UNIVERSIDADE FEDERAL DE SERGIPE
PRÓ-REITORIA DE PÓS-GRADUAÇÃO E PESQUISA
COORDENAÇÃO DE PESQUISA

PROGRAMA DE INICIAÇÃO CIENTÍFICA VOLUNTÁRIA – PICVOL

CAPTURA E ANÁLISE DE SINAIS DE MIOGRAFIA PARA CONTROLE DE PRÓTESES

Área do conhecimento: Sistemas de Computação
Subárea do conhecimento: Hardware, Processamento de Imagens
Especialidade do conhecimento: Implementação do Dispositivo de Captura em Proto-Board

Relatório Final
Período da bolsa: de Agosto 2017 a Julho 2018

Este projeto é desenvolvido com bolsa de iniciação científica

PICVOL

Sumário

1 Introdução

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. Para que esses sinais obtidos possam ser classificados se faz necessário de um treinamento sobre um dataset. Uma vez treinado o classificador, o sinal poderá ser usado para controlar próteses mecânicas. As próteses podem ser prototipadas usando impressoras 3D disponíveis na própria UFS. O trabalho aqui mostrado persiste de um modulo para esse projeto, um software que possa gerar esse dataset para futuros treinamentos.

2 Objetivos

O sistema aqui desenvolvido tem como objetivo permitir que um computador se comunique com a placa Tiva, exiba na tela e salve em arquivo os sinais capturados pela placa. O software desenvolvido interage com o leapmotion ou teclado e de maneira simultânea com a placa Tiva, tendo como função principal, carregar uma rotina pre-estabelecida e exibir na tela os movimentos nela pre-definidos, movimentos esses que devem ser feitos pelo usuário, seja uma sequência de gestos com a mão (leapmotion) ou de teclas pressionadas (teclado). Por fim, criar um arquivo de saída que contenha todas as informações passadas pelo dispositivo utilizado e seu instante de tempo, o dataset. Simultaneamente, outros dois alunos fizeram a parte inicial da captura de sinais miográficos e sua iteração básica entre a interface e a placa Tiva.

3 Metodologia

Para dar o inicio ao desenvolvimento se fez necessitário definir qual linguagem seria utilizada e quais ferramentas seriam responsáveis por auxilar a implementação. Sobre a linguagem, logo foi definida a Python2, o principal fato da escolha se deu pelo falo dos outros dois desenvolvedores estarem a-utilizando para realizar a interação com a placa.

3.1 Interface

Para realizar a implementação da interface usamos Python2 em conjunto do empacotador Pyqt-4 (responsável por realizar e facilitar a criação de interface gráfica). Na tela inicial é apresentado um menu de opções com:

- Menu file: nela são encontrada as funções que permitem ao usuário trabalhar com os arquivos CSV ou EMG signal.

  Load capture: nela o usuário pode carregar um arquivo de captura CSV previamente salvo

Save capture: aqui o usuário pode salvar dados capturados em um arquivo CSV. Esses dados finais são a junção da captura dos dados do Leap Motion ou teclado, em conjunto dos da placa.

Load EMG signal: nela o software pode escolher um aquivo do EMG para emular o uso do dispositivo de captura de sinais de EMG, permitindo assim, testar o sistema mesmo sem o dispositivo.

- Menu capture:

  Start capture: responsável por inicia a captura.

  Stop capture: responsável por parar a captura.

  Show capture: exibe todos os dados previamente capturados que foram salvos.

- Menu settings:

  Load settings: aqui o programa carrega arquivo com os parâmetros de captura e exibição.

  Save settings:nela o programa salva arquivo com os parâmetros de captura e exibição.

  Capture settings: responsável por abrir o diálogo para edição dos parâmetros de captura.

  Display settings: responsável por abrir o diálogo para edição dos parâmetros de exibição.

