32`	Set packet size in bytes
`sm`	`uint32`	Set mode packed or unpacked
`sf`	`float32`	Set function generator wave frequency
`gr`	`N/A`	Get sample rate in Hz
`gc`	`N/A`	Get number of channels per board
`gb`	`N/A`	Get number of boards
`gs`	`N/A`	Get number of bits per sample
`gp`	`N/A`	Get packet size in bytes
`gm`	`N/A`	Get mode packed or unpacked
`gf`	`N/A`	Get function generator wave frequency
`fa`	`N/A`	Select ADC mode
`fq`	`N/A`	Select square wave function mode
`fn`	`N/A`	Select sine wave function mode
`fw`	`N/A`	Select sawtooth wave function mode

O parâmetro `p` = p*acket size* define o tamanho em bytes dos pacotes retornados durante uma captura de EMG. O número de instantes capturados `I` é

```
I = floor(p /(c*b*s/8))
```

e o número total de amostras, igual a

```
I * c * b
```

onde p = *packet size in bytes*, c = *number of channels*, b = *number of boards,* s = *bits per sample.*

6.2. Mensagens de resultado

As mensagens de resultado devem ser capazes de retornar os resultados das funções de *get* e mensagens de erro caso o comando não seja reconhecido ou caso algum valor de *set* esteja fora do intervalo permitido.

Como o tamanho das mensagens agora é variável, é necessário especificar o tamanho da mensagem quando a insttrução for dos tipos me, mw ou ms.

Instrução	Op. 1	Op. 2	Comentário
vu	uint32	N/A	Value of type uint32
vf	float32	N/A	Value of type float32
me	uint32	string	Message length; Error message
mw	uint32	string	Message length; Warning message
ms	uint32	string	Message length; Streaming packet

Seguem alguns exemplos de mensagens de erro:

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

tiva.setSampleRate: Error: sample rate value = -100 outside supported interval [1..4000000].

tiva.setNumBoards: Error: number of boards = 12 outside supported interval [1..6].

tiva.setNumChannels: Error: number of channels = -5 outside supported interval [1..4].

tiva.parseCommand: Warning: instruction = “aa” unknown.
```