UNIVERSIDAD NACIONAL DE LA PLATA

JCS&T Journal of Computer Science & Technology

Artigos Originais

Algoritmo de Processamento de Sinais Mioelétricos para Controle de Mãos Robóticas Protéticas

Rodrigo E. Russo rodrigo.e.russo@gmail.com
Universidade Nacional de Mar del Plata , Argentina

Juana G. Fernández gfernand@fi.mdp.edu.ar
Universidade Nacional de Mar del Plata , Argentina

Raúl R. Rivera rrivera@fi.mdp.edu.ar
Universidade Nacional de Mar del Plata , Argentina

Melisa G. Kuzman
Universidade Nacional de Mar del Plata , Argentina

Juan M. Lopez
Universidade Nacional de Mar del Plata , Argentina

Walter A. Gemin
Universidade Nacional de Mar del Plata , Argentina

Miguel A. Revuelta
Universidade Nacional de Mar del Plata , Argentina

Algoritmo de Processamento de Sinais Mioelétricos para Controle de Mãos Robóticas Protéticas
Journal of Computer Science and Technology , vol. 18 , n. 1 , 2018
Universidade Nacional da Prata

Recebido: 08 de fevereiro de 2018
Revisado: 21 de março de 2018
Aceito: 09 de abril de 2018
DOI: https://doi.org/10.24215/16666038.18.e04

Abstrato:
O desenvolvimento de próteses robóticas para as mãos visa devolver às pessoas com deficiência a capacidade de recuperar

a funcionalidade necessária para manipular os objetos do seu ambiente diário. Os sinais elétricos enviados pelo cérebro através do sistema nervoso estão associados ao tipo de movimento que os membros devem executar. Os sensores mioelétricos são dispositivos não intrusivos que permitem a captura de sinais elétricos do sistema nervoso periférico. A relação entre os sinais originados no cérebro que tendem a gerar uma ação e os mioelétricos como resultado deles, são fracamente correlacionados. Por esse motivo, é necessário estudar sua interação para desenvolver os algoritmos que permitem reconhecer ordens e transformá-las em comandos que ativam os movimentos correspondentes da prótese. O presente trabalho mostra o desenvolvimento de uma prótese baseada no projeto de uma biônica aberta manual para produzir movimentos, o sensor MyoWare Muscle para captura de sinais mioelétricos (EMG) e o algoritmo que permite identificar ordens associadas a três tipos de movimento. O módulo Arduino Nano executa os processos de aquisição e controle para atender aos requisitos de tamanho e consumo desta aplicação.

Palavras-chave:
EMG, Prótese, Robótica, Arduino, Mão, Biônico.

1. Introdução

As próteses manuais controladas por meios bioelétricos constituem o tipo de membro artificial com o mais alto grau de reabilitação, pois podem sintetizar o aspecto estético, a força e a velocidade de preensão, bem como muitas possibilidades de adaptação a diferentes graus de incapacidade. O controle mioelétrico é o esquema de controle mais difundido devido às suas características não invasivas. É baseado em um princípio fisiológico que indica que sempre que um músculo do corpo se contrai ou flexiona, há um pequeno sinal elétrico (bioelétrico) que o produz. Isso provém de uma interação química no corpo e produz um sinal muito pequeno (5 a 20 μV) que pode ser capturado com eletrodos de superfície (chamados adesivos ou almofadas) montados em contato com a pele.

Como existe uma correspondência entre a atividade muscular e os sinais eletromioelétricos (EMG), é possível extrair informações deles para identificar movimentos. Esse processo de classificação foi estudado em profundidade [1] e, para isso, existe uma grande variedade de algoritmos, cuja eficiência será avaliada.

O uso de sinais mioelétricos para o controle de próteses evoluiu de simples implementações ON-OFF para mais avançadas desde a incorporação de microcontroladores para a aquisição, processamento e controle deles que também oferecem benefícios adicionais como tamanho, consumo e custo [2] [3] [4]

Há muitas informações valiosas sobre os procedimentos para a identificação e classificação de movimentos aplicados ao desenvolvimento de próteses [5] [6].

O trabalho atual sobre o tópico inclui a análise de diferentes recursos usando vários sensores, abordando a possibilidade de usar um algoritmo de classificação para detecção de movimento [7] [8] [9]. Apesar de essas informações serem valiosas, entra em conflito com a ideia de usar um único sensor para favorecer a portabilidade, pois isso acabaria classificando um único movimento. Devido à portabilidade, espera-se obter o máximo de informações possível usando um número mínimo de músculos, mas mantendo um número útil de movimentos para serem classificados em tempo real.

O sensor usado para medir o sinal EMG é o MyoWare Muscle Sensor e uma placa Arduino Nano para realizar a amostragem do sinal.

Em particular, é desejável detectar três movimentos da mão a partir de um sensor no músculo flexor radial do carpo no antebraço. Os três movimentos são mostrados na Fig. 1 e consistem em fechamento do punho (chamado c), pressão com a ponta do dedo (p) e flexão do punho para dentro (f).