3.1.1 Slide

Um dos primeiros passos realizados na interface foi implementação do slide. Slide esse que tem como função apresentar uma sequencia de imagens com um tempo específico para sua exibição, qual imagem e o tempo que ela será exibida é definido por cada linha do arquivo $default.txt$ que é escolhida por padrão, outra possibilidade é que o usuário possa criar sua própria rotina e carrega-la no programa.

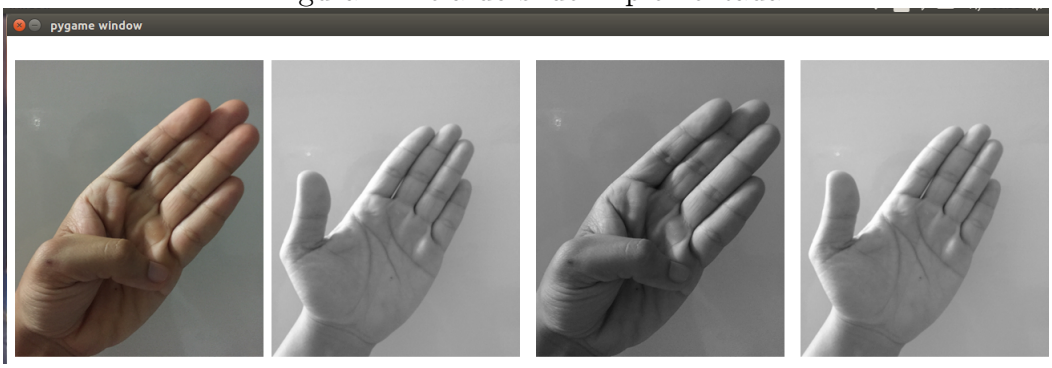
Figura 1: ALTERAR

```
# 2 - indicator
2in_flex;2
hand_open;2
2in_flex;2
hand_open;2
2in_flex_curl;2
hand_open;2
```

Fonte: Criado pelo autor (2018)

Logo quando lido, cada nome e tempo correspondente a um movimento é adicionado a uma lista. Em seguida a função *converter* é acionada, sua principal função é padronizar as imagens que serão exibidas na tela. Assim que acionada ela recebe como parâmetro a lista das imagens e realiza a chamada das imagens na parta local *images* fazendo o tratamento da imagem para escala cinza e a definindo com dimensão 310x370.

Figura 2: Tela do slide implementada



Fonte: Criado pelo autor (2018)

O resultados desse tratamento é salvo no mesmo local, assim, logo em seguida um laço percorre todas as imagens da lista, chamando as imagens resultantes em 4 espaços, onde a imagem é passada para próxima decorrendo do tempo que foi definido ainda no arquivo. Para que o tempo de exibição definido seja satisfeito, se fez necessário da biblioteca *timer* encontrada no padrão do Python2, a função responsável por fazer esse delay foi a *timer.sleep*(). Já para a exibição da imagem resultando foi utilizado a função *pygame.image.load*(*caminho*) da biblioteca Pygame. Além disso o Pygame foi utilizada em outras partes do software, a razão da sua escolha é algo que será discutida no decorrer do relatório.

3.2 Miografia:

Qualquer músculo esquelético quando contraído gera sons que são fáceis de ser notados e gravados. A gravação desses sons musculares é chamada de miografia acústica (AMG)". A existência dessa capacidade vem desdo século 19, porém não havia tentativas de sinais de alguma maneira. Oster e Jaff.Z.J iniciaram uma pesquisa moderna sobre os sons musculares, foi encontrado uma relação linear para amplitude de som com a força isométrica, gerando o estudo do componente de frequência de pouco em torno de 25hz. Os dois fizeram o uso da AMG com a electromiografia(EMG), tratando ela como não invasiva em V LYJ, usando o método para monitorar a dissociação de eletricidade apartir da mecânica, ocorrendo uma fadiga.

3.3 Leap Motion

O Leap Motion é um dispositivo de hardware que possui a capacidade de captar gestos da mão via infravermelho, transformar em dados e enviar essas dados para o dispositivo ao qual ele esteja conectado e devidamente instalado, no caso aqui relatado, o PC. A sua missão final é romper as barreiras entre pessoas e tecnologia.

