Sistema Embarcado de Controle de Prótese de Mão

Trabalho de Conclusão de Curso

Engenharia da Computação

Itúrbide Felipe Agostinho dos Santos
Orientador: Diego José Rátiva Millán

Universidade de Pernambuco
Escola Politécnica de Pernambuco
Graduação em Engenharia de Computação

Itúrbide Felipe Agostinho dos Santos

Sistema Embarcado de Controle de Prótese de Mão

Monografia apresentada como requisito parcial para obtenção do diploma de Bacharel em Engenharia de Computação pela Escola Politécnica de Pernambuco – Universidade de Pernambuco.

Recife, junho de 2015.

De acordo

Recife

____/_____/_____

_____
Diego José Rátiva Millán

A minha mãe, pela paciência nesses árduos meses de trabalho, aos meus amigos, por não me julgarem neste período de ausência, ao meu pai, de onde quer que ele esteja me sinto orgulhoso de ser seu filho, e também a todos aqueles que possam vir a precisar deste trabalho.

"Tudo é impossível, até que seja feito" – Nelson Mandela.

Agradecimentos

Agradeço primeiramente a minha amiga e parceira Maria Clara, por ter caminhado toda essa trajetória comigo sempre me ajudando e se esforçando e me surpreendendo e servindo de exemplo e inspiração para mim. Sempre vi em você um grande potencial e fico estarrecido quando vejo o quanto você aprendeu nesses últimos meses. Obrigado por me deixar fazer parte dessa parte da sua vida.

Agradeço a minha mãe, heroína que me deu apoio, incentivo nas horas difíceis e que sempre apostou toda sua fé em mim. Espero ter superado suas espectativas e sempre lhe deixar orgulhosa pois você é a pessoa mais especial da minha vida.

Agradeço ao meu pai, pelo infinito apoio e compreensão de sua parte. Apesar de não poder o fazer pessoalmente, sempre estará em meu coração.

Ao meu orientador Diego Rátiva, pela orientação, apoio, confiança, por todo esse tempo de trabalho juntos, pelos conselhos, e por ter me proporcionado o conhecimento não apenas racional, mas o desenvolvimento do meu caráter e a paixão por novos conhecimentos.

Obrigado aos professores Sérgio Campello e Diego Rátiva pelo esforço imensurável dedicado a apresentação deste trabalho e a vida de um aluno eternamente grato.

E por fim à todos os meus mentores que diretamente ou indiretamente me trouxeram até onde estou e tanto se dedicaram a mim. E obrigado por me fazer entender o real significado e poder da palavra Professor.

Resumo

No Brasil, estima-se que mais de 7000 pessoas sofrem algum tipo de amputação a cada ano. A partir da necessidade de reaver os movimentos perdidos, essas pessoas podem optar por adquirir próteses, que podem ser classificadas em ativas ou passivas. Para as próteses ativas existe a necessidade de um sistema de controle, e quando efetuado com eletromiografia (EMG) de superfície, existe um problema de classificação do sinal por ser linearmente inseparável. Dessa forma, as redes neurais artificiais (RNA) são bastante utilizadas para a resolução do problema, e, consequentemente, para o controle da prótese. Portanto, o objetivo deste trabalho consiste em construir um sistema capaz de controlar uma prótese de mão a partir de sinais de EMG provenientes de sensores de superfície, utilizando uma rede neural MLP como classificador na plataforma Arduino. Para isto, utilizou-se um kit Muscle Sensor, responsável pela aquisição e pré-processamento dos sinais EMG, que foram injetados em uma rede neural desenvolvida no matlab. Os parâmetros provenientes do treino da rede serviram uma fase *foward* de outra rede neural implementada no arduino para que, juntamente com um Motor Shield, pudessem controlar o motor da prótese. O sistema foi capaz de classificar sinais de EMG e realizar o controle de um motor com um grau de liberdade, ou seja, um movimento de pinça.

Abstract

In Brazil, it is estimated that more than 7,000 people suffer some kind of amputation each year. From the need to recover the lost movements, these people can choose to purchase protheses, which can be classified as active or passive. For the active ones a control system is needed, and when made with surface electromyography (EMG), there is a signal classification problem to be linearly inseparable. Thus, the artificial neural networks (ANN) are widely used to solve the problem, and consequently to control the prosthesis. Therefore, the objective of this work is to build a system capable of controlling a prosthetic hand system from EMG signals with surface sensors, using a neural network classifier MLP as the Arduino platform. For this, we used a Muscle kit sensor responsible for the acquisition and pre-processing of the EMG signals, which were injected into a neural network designed in matlab. The parameters from the network training served a foward phase of another neural network implemented in Arduino so that, together with a motor shield, could control the motor of the prosthesis. The system was able to classify signals EMG and performing control of a motor with one degree of freedom, a pincer movement.

Sumário

Indice de Figuras

Índice de Tabelas

Tabela de Símbolos e Siglas

DC – *Direct Current* (Corrente contínua);

EMG – Eletromiografia;

$f(net)$ – Função de Ativação;

GMDH - *Group Method of Data Handling;*

h - Número de neurônios na camada escondida;

Hz – Hertz;

MLP – *Multi-layer perceptron;*

net – Entrada líquida da rede neural;

PWM – *Pulse Width Modulation;*

RNA – Redes neurais artificiais

$\mathrm{sech}()$ – Secante hiperbólica;

$tagh()$ – Tangente hiperbólica;

UM – Unidade motora;

V – Volts;

W – Matriz de pesos;

α – Coeficiente de aprendizado / taxa de aprendizado;

Ω – Ohms;

δ_i^m – Sensibilidade;

ε – Notação matricial da sensibilidade;

Capítulo 1

Introdução

A mão possui um papel fundamental no desempenho das atividades diárias do ser humano, como por exemplo, escrever, agarrar objetos e tocar instrumentos. Perdê-la impacta diretamente na expressividade do individuo, sendo assim, alterações físicas e psicológicas precisam ser trabalhadas com o objetivo de minimizar o sofrimento do paciente (1). As próteses de mão, então, podem ser uma das soluções para estes problemas.

A amputação pode ter várias causas, dentre elas, congênita, traumática e crônica. Estima-se que no Brasil a incidência de amputações seja de 13,9 por 100.000 habitantes/ano (2). Contudo existe uma subnotificação desses dados que impossibilitam um número mais fidedigno. Um estudo quantitativo realizado em 2006 demonstrou que de todos os traumas 27,6% corresponde a traumas de mão e dedo, desses, 11,5% são diagnosticados com amputação total ou parcial da mão. (3)

Próteses de mão podem ser classificadas em ativas e passivas. As passivas também são conhecidas como estéticas e não possuem nenhuma movimentação. Já as próteses ativas têm o objetivo de recuperar algumas das funções do membro perdido. Esse tipo de prótese pode ser controlada de várias formas, sendo as mais comuns as que são ativadas através de tirantes e de sinais eletromiográficos (próteses mioelétricas). Com próteses ativas, o amputado consegue uma maior independência nas suas atividades diárias, aumentando, assim, sua qualidade de vida (4).