O sinal obtido é enviado para um PC portátil, onde é condicionado, segmentado e filtrado para extrair recursos úteis. Esses recursos são o valor médio do sinal em valor absoluto (MAV), comprimento da forma de onda (WL), cruzamentos zero (ZC) e alteração de sinal na inclinação (SSC).

Fig. 1 c

Fechamento dos movimentos da mão

Fig. 1 p

Pressão dos movimentos da mão

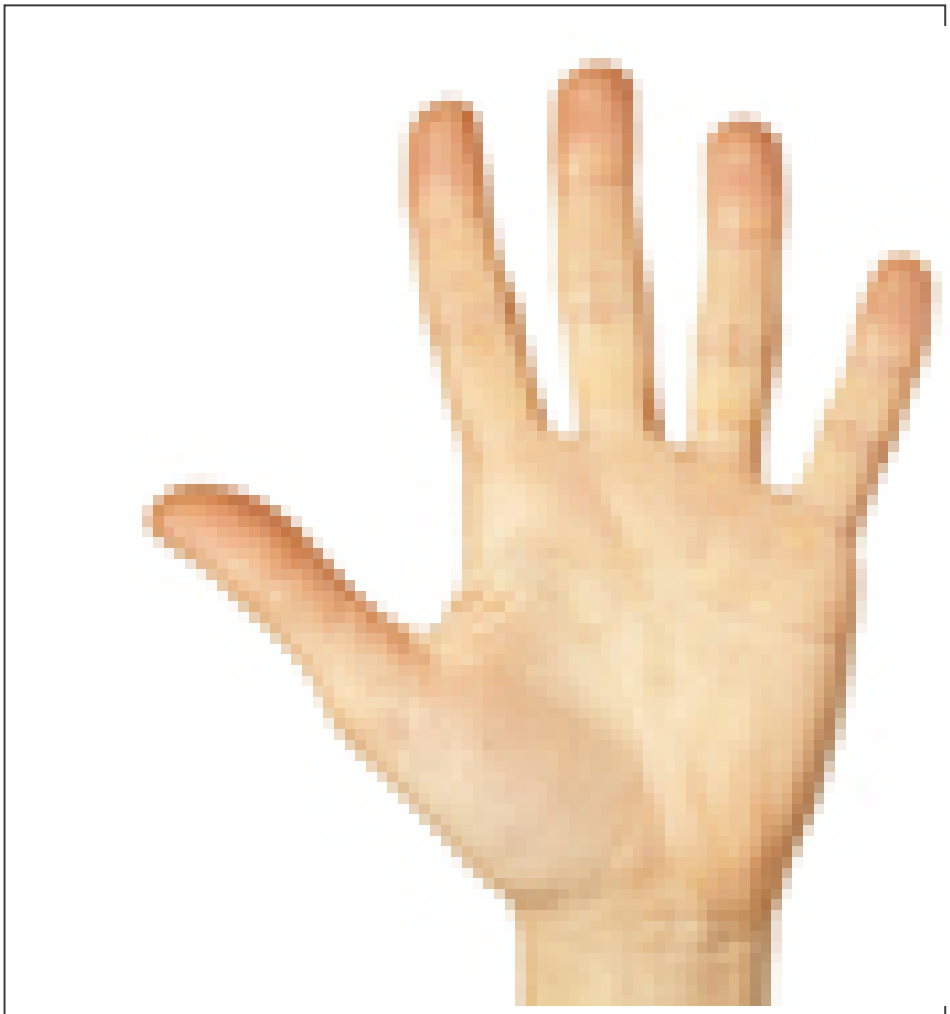

Fig. 1 f

Flexão dos movimentos da mão

Para diferentes movimentos, os valores médios de algumas características apresentam diferenças, portanto, é possível dividir o espaço de forma que uma classificação possa ser feita. Três algoritmos de classificação foram testados: Máquinas de Vetor de Suporte, Vizinhos Mais Próximos e Redes Neurais, usando as bibliotecas de Machine Learning para Python, scikit-learn [10] As maneiras de implementar a classificação consistem em dois procedimentos: um em lotes e outro em linha. Para o primeiro caso, são obtidas amostras dos diferentes movimentos e designados conjuntos de treinamento e teste. Para o segundo, que é o caso de utilidade e interesse, as amostras são coletadas simultaneamente com o processamento. Depois de coletar as primeiras amostras, que treinam o classificador, as próximas amostras são classificadas simultaneamente com a captura do movimento.

2. Componentes da prótese

2.1 Mão robótica de biônica aberta

A Open Bionics é uma empresa dedicada à difusão e comercialização de mãos biônicas construídas usando

tecnologias de digitalização e impressoras 3D. Essas próteses baseiam-se na estrutura e na funcionalidade da mão humana, uma vez que é o efetor final mais versátil e hábil conhecido. O projeto utiliza um modelo antropomórfico tanto na cinemática quanto no sistema de transmissão e transmissão. As etapas da construção começam com a varredura que permite replicar o órgão amputado do paciente, para criar um modelo tridimensional de sua geometria. Este procedimento permite construir próteses que se encaixam corretamente. Sensores mioelétricos colocados em um músculo saudável no membro do paciente são usados para capturar os sinais elétricos e gerar os diferentes padrões de aderência nos dedos.