Segundo GUNA et al. (2014) o Leap Motion Controller representa um grande avanço na tecnologia sobre entrada, seja atraves das mãos e gestos do usuário. Disponibilizado em 2013 o dispositivo tem tido uma grande crescente nas pesquisas que buscam trabalhar com a manipulação de braços robóticos. No seu funcionamento o Leap usa de imagens infravermelhas (IR) para poder estabelecer a posição dos objetos que estão sendo definidos no espaço em tempo real. Atualmente pouco se sabe dos algoritmos utilizados pela empresa no aparelho para se fazer essa captação.

Porém, depois de uma analise feita no seu controlador ficou constatado o uso 3 LEDS IR com 2 câmeras IR, fazendo assim com que o controlador possa ser categorizado como um sistema de rastreamento óptico, baseando-se no principio de visão estéreo. O software do Leap realiza a inspeção dos objetos no seu campo de visão e reconhece mãos, dedos e ferramentas, se referindo a posições discretas como gestos e movimentos feitos. Como mostra informações oficiais no site Leap Motion Controller (2010), o software Leap realiza uma analise dos objetos observados que se encontram no campo seu campo de visão e seu controlador é acessado através de APIS e suportes para diversas linguagens de programação, no caso aqui implementado, o Python2.

Figura 3: Imagem Original do Produto

Fonte: https://leapmotion.com/about

Implementando

Inicialmente se fez necessitário realizar um estudo para descobrir as ferramentas que seriam utilizadas, nesse caso o para interagir com o Leap Motion, a própria empresa responsável pelo dispositivo fornece uma biblioteca sua biblioteca especifica para linguagem Python2 que assim que instalada, não só possibilita, como facilita a detecção dos dedos das mãos. Em seguida se deu o inicio ao estudo da documentação do Leap Motion, que teve como primeiro procedimento a instalação da sua biblioteca no SO (Linux), instalação essa que ocorreu de maneira esperada onde todo material necessitário foi encontrado no site da empresa.

Com o Leap Motion já devidamente instalado, se deu inicio ao estudo da biblioteca do dispositivo e logo foi notado que, sempre que inserido, o dispositivo envia informações para o computador através da classe *Controller* que pode ser facilmente acessada após a importação da biblioteca do Leap Motion. Assim, logo na primeira implementação de teste, foi criado uma classe *LeapListener* onde a mesma possui uma lista de fingers ('Polegar','Indicador','Meio','Anular','Mindinho') e uma lista de bones ('Metacarpo','Proximal','Intermediário','Distal'), onde cada dedo possui todos os ossos e cada osso é tido como uma variável do tipo int. Após a criação da classe ela foi instanciada no *main.py* junto com classe *Controller* (Pertencente ao Leap

Motion) e logo quando dispositivo é inserido atraves do *controller.add* é passado como parâmetro a classe *LeapListener* instanciada e lhe é atribuído a cada variável que representa o osso do dedo um valor inteiro passado pelo dispositivo. Porém se fazia necessário que todos essas informações fossem constantes, então, foi feito um laço com valor *true* que a cada iteração faz o *controller.add* e o *controller.remove* (Insere e Remove o dispositivo) fazendo assim com que se possa ter as informações do Leap Motion por um tempo indeterminado.

Alterando a Escala dos Dados e Verificando Posição dos Dedos

Inicialmente o Leap Motion entrega essas informações representadas em números numa escala de -100 até 100 para cada osso do dedo instanciado. Porém para facilitar a leitura feita pela placa que receberá essas informações, foi feito uma função responsável por alterar a faixa para 0 - 100, fazendo com que o resultado obtido possa ser compreendido facilmente. Assim, cada dedo ficou assumido como positiveis 3 estados para mão, aberta, medial dobrado, proximal dobrado e fechada.