A maior parte dos movimentos da mão são realizados pela cooperação de diversos músculos. Os sinais eletromiográficos (EMG) provenientes de eletrodos de superfície são muito sensíveis a *crosstalking* de sinais de vários músculos que estão em torno do eletrodo, e também diferem muito entre pessoas com capacidades musculares diferentes, o que complica a tarefa de separar e classificar os sinais de EMG derivados dos músculos.

Para realizar essa tarefa, vários estudos já aplicaram mecanismos de aprendizado como discriminância quadrática (5), discriminância linear (5), *K-Nearest Neighborhood* (knn) (5) e *Supported Vector Machine* (SVM) (6). Porém esses mecanismos ainda possuem dificuldades de convergência e pouca capacidade de classificação.

A utilização de diversas metodologias de aprendizado computacional é cada vez mais frequente (7) (8) (9) (10) e dentro delas podemos destacar as redes neurais Multi-Layer Perceptron (MLP) que após treinadas conseguem realizar classificações em tempo real (10).

Dessa forma, a proposta deste trabalho é construir um sistema capaz de controlar uma prótese de mão a partir de sinais de EMG provenientes de sensores de superfície, utilizando uma rede neural MLP como classificador na plataforma Arduino. Ele será parte de um trabalho, na Universidade de Pernambuco, cujo objetivo é desenvolver uma prótese completa funcional, que possa substituir uma mão. É importante ressaltar que esse sistema foi aplicado numa primeira versão da prótese capaz de realizar apenas o movimento de pinça. Portanto, um outro objetivo deste trabalho é desenvolver um sistema que possa vir a ser utilizado com próteses que possuem outras funcionalidades.

No Brasil, os trabalhos desenvolvidos sobre prótese de mão ainda são bem incipientes. Um dos trabalhos mais significativos, principalmente por ser pioneiro em sua época, foi a mão de São Carlos. Nesse trabalho, existe a integração da prótese com sensores de força e temperatura a fim de possibilitar uma retroalimentação do sistema (4). Um outro projeto desenvolvido na Universidade Estadual de Campinas, tinha como função projetar e construir uma prótese de mão com a premissa de simplificar a concepção do projeto em si. O pesquisador utilizou engrenagens para fazer a movimentação de quatro dedos em oposição ao polegar (11).

Outros trabalhos foram desenvolvidos em Porto Alegre (12), Rio de Janeiro (13) já direcionados a um controle por rede neural e a utilização de sensores de força, tátil e de temperatura a fim de melhorar o desempenho do protótipo.

Capítulo 2

EMG

A eletromiografia é o estudo da função muscular através da averiguação do sinal elétrico que emana do músculo (14) (15). Os primeiros experimentos com EMG foram realizados por Francesco Redi em 1666 e sua utilização em tratamentos clínicos começou em 1960 (16), por proporcionar avaliações capazes de determinar características elétricas de um músculo ou grupo muscular (17).

O sistema neuromuscular (18) é composto por células musculares, neurônios, tecido conjuntivo, tecido ósseo e vasos sanguíneos e é organizado em estruturas funcionais básicas chamadas de unidades motoras. Essas unidades motoras (UM) são responsáveis por gerar tensão mecânica, contraindo os músculos. Como ilustrado na Figura 1, as UM são constituídas por um motoneurônio-alfa e todas as fibras inervadas por ele, e, para que seja gerada uma tensão mecânica, é necessário que um potêncial de ação elétrica seja gerada pelo sistema nervoso central, e transmitida pelo motoneurônio até as fibras musculares.

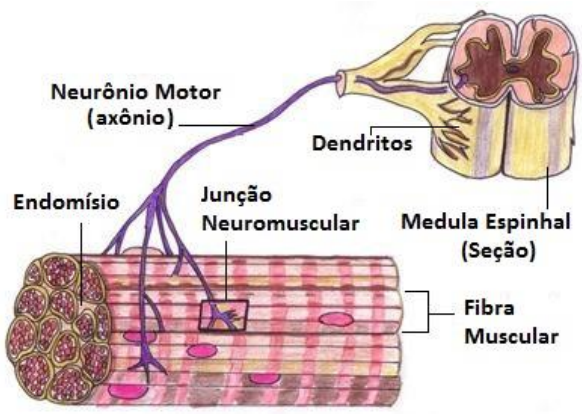

Figura 1. Sistema Neuromuscular.

De Acordo com Burke (19) a capacidade de produção de força das unidades motoras varia amplamente e é dependente das suas características fisiológicas, bioquímicas e anatômicas. Logo pode-se concluir que o dois indivíduos que realizem

o mesmo movimento quando submetidos à um exame de EMG, muito provavelmente terão resultados distintos.

Atualmente, a eletromiografia de superfície vem sendo utilizada em diversas áreas de pesquisas e clínicas como neurofisiologia, ciências do esporte e reabilitação, por ser um método de avaliação neuromuscular não invasivo (20). Segundo Woods, J. J. et. al. (21) é possível relacionar este exame com a quantidade de força exercida por um músculo. Já Roy, S. et. al. (14) disse que essa ferramenta possui diversas limitações que precisam ser consideradas e eventualmente removidas.

Os principais problemas associados aos eletrodos de superfície são: a atenuação causada pelo tecido subcutâneo e a possível contaminação do sinal por interferências de sinais provenientes de outros grupos musculares próximos (*crosstalking)* (22).

De modo geral, os eletrodos estão sempre sujeitos à interferências elétromagnéticas de fontes externas, pois o corpo humano é um bom condutor, agindo como uma antena quando próximo de redes elétricas ou motores elétricos. Para eliminar tal problema utiliza-se a configuração bipolar de eletrodos (22), ou seja, dois eletrodos submetidos a um amplificador diferencial, como ilustrado na Figura 2. Desta forma, ambos os eletrodos estarão sujeitos a mesma interferência eletromagnética, ou seja, um sinal comum, que será cancelado pelo amplificador diferencial.