2.2 Sensor Myoelétrico MyoWare

Esse sensor de sinais mioelétricos mede, filtra, retifica e amplifica a atividade elétrica do músculo e produz na saída um sinal analógico com amplitude suficiente para ser lido e processado por um microcontrolador que controla os movimentos da mão artificial. Esse tipo de sensor é tradicionalmente usado para a investigação e diagnóstico de distúrbios neuromusculares. Neste trabalho, ele é combinado com um módulo de microcontrolador Arduino Nano que oferece as características de tamanho, consumo e potência de computação mais apropriadas para esta aplicação.

2.3 Módulo de Aquisição

Uma placa Nano Arduino é usada para coletar amostras do sinal emitido pelo sensor MyoWare a 1ks / s. O sensor é conectado ao músculo flexor radial do carpo, como indicado na Fig. 2, e ao eletrodo de referência fora da zona de atividade mioelétrica do músculo em análise.

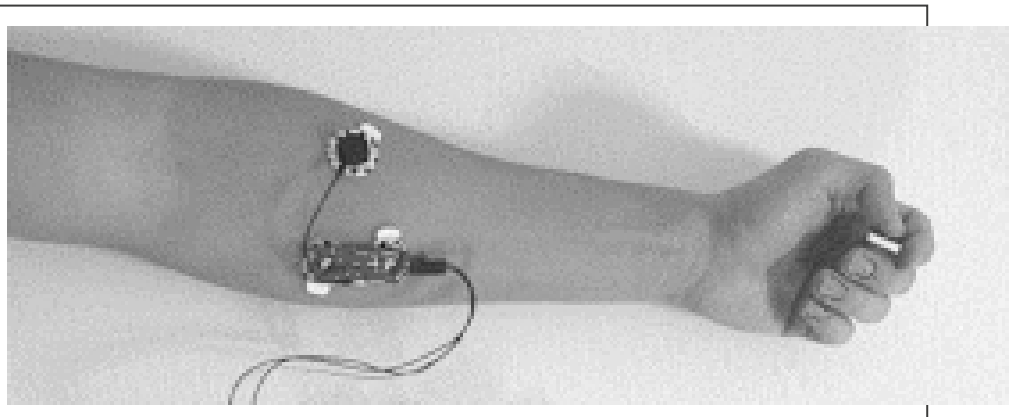

Fig. 2

Posição dos eletrodos

Os conjuntos de teste realizados consistem em 100 movimentos repetidos a cada 3 segundos de cada um,

chamados c, f e p. As amostras foram enviadas em série para um PC portátil executando um programa de aquisição em Python e posteriormente são usadas para a caracterização de cada um dos três movimentos.

3. Processando

Os conjuntos de dados obtidos correspondem ao sinal EMG, estão centralizados em um nível de tensão, apresentam variações em seu valor médio e não são estacionários. Antes da extração dos recursos que permitem a classificação, o sinal é filtrado e segmentado.

Os sinais obtidos durante esse processo são mostrados na Fig. 3.

- Filtragem: um Butterworth 10^{th} ordem IIR banda passante do filtro digital é utilizada usando SciPy [11] com frequências de corte de 50 e 400 Hz com base nos resultados empíricos e estudos [12].
- Segmentação: os segmentos de sinal que não são nulos são extraídos, ou seja, onde o músculo está ativo. Para isso, é realizada a operação Eq (1) , que consiste em diferenciar o sinal e obter seu valor médio, obtendo janelas de comprimento N amostras. Em seguida, um limite é aplicado para detectar os picos desse novo sinal e o EMG é segmentado com os picos como centro.