Para fazer o calculo e saber diferenciar cada movimento, foi usada uma função chamada $verificaMov()$ que fica encontrada dentro da classe *LeapListener*, ele usa as informações dos ossos, distal, medial e proximal. Para diferenciar o as posições da mão, basta analisar seu respectivo osso do dedo. Supondo que a mão esteja aberta e o angulo do medial mude, se o proximal não, significa que ele possivelmente esteja no medial dobrado. Como a função *LespListener* é instanciada e usada a cada iteração, a verificação de cada dedo ocorre constantemente. Todas essas informações captadas são armazenadas em um arquivo *.log*. Nele se encontra a informação de todos dedos e em qual posição ele se encontra em um determinado instante devido, tudo graças laço *whiletrue*, que faz com que a $verificaMov()$ seja chamada a cada iteração. Permitindo assim uma criação de rotina de movimentos como $moaberta$, ou $mofechada$.

3.3.1 Melhorias na Interface

Depois da implementação do slide que passa a sequencia de imagens, alguns outros métodos se fizeram necessários para realizar melhorias de interface e consequentemente, uma melhor resposta na interação com o usuário:

- Barra de tempo da Imagem: Seu intuito é representar o tempo falta para que a imagem que esta sendo exibida atualmente seja exibida. Retorna ao usuário uma perspectiva de quanto tempo falta para acabar a exibição da imagem atual. Barra de rotina: Representa o tempo total para o fim da rotina, garante uma perspectiva ao usuário de quanto tempo falta para o fim.

- Barras de status: A barra de status foi implementada em com o uso do Pygame. É de uso pratico e fácil, ela representa o estado atual do dedo correspondente, seja - ou 100 pela simulação com o teclado, ou de 0 até 100 com o leapmotion.

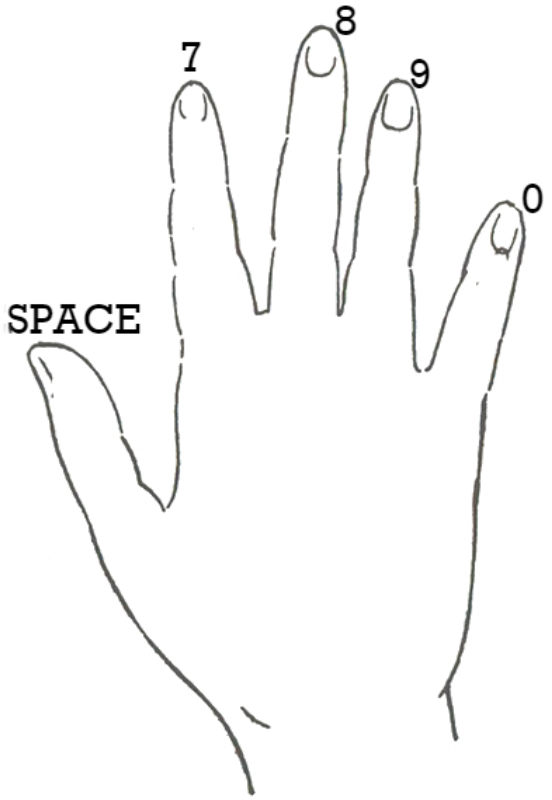
- Tela com a próxima imagem a ser exibida: A implementação das próximas 3 telas tela foi feita com o intuito de auxiliar o usuário informando-o os proximismos gestos pretendidos a captação.

-

3.4 Implementação do Teclado

A implementação do teclado foi uma etapa um tanto custosa. A ideia final era fazer com que enquanto ocorra a passagem das imagens com os gestos, o usuário possa pressionar as teclas correspondentes que pre-estabelecidas. O que vale ser ressaltado é diferentemente do Leap Motion, que retorna a faixa numa escala de 0 a 100, pelo fado de cada tecla possuir apenas dois valores possíveis (pressionado e não pressionado), logo o arquivo de saída ".log"que ele dará, terá apenas dois possíveis valores, 0 ou 100, o mesmo vale para os retornos na interface. Os dedos das mãos são representados por cada tecla no programa, porém a simulação é feita somente com uma mão, logo, 5 teclas devem ser tratadas simultaneamente.