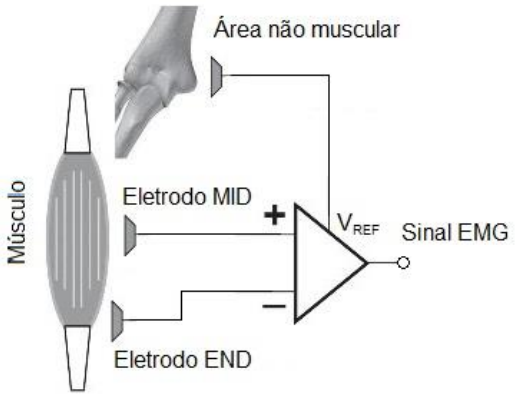

Figura 2. Eletromiografia superficial bipolar. O Eletrodo MID é colocado na parte ventral do músculo, quanto o END na extremidade.

O sinal eletromiográfico dos músculos, quando captado por eletrodos de superfície, possui uma faixa entre 0 e 500 Hz, sendo o intervalo entre 50 e 150 Hz, o de maior energia (23). Quanto à amplitude do sinal, segundo Sodeberg e Cook (24), o valor máximo que um músculo normal consegue atingir é de apenas 3 mV.

Capítulo 3

Redes Neurais Artificiais

O cérebro é um sistema de processamento de informação altamente complexo, não linear e com alto poder de paralelismo. Ele diverge dos computadores atuais nestes aspectos e possui a capacidade de organizar os seus componentes estruturais, conhecidos como neurônios (25). Desta forma, o cérebro é uma ferramenta poderosa em várias tarefas, como reconhecimento de padrões, percepção e controle motor, e é muito mais rápido que o computador mais poderoso da atualidade. (26) (27).

O aprendizado do cérebro é realizado de acordo com as suas “experiências”. Nos primeiros momentos da vida, um cérebro já possui uma boa estrutura e tem a capacidade de se desenvolver e construir suas próprias regras, e conexões neurais são construídas e destruídas durante toda a vida do cérebro (26).

Com o intuito de modelar essa poderosa ferramenta da natureza, McCulloch e Pitts propuseram as redes neurais artificais (RNA) (28). As RNA seriam capazes de ter plasticidade, ou seja, seriam capazes de se adaptar ao ambiente em que estivessem submetidas para uma tarefa ou função específica. Segundo Haykin: “*Uma Rede Neural Artificial é um processador paralelo distribuído em massa feito de unidades simples de processamento (neurônios), que possuem a habilidade natural de armazenar conhecimento a partir de experiências tornando-os utilizáveis. Elas parecem com o cérebro de duas maneiras: 1 – o conhecimento é obtido pela rede através do ambiente por um processo de aprendizado; 2 – os pesos das conexões interneurais, conhecidos como pesos sinápticos, são utilizados para armazenar o conhecimento obtido*”.

A primeira regra de aprendizado das RNA foi proposta por Donald Hebb (29) que é a base de todas as regras de aprendizagem subsequentes. Segundo Hebb: *“Quando um Neurônio recebe um estímulo de outro neurônio, e se ambos estão altamente ativos, o peso entre estes deve ser fortalecido, caso contrario enfraquecido"*.

3.1 O Neurônio Natural e o Neurônio Artificial

O neurônio natural funciona baseado na Lei do Tudo ou Nada, onde, dependendo do nível de estímulo de entrada, e de seu limiar excitatório, ele é capaz de ativar ou desativar, disparando ou não um impulso nervoso. Se este limiar estiver acima do seu estímulo de entrada, o neurônio não emana nenhum impulso nervoso, caso contrário, o potencial de ação terá uma intensidade padrão, sempre que ativar.

Para modelar matemáticamente tal comportamento, foram criadas entidades chamadas de neurônios artificiais (28). Estes neurônios utilizam de uma regra de propagação e uma função de ativação que são análogos a Lei do Tudo ou Nada.

3.1.1 Propagação do impulso

A regra de propagação é dada por

$$net_i = \sum_{j=1}^{n} w_{ij} x_j - \theta. \qquad (1)$$

Onde x_j são variáveis de entrada, net_i é a entrada líquida do neurônio, w_{ij} são os pesos sinápticos que guardam o aprendizado do sistema, e θ é o limiar de ativação do neurônio. Essa estrutura pode ser descrita como na Figura 3.

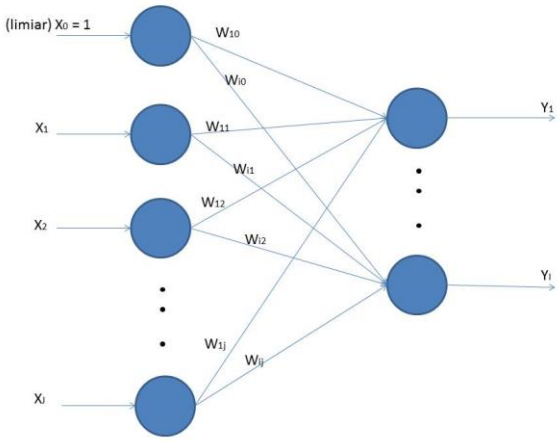

Figura 3. Exemplo de estrutura de uma RNA.

Para facilitar a modelagem matemática, a representação do limiar de forma implícita é tomada, desta forma o primeiro neurônio reflete sua função, guardando seu valor na conexão sináptica w_{10}, entre ele e o neurônio de saída

A função de ativação aplicada à entrada líquida dos neurônios é chamada de $f(net_i)$ ou y_i, e pode ser considerada a função degrau, representando a Lei do Tudo ou Nada, como dito anteriormente, ou outras funções como a tangente hiperbólica, sigmóide logística entre outras (30).Esta função vai caracterizar a saída do neurônio, e deve ser escolhida dependendo da natureza do problema. A função degrau tem a característica de assumir apenas dois valores, 0 ou 1; a sigmóide logística pode assumir qualquer valor entre 0 e 1 e a tangente hiperbólica , qualquer valor entre -1 e 1.

A notação matricial foi adotada neste trabalho, pois a RNA foi implementada na ferramenta Matlab, de natureza matricial, facilitando assim a representação desta entidade na linguagem utilizada pela ferramenta. Desta forma, considerando que as variáveis de entradas são representadas pelo vetor $\vec{x}$, os pesos sinápticos são representados pela matriz W e a entrada líquida dos neurônios são representadas pelas colunas do vetor $\overrightarrow{net}$, então

$$\overrightarrow{net} = W\vec{x}. \tag{2}$$

3.1.2 Análise Dimensional da regra de propagação

Sendo $\overrightarrow{net}$ e $\vec{x}$ vetores coluna i x 1 e j x 1, respectivamente, onde j é a quantidade de variáveis de entrada e i é a quantidade de neurônios na camada de saída e W sendo uma matriz i x j, utilizando a Equação (25), do Anexo A, temos que

$$(W\vec{x})_{i1} = \sum_{i=0}^{n} w_{ij}x_{j1} = \overrightarrow{net}. \tag{3}$$

Desta forma, pode-se afirmar que, utilizando o limiar implícito, por j ter um intervalo de 0 à n, é possível definir a regra de propagação como sendo a produto matricial entre a matriz dos pesos sinápticos W e o vetor de variáveis de entrada $\vec{x}$.