$$MAD = \frac{1}{N}\sum_{i=0}^{N-1}|x_{i+1} - x_i| \quad \text{[Eq.1]}$$

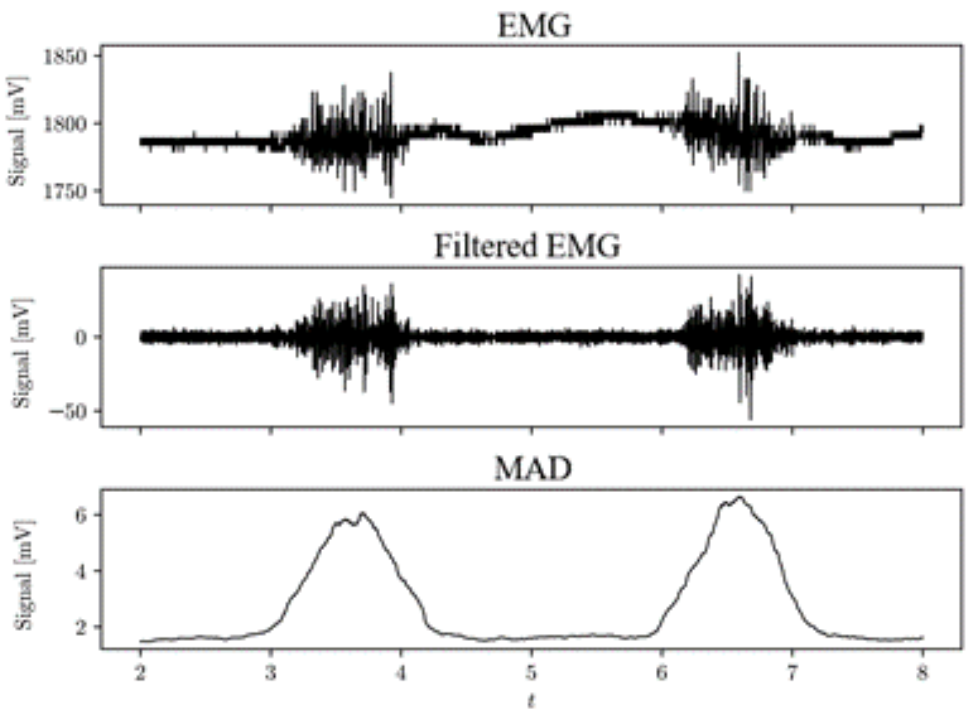

Fig. 3

EMG: sinal bruto, filtrado e seu MAD

3.1 Extração de recursos

A partir dos sinais processados, foram extraídas quatro características do domínio temporal, capazes de diferenciar os três movimentos de interesse. A seleção foi decidida com base em sua simplicidade, embora outros trabalhos estendam a análise a um número muito maior [7] [8] [9]. Esses recursos são calculados em uma janela de tempo do sinal. Após a segmentação, é feito um número fixo de amostras para avaliar cada característica. As operações correspondentes são as seguintes:

1. Valor Absoluto Médio (MAV): o valor médio dos valores da janela é calculado no valor absoluto Eq (2) .

$$MAV = \frac{1}{N}\sum_{i=0}^{N}|x_i| \quad \text{[Eq. 2]}$$

2. Comprimento da forma de onda (WL): fornece informações sobre a complexidade do comprimento da forma de onda, sendo uma medida de sua amplitude, frequência e duração em um único parâmetro [1]. É calculado como a soma no valor absoluto das diferenças entre duas amostras sucessivas Eq (3) .

$$WL = \sum_{i=0}^{N-1}|x_{i+1} - x_i| \quad \text{[Eq.3]}$$

3. Cruzamento Zero (ZC): conte o número de cruzamentos por zero no segmento. Para evitar o cruzamento de zero devido ao ruído, é adicionado um limiar a partir do qual o cruzamento é registrado na Eq (4) .

$$ZC = \sum_{i=0}^{N-1} f(x_i . x_{i+1}) \quad \text{[Eq.4]}$$

$$f(x) = \begin{cases} 1 & x \geq umbral \\ 0 & x < umbral \end{cases} \quad \text{[Eq.5]}$$

4. Alteração do sinal de inclinação (SSC): semelhante ao recurso ZC, mas aplicado na inclinação da forma de onda Eq (6) .

$$SSC = \sum_{i=0}^{N-2} f[(x_{i+1} - x_i).(x_{i+2} - x_{i+1})] \quad \text{[Eq.6]}$$

Para garantir a repetibilidade, os recursos são normalizados com relação à média e desvio padrão. Na Fig. 4 , é mostrado como essa normalização influencia a separação das categorias, ao aplicar o PCA (Principal Component Analysis).

As melhorias após a normalização são notáveis e suas coordenadas de referência são calculadas durante o treinamento e armazenadas para definir os clusters que permitem, posteriormente, identificar o movimento associado ao sinal mioelétrico.