Figura 4: Trecho do Codigo Responsável por Alterar a Faixa



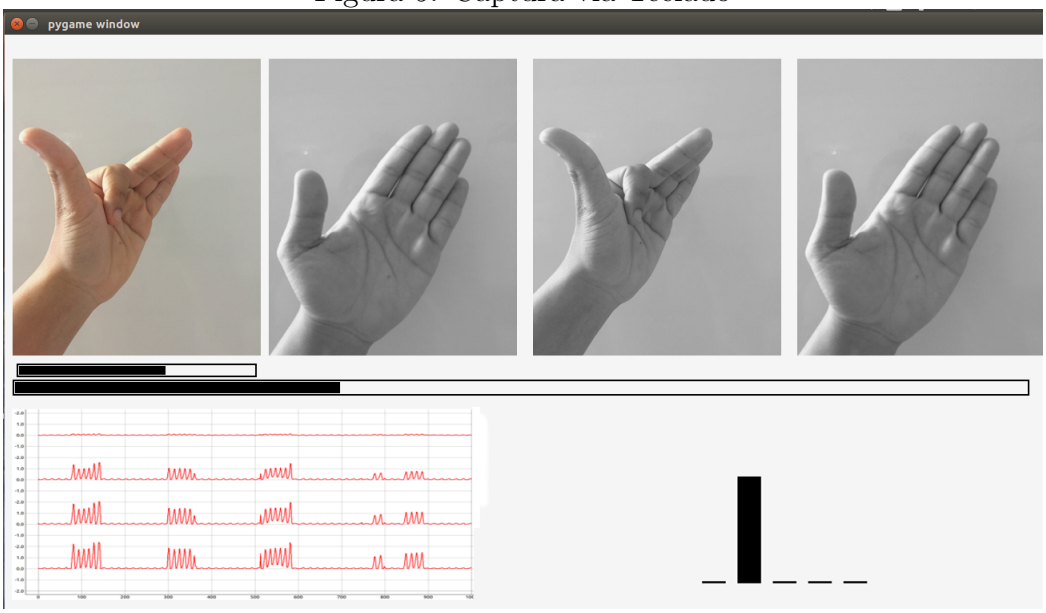
Fonte: Criado pelo autor (2018)

Para realizar a captura simultânea da tecla foi utilizado o $pygame.key.get_pressed()$, função que retorna o estado atual da tecla, nesse caso se ela está pressionada ou não.

4 Resultados e discussões

Como resultado foi o obtido um software final com todas as funcionalidades estabelecidas no documento.

Figura 5: Captura via Teclado



Fonte: Criado pelo autor (2018)

Na imagem acima podemos ver todas as funcionalidades implementadas.
1º Slide contendo a imagem atual e as próximas que serão exibidas;
2º Barra de imagem atual e a total, exibindo quanto tempo falta para a exibição da imagem atual acabar;
3 º Exibição dos sinais da eletromiografia captados pela placa Tiva localizado no canto inferior esquerdo
4 º Exibição da tecla 7 que estava pressionada no instante canto inferior direito;
Vale notar que a tecla atual pressionada está seguindo o movimento esperado indicado pela imagem colorida atual do slide, imaginando com o leapmotion seria correspondente ao dedo indicador;