3.1.3 Aprendizado da Rede Neural Artificial

Como definido por Hebb: “Quando um Neurônio recebe um estímulo de outro neurônio, e se ambos estão altamente ativos, o peso entre estes deve ser fortalecido, caso contrario enfraquecido". Baseando-se em tal afirmação pode-se concluir que os pesos sinápticos guardam todo o aprendizado da rede, alterando seus valores em função dos exemplos de entrada que são repetidamente apresentados à ela, transformando tais exemplos no que conhecemos como experiência (25). Os principais tipos de aprendizado são:

1. Aprendizado supervisionado: todos os exemplos apresentados a rede, enquanto ela estiver na fase de aprendizado (treinamento da rede), têm seu resultado previamente conhecido e os valores destes resultados são utilizados para cálculos dentro do processo de aprendizado. As redes Perceptron, Multi-Layer Perceptron (MLP), GMDH (31) e NSRBN (32) utilizam este tipo de aprendizado.
2. Aprendizado não-supervisionado: Neste tipo de rede neural, os resultados dos exemplos não são apresentados à rede, como a rede de Kohonen (33).

Neste trabalho vamos nos concentrar em redes neurais supervisionadas, pois pode-se realizar uma prévia coleta de dados dos usuários da prótese de mão.

Um método simples de treinamento de uma rede neural pode ser observada na Figura 4 que descreve um simples algoritmo o aprendizado de uma rede Perceptron.

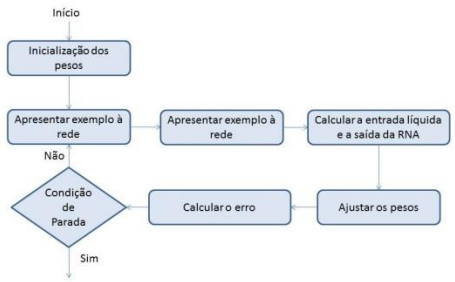

Figura 4. Exemplo de Algoritmo de Treinamento de uma Rede Neural

A equação da atualização dos pesos das conexões sinápticas entre os neurônios é dada por

$$w_{ij}(n+1) = w_{ij}(n) + \alpha \cdot (d_i - y_i) \cdot x_j. \quad (4)$$

Onde $w_{ij}(n+1)$ é o novo peso, $w_{ij}(n)$ é o peso anterior, α é a taxa de aprendizado, que indica o tamanho da variação entre os pesos novos e antigos, d_i e y_i reprensentam a saída desejada e a saída calculada do exemplo apresentado, respectivamente, e x_j é a variável de entrada conectada a esse peso. A diferença entre as saídas desejadas e as saídas calculadas representam o erro do neurônio de saída i no instante de tempo n.

3.2 Vantagens das Redes Neurais Artificiais

As redes neurais são poderosas ferramentas computacionais que possuem a habilidade de generalizar a partir de seu aprendizado. A generalização se refere à possibilidade de prover respostas à entradas que não foram apresentadas em seu treinamento, possibilitando trazer bons resultado diante de uma taxa de erros aceitáveis de acordo com o problema (25).

As redes neurais possuem as seguintes características:

- Não-linearidade: Uma RNA pode ser linear ou não, dependendo de sua arquitetura, com isso, ela é capaz de modelar um problema não linear.
- Mapeamento de entrada/saída: A rede neural é capaz de aprender com exemplos (aprendizado supervisionado).
- Adaptatividade: elas possuem a habilidade de adaptar os seus pesos sinapticos de acordo com o ambiente (exemplos apresentados a RNA).
- Indicativo de certeza: No contexto de classificação de padrões, uma rede neural pode fornecer informações sobre o quão confiável é aquela classificação feita por ela.
- Tolerância a falhas: Se uma RNA for implementada em *hardware* e uma falha ocorrer, apenas um pouco de seu desempenho será

degradado, pois a RNA guarda suas informações espalhadas em toda a rede. Seria necessário um grande dano para degradar completamente uma rede neural.

- Analogia Neurobiológica: A arquitetura das RNA são baseadas no cérebro, seus neurônios e suas conexões, sendo este a prova de que é possível realizar um processamento paralelo massivo rápido e poderoso.

3.3 Multi-Layer Perceptron

Segundo Hakin, S. (25): "*Normalmente, uma rede neural se constitui de unidades sensoriais (camada de entrada do sistema), uma ou mais camadas escondidas, e uma camada de saída*". Esta é uma generalização da rede perceptron, com a adição de pelo menos uma camada intermediária, chamada de camada escondida.

Com esta modificação, as redes neurais são capazes de resolver problemas não linearmente separáveis (30). Essas camadas escondidas normalmente possuem sigmóides logisticas ou tangentes hiperbólicas como funções de ativação (30). O comportamento das funções é ilutradas na Figura 5.

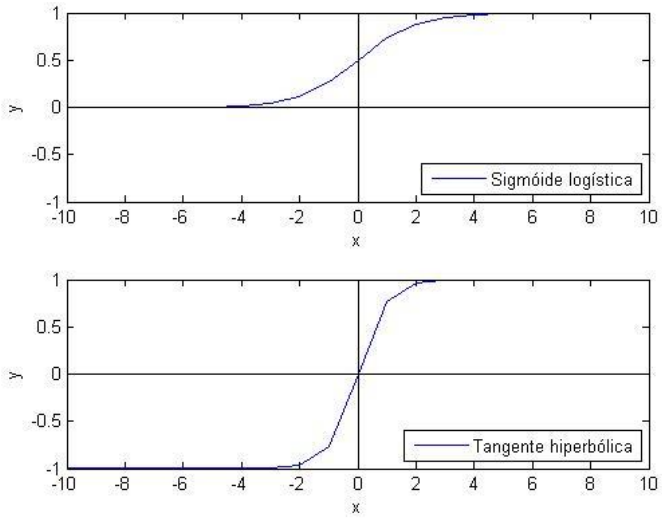

Figura 5. Comportamento das funções Sigmóide Logística e Tangente Hiperbólica.