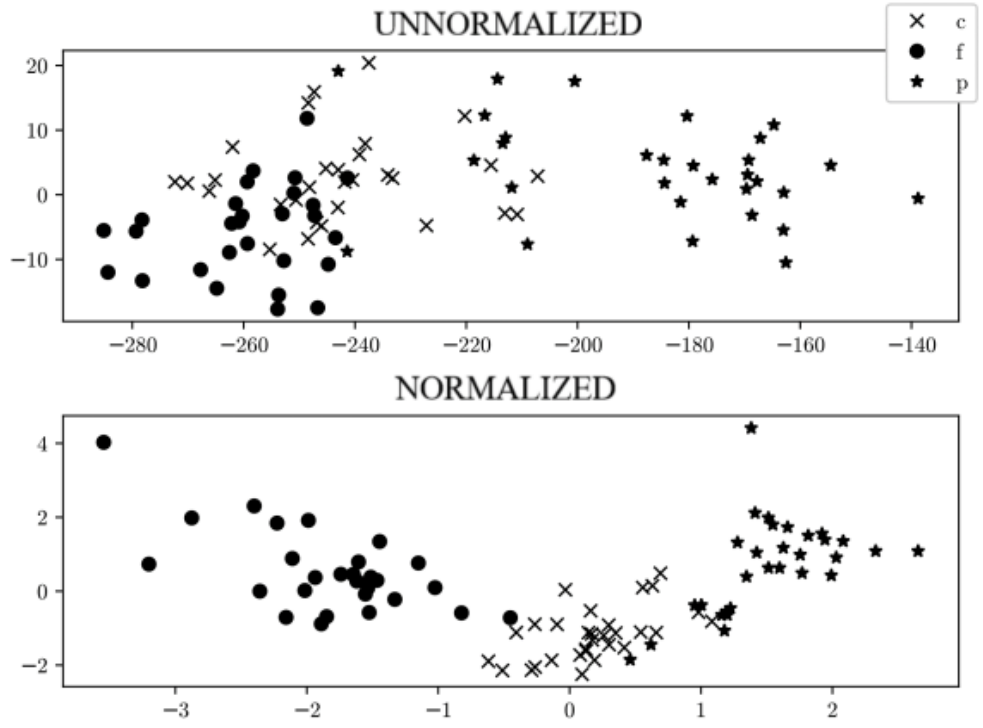

Fig. 4

Recursos não normalizados e normalizados reduzidos para duas dimensões

3.2 Classificação

Uma vez obtidos os recursos para diferentes repetições dos diferentes movimentos, é pesquisado um classificador que, para um número mínimo de amostras de treinamento, pode classificar com um erro tolerável as novas amostras.

Três algoritmos de classificação foram escolhidos para comparar os resultados posteriormente:

Máquinas de vetores de suporte: consiste em um mapeamento não linear da entrada para um espaço maior. Em seguida, é encontrado um hiperplano que generaliza a separação do espaço de feições [13] [14].

K-vizinhos mais próximos: os rótulos são atribuídos às entradas com base em seus k vizinhos. Um volume com a nova entrada em seu centro é avaliado como uma classe de acordo com os pontos contidos nesse volume [9] [15].

Redes neurais: também chamadas de perceptron de múltiplas camadas, consistem em um conjunto de entradas que são transformadas em saídas de acordo com os coeficientes da (s) camada (s) oculta (s). Em particular, ele treina com as entradas e saídas, e os coeficientes são obtidos usando um algoritmo de retropropagação baseado em um esquema de gradiente descendente [16].

Para avaliar o desempenho dos classificadores, as amostras de treinamento são geradas inicialmente e, em seguida, um novo conjunto de amostras para validar a classificação. Isso emula o comportamento pretendido do aplicativo proposto.

4. Resultados

Para avaliar os resultados do sistema, duas fases foram implementadas: a primeira consiste em gerar um grande conjunto de dados de movimentos envolvendo 100 de cada uma delas; o segundo consiste em avaliar dados online de diferentes usuários. O objetivo da primeira implementação é testar diferentes parâmetros do classificador e obter resultados preliminares. Também são propostos tamanhos diferentes de conjuntos de treinamento (10, 20 e 30), para cada movimento, a fim de encontrar um que produza uma melhor adaptação aos dados atuais, usando o menor número possível de amostras de treinamento. A segunda fase é usada como um teste real de como o sistema se comporta com diferentes usuários.

4.1 Resultados preliminares

Foram gerados conjuntos de treinamento de 30 amostras por movimento, desafiando os diferentes parâmetros associados a cada classificador. Cada conjunto de treinamento é testado em um conjunto de dados contendo 100 movimentos por turma, preparados previamente. Com esse procedimento, espera-se encontrar um conjunto de treinamento que consiga classificar, com a maior taxa de sucesso, cada tipo de movimento. Os algoritmos utilizados foram os implementados pela biblioteca Python scikit-learn [10]:

SVM:

- Kernel Linear
- RBF do kernel (gaussiano)

K-vizinhos mais próximos:

- Peso uniforme
- Pesos inversamente proporcionais à distância

Rede neural:

- 10 neurônios na camada oculta
- 50 neurônios na camada oculta

- 100 neurônios na camada oculta

Cada classificador é treinado com 10, 20 e 30 amostras por movimento, para determinar para cada número de amostras que tem o melhor desempenho. Para avaliar esse procedimento, é utilizada uma pontuação entre 0 e 1, calculada conforme mostrado na Eq. (7)

Pontuação = tentativas bem-sucedidas / total de tentativas (Eq.7)

tabela 1

Resultados para diferentes classificadores

Classificador		10	20	30
SVM				
	Kernel linear	0.822	0.844	0,800
	RBF Kernel	0,578	0,644	0.811
K-NN				
	Uniforme	0,411	0,600	0.811
	Distância	0,622	0,667	0.867
Rede neural				
	10	0.856	0.844	0.867
	50.	0.822	0.811	0,889
	100	0.878	0.878	0.878

Para implementar o sistema, os classificadores mais consistentes foram utilizados para diferentes números de amostras de treinamento, especialmente aqueles que têm o melhor desempenho para um número reduzido de amostras. Este procedimento procura encontrar o número mínimo de amostras de treinamento que geram uma classificação o mais precisa possível. Isso ocorre porque, após um certo tempo, o usuário fica cansado e perde a concentração, produzindo resultados incorretos. Além disso, foram estabelecidos protocolos de treinamento que permitem padronizar o teste de diferentes usuários.