Figura 6: Arquivo de saída CSV da Captura via Teclado com a Placa Tiva

```
# File  generated  by  leap_cap  software
## Available  from github.com/ddantas/leap_cap##  Timestamp:  2018-05-23_20-57-3
##
## EMG  capture  settings
##
# sampleRate:  2000
# channelsPerBoard:  4
# nBoards:  1
# bitsPerSample:  12
##
## Gesture  capture  settings
#
## device:  keyboard
# routine: default.csv
# hand:  right
##
## Datath_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; 0; 0; 0;
0.1;0 ;0 ;0 ;0 ; 0; 142; 187 ; 254; 290
0.2; 0; 0; 0; 0; 0; 68; 106; 186; 219
0.3; 0; 0; 0; 0; 0; 46; 70; 138; 158
0.4; 0; 0; 0; 0; 0; 38; 51; 98;130
0.5; 0; 0; 0; 0; 0; 36; 46; 79;106
0.6; 0; 0; 0; 0; 0; 30; 38; 74;95
0.7; 0; 0; 0; 0; 0; 36; 40; 62;80
0.8; 0; 0; 0; 0; 0; 36; 40; 58;74
0.9; 0; 0; 0; 0; 0; 36; 43; 58;66
1.0; 0; 0; 0; 0; 0; 30; 34; 51;63
1.1; 0; 0; 0; 0; 0; 30; 34; 50;58
1.2; 0; 0; 0; 0; 0; 32; 34; 46;50
```

Fonte: Criado pelo autor (2018)

5 Conclusões

O sistema foi desenvolvido de maneira correta, sua interação com espera-se um periodo de adaptação com os dispositivos envolvidos, seja a placa tiva, leapmotion ou teclado. Como foi desenvolvido com Python2, sua instalação e uso possuiO arquivo de saída CSV servirá como um dataset importante para trabalhos futuros.

O sistema aqui desenvolvido tem como objetivo permitir que um computador se comunique com a placa Tiva, exiba na tela e salve em arquivo os sinais capturados pela placa. O software desenvolvido interage com o leapmotion ou teclado e de maneira simultânea com a placa Tiva, tendo como função principal, carregar uma rotina pre-estabelecida e exibir na tela os movimentos nela pre-definidos, movimentos esses que devem ser feitos pelo usuário, seja uma sequência de gestos com a mão (leapmotion) ou de teclas pressionadas (teclado). Por fim, criar um arquivo de saída que contenha todas as informações passadas pelo dispositivo utilizado e seu instante de tempo, o dataset.

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. Para que esses sinais obtidos possam ser classificados se faz necessário de um treinamento sobre um dataset. Uma vez treinado o classificador, o sinal poderá ser usado para controlar próteses mecânicas. As próteses podem ser prototipadas usando impressoras 3D disponíveis na própria UFS. O trabalho aqui mostrado persiste de um modulo para esse projeto, um software que possa gerar esse dataset para futuros treinamentos.

6 Perspectivas

A ideia é que os próximos trabalhos usem os resultados gerados pelo software para criar datasets de testes e treinamento, para que assim, esses datasets possam ser treinados através de técnicas de aprendizado computacional (redes neurais) e em seguida possa ser classificado. Uma vez que este classificador estiver treinado, poderá ser usado para controlar próteses mecânicas. Futuramente esse projeto poderá aproximar o DCOMP de outros departamentos, como o de Fisiologia, Engenharia Mecânica, Engenharia Elétrica e Medicina. Caso seja bem sucedido, a ambição final deste projeto é implementar mãos mecânicas de baixo custo que podem vir a ser usadas por amputados.

7 Referências bibliográficas

JOZE, G.; GREGA, J.; MATEVZ, P.; SASO, T.; JAKA, S.;. An Analysis of the Precision and Reliability of the Leap Motion Sensor and Its Suitability for Static and Dynamic Tracking. MDPI, Basileia,Faculty of Electrical Engineering, University of Ljubljana, Tržaška 25, Ljubljana 1000, Slovenia, 10.3390, 2-5, fevereiro de 2014.

Leap Motion Controller. Disponível on-line: https://www.leapmotion.com (acessado em 23 de janeiro de 2018).

DANIEL, T.; BARRY, MD.; JAMES, A.; LEONARD, JR.; ANDREW, J.; GITTER, MD.Acoustic Myography as a Control Signal for an Externally Powered Prosthesis . Archives of Physical Medicine and Rehabilitation, Departament of PM&R, University of Michigan, Ann Arbor, MI 48109, 2-3, abril de 1986.

8 Outras atividades

Não houve outras atividades.