A função de ativação de um neurônio pode ser dada, matricialmente pela aplicação sua aplicação a todos os elementos do vetor $\overrightarrow{net}$. Por exemplo, a tangente hiperbólica, definida por

$$\vec{y} = tanh(\overrightarrow{net}). \quad (5)$$

Onde $\vec{y}$ é um vetor que representa o sinal emitido pelo neurônio, de acordo com seu sinal de entrada.

3.3.1 Backpropagation

Para realiazar o treinamento, se faz necessário uma nova técnica de aprendizado, visto que a adição da camada escondida implica no desconhecimento do erro de cada um dos neurônios desta camada, assim este erro é apenas calculado na saída.

Nas MLPs, o backpropagation é muito popular (25). e consiste numa fase de propagação do erro no sentido oposto ao da rede neural, ou seja, da camada de saída até a camada de entrada, podendo assim otimizar os pesos sinápticos das conexões intraneurais.

A equação para reajustes do peso de uma MLP (32) é

$$w_{ij}^{m}(novo) = w_{ij}^{m}(antigo) + \alpha\delta_{i}^{m} f^{m-1}(net_{j}^{m-1}) . \quad (6)$$

Sendo w_{ij}^{m} o peso sináptico a ser ajustado, $f^{m-1}(net_{j}^{m-1})$ o sinal de entrada emitido pelo neurônio da camada anterior e δ_{i}^{m} é a sensibilidade, que representa, análogamente, o erro do neurônio e pode ser calculado por

$$\delta_{i}^{m-1} = \dot{f^{m-1}}(net_{j}^{m-1}) \sum_{i=1}^{n} w_{ij}^{m}\delta_{i}^{m} . \quad (7)$$

Onde $\dot{f^{m-1}}\left(net_j^{m-1}\right)$ é a derivada da função de ativação dos neurônios da camada que emite o sinal (m-1). Para calcular a sensibilidade da camada de saída

$$\delta_i^{m-1} = \dot{f^{m-1}}(net_j^{m-1})(d_i - y_i^{saída}). \quad (8)$$

Como dito anteriormente, foi utilizada a ferramenta Matlab para implementação da rede neural, sendo assim foi adotada a seguinte notação matricial para o cálculo das sensibilidades da camada escondida e da camada de saída respectivamente:

$$\varepsilon^{saída} = \left(\mathrm{sech}\left(Y^{saída}\right)\right) \circ E, \quad (9)$$

$$\varepsilon^{m} = (\mathrm{sech}(Y^{m})) \circ (W^{m+1} \cdot \varepsilon^{m+1}) . \quad (10)$$

A secante hiperbólica nessas equações representa a derivada da tangente hiperbólica, que foi a função de ativação escolhida para a rede neural e a multiplicação definida pelo operador ∘ é o produto de Hadamard (34) definido na Equação (27), no Anexo A.

Na camada de saída, multiplica-se o resultado pela matriz de erro, definida por

$$E = D - Y. \quad (11)$$

Onde D é a matriz dos valores desejados da saída e Y é a resposta do sistema aos impulsos de entrada. Já nas camadas escondidas é necessário multiplicar os pesos entre a camada *m* e a camada *m+1* pelas sensibilidades da camada *m+1*.

Após o cálculo das sensibilidades, podemos enfim realizar o reajuste dos pesos, promovendo, desta forma, o aprendizado da rede neural. A equação matricial que define esta operação é

$$W_{novo}^{m} = W_{antigo}^{m} + \alpha \cdot \varepsilon^{m} \cdot (Y^{m-1})^{T}. \quad (12)$$

Onde α é a coeficiente de aprendizado da rede neural, ε^{m} é um vetor que representa as sensibilidades dos neurônios da camada m e Y^{m-1} é também um vetor que possui os valores de entrada do neurônio, ou seja, os valores de saída dos neurônios da camada anterior.

Capítulo 4

Arduino e Kit de EMG

4.1 Arduino

O dispositivo consiste em uma plataforma eletrônica *open-source* produzida com o intuito de facilitar seu uso e de otimizar implementações de *hardwares* e *softwares*.

Existem várias versões do Arduino disponíveis no mercado, como Uno, Leonardo, Due, Yún, Mega, Nano, entre outras (35). Neste trabalho será utilizado o Arduino Uno, Figura 6, pois utilizaremos a placa Motor Shield V3 que será conectada ao Arduino de maneira a expandir suas funções. As características do Arduino Uno estão listadas na Tabela 1.

Figura 6. Arduino Uno.

Tabela 1. Características do Arduíno Uno

Microcontrolador	Atmega328
Tensão de operação	5V
Pinos de Entrada e Saída Digitais	8 + 4 PWM

Pinos Analógicos	6
Memória Flash	31.5 KB + 0.5 *bootloader*
SRAM	2 KB
Conversor Digital / Analógico	10 bits / 10 kHz

É importante ressaltar que o tempo mínimo entre leituras da porta analógica é de 1 microssegundo, devido a frequência de amostragem de 10 kHz do conversor digital/analógico. Como citado anteriormente, o sinal eletromiográfico, quando captado por eletrodos de superfícies, possui uma frequências entre 0 e 500 Hz, sendo o intervalo entre 50 e 150 Hz, o de maior energia (23), sendo assim, o arduino respeita ao Teorema de Nyquist, que determina que para um sinal não possuir *aliasing* e não perder informação útil, a taxa de amostragem precisa ser de no mínimo o dobro da frequência máxima do sinal.

4.2 Motor Shield v3

É uma extensão para o Arduino baseada no chip L298 (36), que possui duas ponte-h duplas, que permitem controlar relays, solenóides, motor de passo e motor de corrente contínua (DC). Com este chip é possível controlar a direção e a velocidade de até dois motores DC independentemente e também medir a corrente de cada um deles.

A Figura 7 ilustra um motor shield, e nela pode-se observar os canais A e B de conexões, um para cada possível motor. Os pinos estão descritos na Tabela 2.

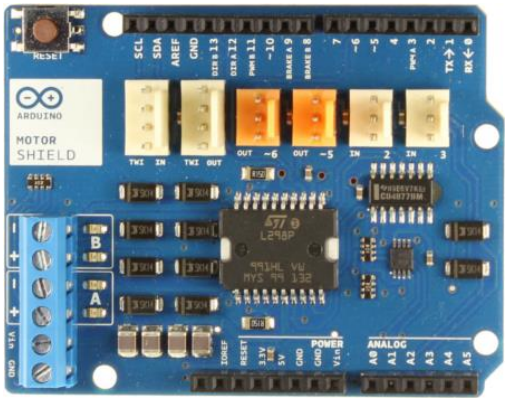

Figura 7. Motor Shield v3.

Para utilizar motores é necessário conectar uma fonte externa de até 18 V.