4.2 Resultados Online

Para avaliar os resultados, foi desenvolvido um programa que gera diferentes treinamentos e classificações em tempo real. Foram utilizados os classificadores com melhor desempenho (rede neural e SVM) com diferentes números de amostras e 30 movimentos foram classificados

sequencialmente para determinar qual combinação é mais apropriada. Os resultados de cada caso são apresentados na Tabela 2 , 3 .

mesa 2

	Pontuação para SVM		
	10	20	30
Fecho	0,9	0.8333	0,8
Flexão	0,9667	0,9667	1
Pressão	1	0.8667	1
Pontuação Média	0,9556	0,8889	0,9333

Tabela 3

	Pontuação para Rede Neural		
	10	20	30
Fecho	0,6667	0.8333	0,9333
Flexão	0,9	0.8667	0,7667
Pressão	1	0,9333	0,9667
Pontuação Média	0.8556	0.8778	0,8889

A tabela 2 mostra que as melhores pontuações foram obtidas para o classificador SVM. Optou-se por coletar 10 amostras, pois os resultados indicam que é um número adequado, considerando também que o usuário pode manter a concentração durante o procedimento. Quanto maior o número de repetições, mais rápido fica cansado e distraído, o que pode introduzir variações nos sinais mioelétricos gerados e registrados.

Para validar esses procedimentos e verificar a repetibilidade dos parâmetros usados em outras pessoas, esse treinamento foi repetido com 6 usuários diferentes. A metodologia empregada consistiu em colocar o sensor no braço dos diferentes usuários, registrando os resultados e calculando a pontuação da classificação para cada movimento solicitado.

O desempenho médio dos testes é mostrado na Tabela 4, onde é observada a eficiência das classificações estudadas.

Quadro 4

Pontuação média para 6 usuários

	SVM	Rede neural
Fecho	0,9	0,7333
Flexão	0,9	0,6
Pressão	0.8667	1
Pontuação Média	0,8889	0,7778

5. Aplicações

Os resultados obtidos foram testados para controlar uma mão experimental, como mostrado na Fig. 5 , onde são vistos os dedos e os servos que os movem. Foram programados comandos que são enviados por porta serial a uma placa Arduino Nano que controla a operação dos servos e que correspondem aos movimentos classificados, como pode ser visto em [17].

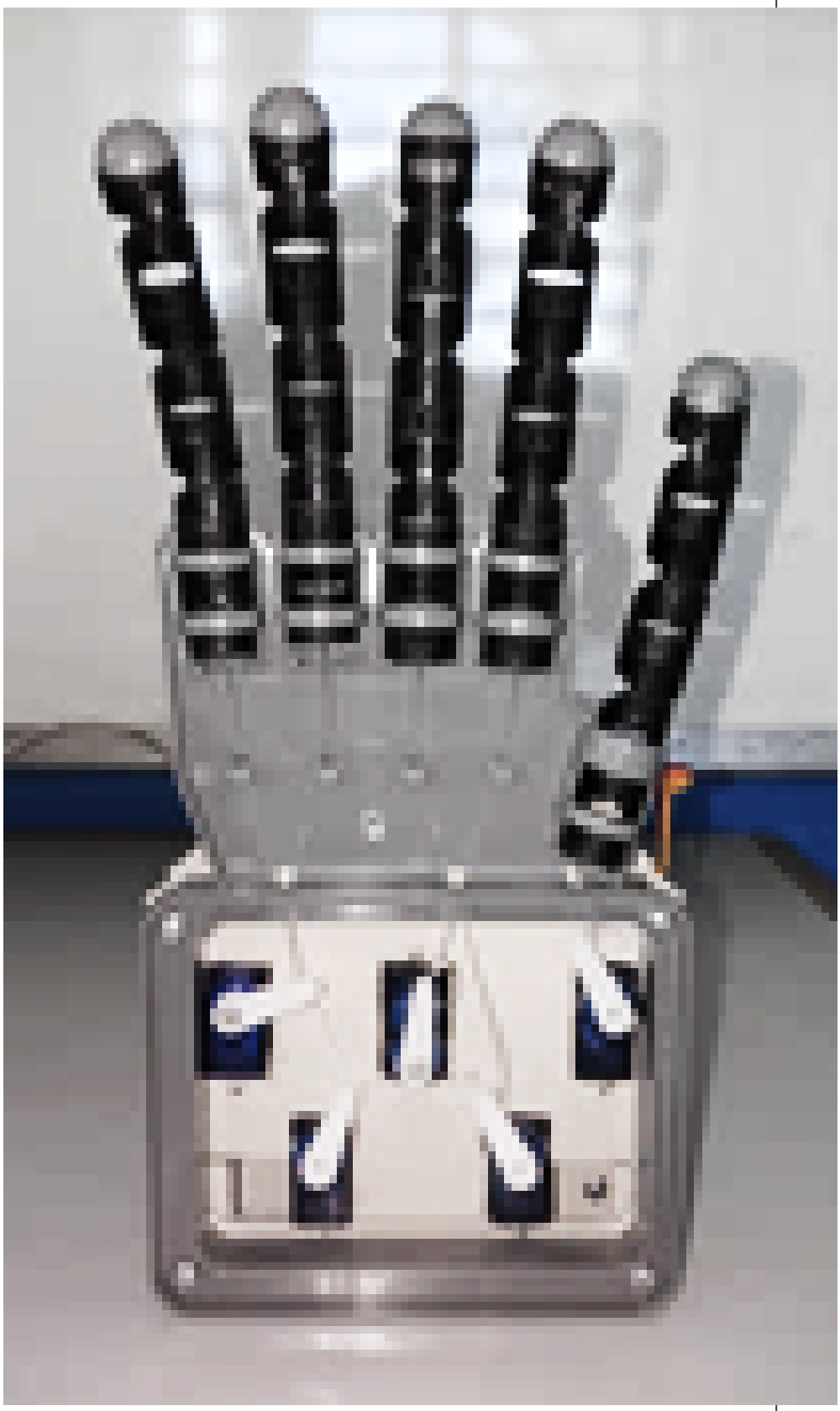

Fig. 5

Mão robótica experimental

O esquema da Fig. 6 mostra o circuito completo da mão, com o sensor mioelétrico conectado ao canal analógico 0 da placa Arduino, o PWM emite para os cinco servos que atuam nos dedos da mão e a conexão USB ao notebook para testar os algoritmos.

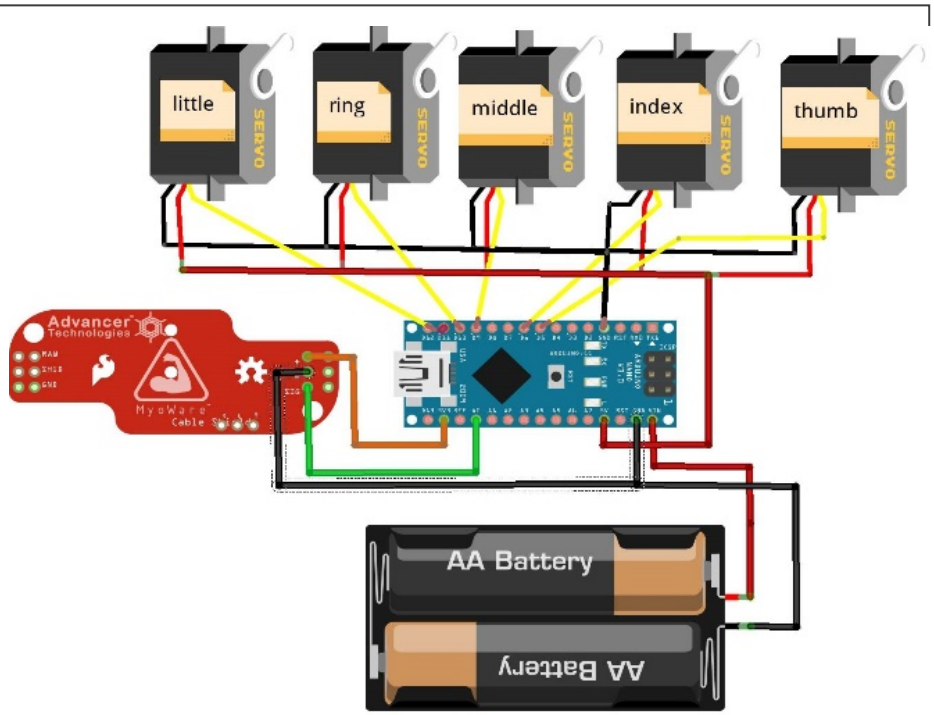

Fig. 6

Circuito completo experimental

As funções da biblioteca do Arduino que permitem operar os servos para o movimento dos dedos estão incluídas no *servo.h* e as instruções para configurá-los são apresentadas no Algoritmo 1 .

Algorithm 1 Servos configuration

```
#include <Servo.h>
Servo little; // create servo object to control a
    finger
Servo ring;
Servo middle;
Servo index;
Servo thumb;
```

Configuração de Servos do Algoritmo 1

As funções usadas para produzir os movimentos dos dedos são aquelas que permitem programar as saídas PWM com o ângulo requerido, a instrução little.write (150) é um exemplo de código para o dedo mindinho e um ângulo de 150º:

A amplitude de movimento de um servo cobre de 0º a 180º, com esses valores é obtido o deslocamento máximo. Neste trabalho, uma faixa de 130º, entre 20 e 150, é suficiente para

mover os dedos. A Tabela 5 mostra os valores em graus com os quais cada um dos servos está programado para reproduzir as posições c, p, f.