Tabela 2. Pinos utilizados pelo Motor Shield v3.

Função	**Pino para o Canal A**	**Pino para o Canal B**
Direção	D12	D13
Velocidade (PWM)	D3	D11
BRAKE	D9	D8
Sensor de Corrente	A0	A1

Um extenso detalhamento sobre o Motor Shield V3 pode ser encontrado em (37).

4.3 Muscle Sensor v3

Com o objetivo de medir a atividade muscular, o Muscle Sensor v3 , Figura 8, é um pequeno módulo que utiliza eletrodos superficiais para detectar o potencial elétrico gerado pela contração dos músculos, conhecido como sinais eletromiográficos (EMG).

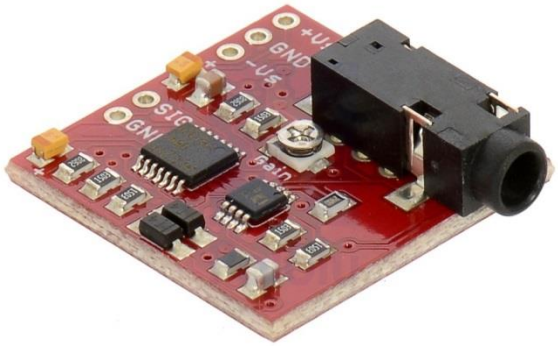

Figura 8. Muscle Sensor V3.

Como dito anteriormente, os sinais EMG adquiridos por sensores de superfície possuem algumas características:

a. Têm baixa amplitude e variam em torno de ±3mv;

b. Sua frequência está limitada entre 50Hz e 150Hz;

c. São sensíveis à interferências eletromagnéticas;

Por ser um sinal de baixa amplitude e que pode adquirir valores negativos, os microcontroladores não conseguem interpretá-lo, podendo até causar problemas no funcionamento de componentes eletrônicos.

O kit Muscle Sensor v3 é capaz de medir as atividades musculares e disponibiliza os sinais EMG amplificados, filtrados e retificados, com sua saída entre 0 e Vs Volts, onde Vs é o valor de tensão da fonte de energia do sistema (38). Dessa forma pode-se conectar este kit diretamente a um microcontrolador sem se preocupar com o comportamento dos sinais que estão sendo captados pelos eletrodos.

Podemos dividir o esquema elétrico representado na Figura 9 , em 4 partes, cada uma responsável por uma determinada operação:

1. Medição do potêncial elétrico do músculo;
2. Retificação;
3. Suavização;
4. Amplificação;

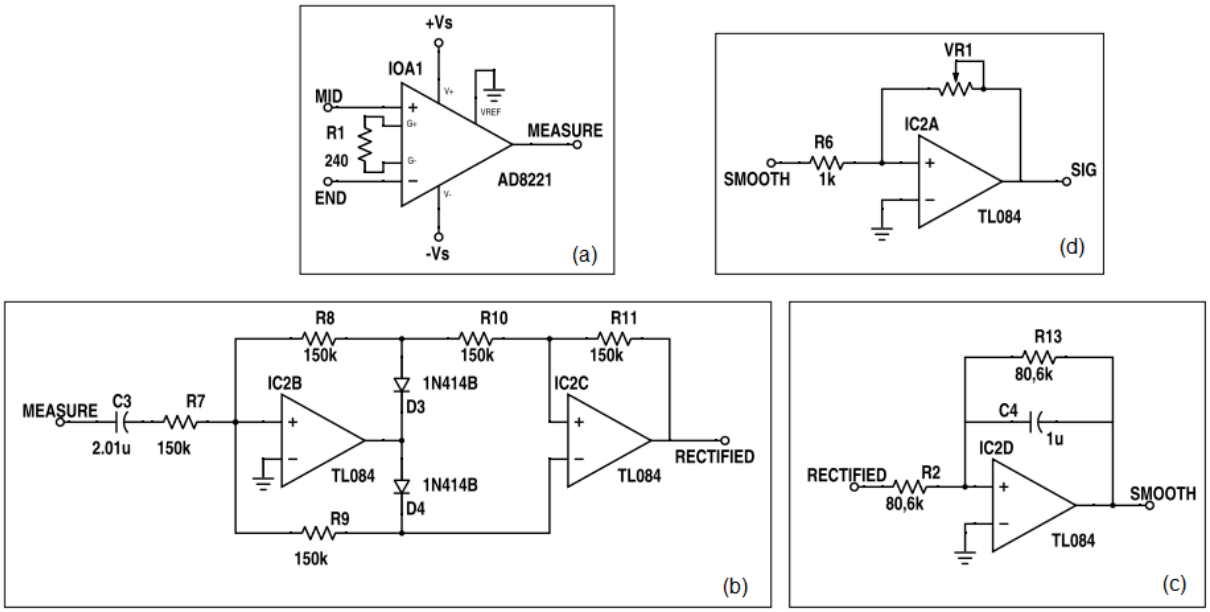

Figura 9. Esquema Elétrico do Muscle Sensor.

4.3.1 Medição do Potencial Elétrico do Músculo

Um Amplificador operacional é utilizado para computar a diferença de tensão entre os dois eletrodos que estão localizados no músculo alvo da medição. O eletrodo conetado ao terminal MID é colocado na porção ventral do músculo, enquanto o que está conectado ao terminal END é posicionado numa das extremidades do músculo. Nesta forma bipolar é possível anular a interferência eletromagnética cujo os eletrodos estão submetidos (22).

Como pode ser observado no Datasheet (39) do dispositivo utilizado (AD8221), este amplificador operacional tem um ganho configurável de acordo com o valor do resistor colocado entre suas entradas RG. De acordo com a Figura 9, o valor deste resistor é de 240 Ω. O ganho pode ser calculado pela Equação (13), disponibilizada no datasheet (39)

$$G + 1 = \frac{49{,}4\ k}{R_g}. \qquad (13)$$

aplicando o valor de 240 Ω ao resistor R_g da Equação (13), tem-se que o ganho é de aproximadamente 205. Então ele é aplicado na diferença entre os sinais dos dois eletrodos de acordo com a equação geral dos amplificadores operacionais

$$Vo = A(V^+ - V^-),$$

e, de forma análoga:

$$MEASURE = A(MID - END). \quad (14)$$

Os amplificadores operacionais têm seu limite de operação +Vs e -Vs, e o seu comportamento pode ser observado na Figura 10.