Quadro 5
Programação de servos

	pouco	anel	meio	índice	polegar
c	20	20	20	20	150
p	150	150	150	20	150
f	15	150	150	150	20

6. conclusões

Três classes de movimentos foram classificadas usando SVM e Redes Neurais. Os resultados desses testes mostraram que o SVM é o mais eficiente sem a necessidade de grandes quantidades de amostras para treinar o classificador. Esse aspecto é relevante porque consiste em um procedimento que não leva tempo para distrair ou cansar o usuário.

A classificação desenvolvida foi bem-sucedida, apresentando aproximadamente 90% de sucesso, considerando que emprega um treinamento simples e um design econômico e simples, constituído por um sensor e um módulo de microcontrolador de baixo custo, tamanho e consumo. O algoritmo de processamento apresenta características de complexidade, tamanho e velocidade que permitiriam sua adaptação à linguagem C do ambiente Arduino. Dessa forma, em um módulo Nano, as etapas de aquisição, processamento e controle que conferem à prótese as características de autonomia de um dispositivo portátil podem ser integradas. Esse é o objetivo da próxima etapa do projeto.

Por fim, os aspectos desta proposta de design, inicialmente orientados para uma prótese de mão, fazem com que ela ofereça soluções acessíveis e extensíveis para abordar um amplo espectro de deficiências.

Referências

[1] B. Hudgins, P. Parker e RN Scott, "Uma nova estratégia para controle mioelétrico multifuncional", *IEEE Transactions on Biomedical Engineering* , vol. 40, pp. 82-94, 1993.

[2] J. Brazeiro, S. Petraccia, M.Valdés. *Mano controlado por musculares* . Tese de bacharelado, Universidade da

República, 2015.

[3] RG Clement, KE Bugler, CW Oliver. “Mãos protéticas biônicas: uma revisão da tecnologia atual e aspirações futuras”. *O cirurgião* . vol. 9, pp. 336-340, 2011.

[4] AE Schultz, TA Kuiken. “Interfaces neurais para controle de próteses de membros superiores - o estado da arte e as possibilidades futuras”. *Academia Americana de Medicina Física e Reabilitação* . vol.3, pp. 55-67,2011.

[5] M. Hakonena, H. Piitulainenb, A. Visala. “Estado atual do processamento de sinal digital em interfaces mioelétricas e aplicações relacionadas”. *Processamento e controle biomédico de sinais* . vol. 18, pp. 334-359, 2015.

[6] MA Oskoei, H. Huosheng. "Sistemas de controle mioelétrico uma pesquisa". *Processamento e controle biomédico de sinais* . Vol.2, pp. 275-294, 2011.

[7] A. Phinyomark, S. Hirunviriya, C. Limsakul e P. Phukpattaranont, “Avaliação da extração de feições emg para reconhecimento de movimento das mãos com base na distância euclidiana e desvio padrão”, *Conferência Internacional de Engenharia Elétrica / Telecomunicações e Informações em Computação Eletrônica Tecnologia (ECTI-CON)* , pp. 856–860, 2010.

[8] A. Phinyomark, P. Phukpattaranont e C. Limsakul, "Redução e seleção de recursos para classificação de sinais emg", *Expert Systems with Applications* , vol. 39, n. 8, pp. 7420–7431, 2012.

[9] P. Azaripasand, A. Maleki e A. Fallah, “Classificação de adls usando forma de onda de ativação muscular versus características de treze emg”, *22ª Conferência Iraniana de Engenharia Biomédica (ICBME)* , pp. 189–193, 2015.

[10] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Van derplas, A Passos, D. Cournapeau, M. Brucher, M. Perrot e E. Duchesnay, “Scikit-learn: Machine Learning in Python”, *Journal of Machine Learning Research* , vol. 12, pp. 2825–2830, 2011.

[11] E. Jones, T. Oliphant, P. Peterson, et al., “SciPy: Ferramentas científicas de código aberto para Python.” 2001. Disponível em:

[12] V. Zschorlich, "Filtragem digital de sinais EMG", *Eletromiografia e Neurofisiologia Clínica* , vol. 29, pp. 81–86, 1989.

[13] C. Cortes, V. Vapnik, "Redes de vetores de suporte", *Machine Learning* , vol. 20, pp. 273–297, 1995.

[14] “Support vector machines.” Disponível em: http://scikit-learn.org/ stable / modules / svm.html. Acesso em 17/10/2017.

[15] “Vizinhos mais próximos”. Disponível em: http://scikit-learn.org/ stable / modules / neighbours.html. Acesso em 17/10/2017.

[16] “Modelos de redes neurais (supervisionadas).” Disponível em: http://scikit-learn.org/ stable / modules / neural_ networks_supervised.html. Acesso em 17/10/2017.

[17] Prótesis Mano Robótica. Disponível em: https://www.youtube.com/watch?v=-85fNRu4yp8&feature=youtu.be