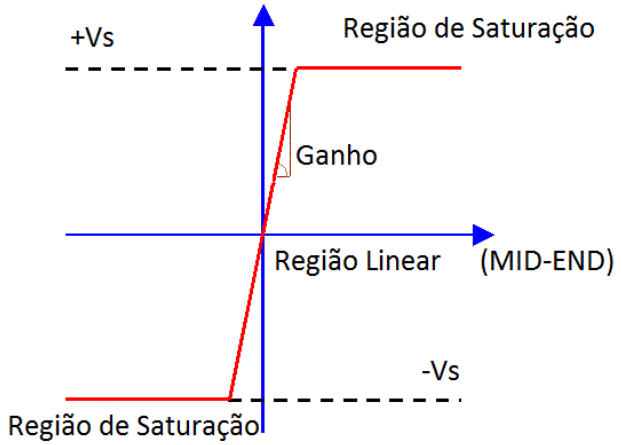

Figura 10. Funcionamento do Amplificador Operacional.

4.3.2 Amplificação

Para esta fase é utilizado um circuito capaz de amplificar o sinal, pois sua natureza é de baixa amplitude. Desta forma é utilizado um único amplificador operacional no modo inversor, que, segundo a Figura 10, tem seu comportamento definido por

$$SIG = A \cdot (SMOOTH - V^-). \quad (15)$$

Onde V^+ e V^- são tensões aplicadas nos terminais não inversor e inversor, respectivamente. Uma divisão de tensão pode ser realizada, conforme a Figura 11, com o objetivo de encontrar a tensão no terminal V^- do amp Op.

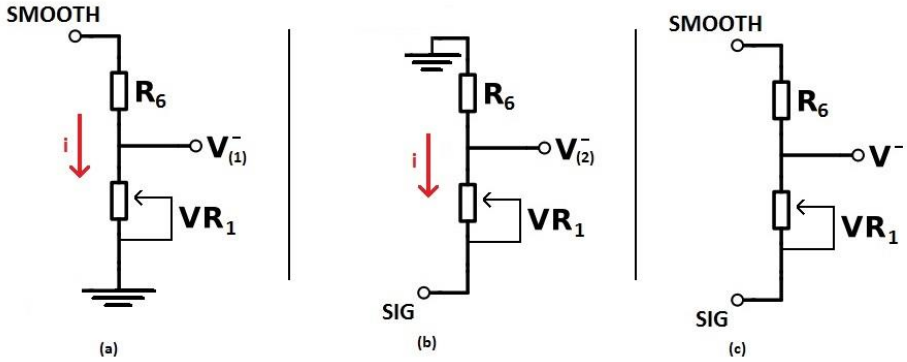

Figura 11. Análise do divisor de tensão do Amp Op.

Onde no momento 1, Figura 11 (a), a corrente i é dada por

$$i = \frac{SMOOTH}{R_6 + VR_1}. \tag{16}$$

Assim, podemos calcular $V_{(1)}^-$

$$i = \frac{SMOOTH}{R_6 + VR_1}.$$

$$V_{(1)}^- = VR_1 \cdot i\,,$$

$$V_{(1)}^- = \frac{VR_1 \cdot SMOOTH}{R_6 + VR_1}.$$

Aplica-se o mesmo processo para (b) e obtém-se que

$$V_{(2)}^- = \frac{R_6 \cdot SIG}{R_6 + VR_1}.$$

Como resistores são componentes lineares e invariantes no tempo, temos que

$$V^- = V^-_{(1)} + V^-_{(2)},$$

$$V^- = \frac{(VR_1 \cdot SMOOTH) + (R_6 \cdot SIG)}{R_6 + VR_1}.$$

Desta forma, podemos substituir na Equação (15)

$$SIG = -A \cdot \frac{(VR_1 \cdot SMOOTH) + (R_6 \cdot SIG)}{R_6 + VR_1},$$

$$(R_6 + VR_1)SIG = -A \cdot [(VR_1 \cdot SMOOTH) + (R_6 \cdot SIG)],$$

$$(R_6 + VR_1 + A \cdot R_6)SIG = -A \cdot (VR_1 \cdot SMOOTH),$$

$$SIG = \frac{-A \cdot VR_1}{(R_6 + VR_1 + A \cdot R_6)} \cdot SMOOTH,$$

$$SIG = \frac{-VR_1}{\left(R_6 + \frac{VR_1 + R_6}{A}\right)} \cdot SMOOTH.$$

Como A é, idealmente, um valor muito grande

$$SIG = \frac{-VR_1}{R_6} \cdot SMOOTH.$$

Definido H como a função transferência deste circuito, temos que

$$H = \frac{SIG}{SMOOTH},$$

$$H = \frac{-VR_1}{R_6}. \tag{17}$$

Conclui-se que, dependendo do valor do potenciômetro VR_1, o ganho do sistema pode aumentar ou diminuir, caso necessário.

4.3.3 Suavização

Para criar um sinal suavizado, eliminando problemas de flutuações muito rápidas no sinal, foi utilizado um filtro suavizador inversor ativo de primeira ordem, como pode ser observado na Figura 9c. Podemos simplificar o circuito, calculando as impedâncias de cada um dos componentes que estão ligados ao amp op e reduzir ao circuito observado na Figura 12. Com isso, teremos um circuito idêntico ao item anterior, e sabemos sua função transferência pela Equação (17).

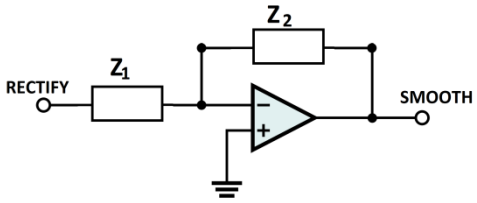

Figura 12. Simplificação do Filtro Passa-Baixa Ativo.

De forma análoga

$$H = \frac{-Z_2}{Z_1}. \tag{18}$$

A impedância Z_1 é igual a R_2, e Z_2 é a disposição em paralelo entre o resistor R_{13} e o capacitor C_4, assim podemos expandir Z_2:

$$Z_2 = \frac{R_{13} * \frac{1}{jwC_4}}{R_{13} + \frac{1}{jwC_4}},$$

$$Z_2 = \frac{R_{13}}{jwR_{13}C_4 + 1}.$$

Substituindo na equação (18), temos que

$$H = \frac{\frac{-R_{13}}{jwR_{13}C_4 + 1}}{R_2},$$

$$H = \frac{-R_{13}}{R_2} \cdot \frac{\frac{1}{R_{13}C_4}}{jw + \frac{1}{R_{13}C_4}}. \quad (19)$$

O ganho do sistema é dado pelo quociente entre as resistências, e é igual a 1. A frequencia de corte $w_c = \frac{1}{R_{13}C_4} \cong 12{,}4$ rad/s ou 1,97 Hz. Como trabalharemos com o sinal retificado, a frequência de corte precisa ser baixa com o intuito de suavizá-lo.

4.3.4 Retificação

A Figura 9 (b) representa um retificador de onda completa de precisão. Como todos os resistores possuem o mesmo valor, temos um sistema sem ganhos. O funcionamento deste retificador pode ser analisado dividindo-o em quatro partes, ilutradas nas Figura 13 a Figura 16

É importante ressaltar que a função do capacitor na entrada do circuito é filtrar as frequências baixas do sinal, removendo o deslocamento dele, também chamado de nível DC.

Considerando o sinal de entrada positivo em relação ao terra, a Figura 13 mostra que, o diodo D3 estará conduzindo, enquanto o D4 não. Isto ocorre devido ao sentido da corrente e a polarização dos diodos, logo pode-se considerar o D4 como uma chave aberta. Vo' é a diferença de potencial entre os terminais A e B, e equivale ao inverso do sinal de entrada. Já para o caso do sinal de entrada negativo, a condutividade dos diodos é inversa, portanto o D3 se comporta como uma chave aberta e o D4 conduz a corrente. Novamente este amplificador funcionará como um inversor, e podemos observar na Figura 14 que a tensão Vo'', diferença de potencial entre os terminais A e B, é exatamente o inverso da entrada.

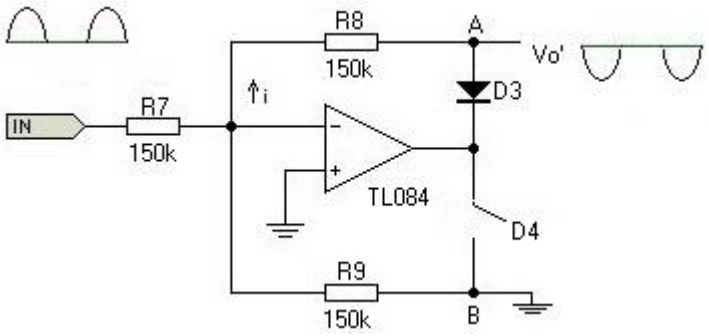

Figura 13. Retificação de onda completa - caso 1: sinal de entrada positivo.

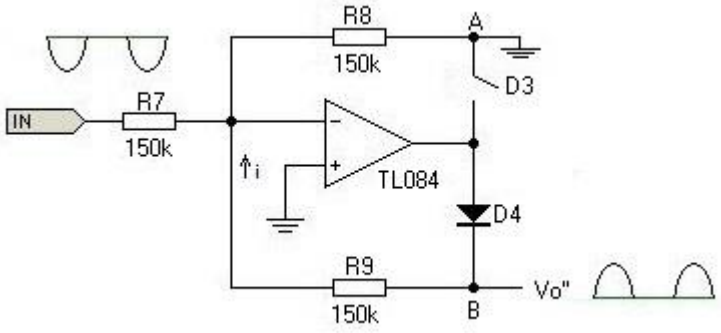

Figura 14. Retificação de onda completa - caso 2: sinal de entrada negativo.

A grande diferença entre os casos 1 e 2 citados acima é a posição do sinal em relação ao terra. Enquanto no primeiro caso a saída Vo' é negativa e está no terminal A, em relação ao terra no terminal B, no segundo caso temos Vo'' positiva no terminal B e o terra o terminal A.

Com as fases positivas e negativas do sinal separadas as Figura 15 e Figura 16 mostram que, no primeiro caso onde temos o terminal A negativo e o terminal B

como sendo o terra, o amplificador operacional funciona como inversor, tornando todo o sinal positivo, pois Vo' está conectado ao terminal inversor. Já no segundo caso temos o sinal Vo'' positivo conectado ao terminal não inversor do amplificador operacional, e o terra ligado ao terminal inversor, mantendo o sinal positivo.

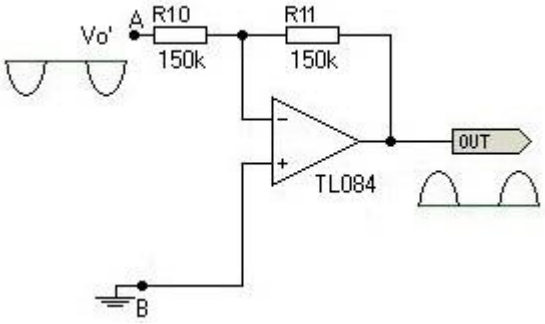

Figura 15. Retificação de onda completa - caso 3: sinal de entrada Vo'.

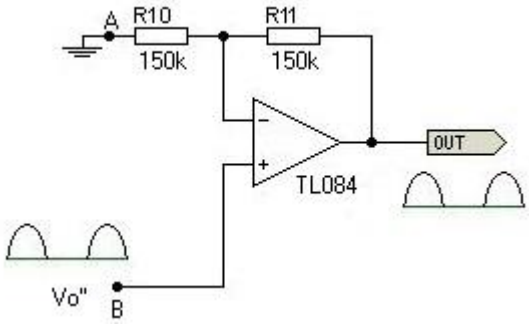

Figura 16. Retificação de onda completa - caso4: sinal de entrada Vo''.

Assim, somando todas as partes citadas anteriormente, o circuito pode completo pode ser observado na Figura 17, onde a entrada é um sinal arbitrário, contendo tensões positivas e negativas e a saída é o sinal completamente retificado, todo positivo.

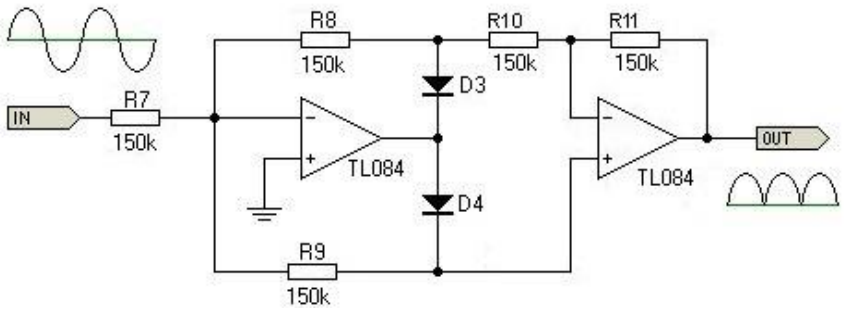

Figura 17. Retificador de onda completa.

Capítulo 5

Setup Experimental

5.1 Aquisição de dados

Para captar os sinais pela EMG foi utilizado o Muscle Sensor v3. Este kit foi conectado ao corpo por três eletrodos: um na porção ventral do musculo, outro na extremidade e um terceiro num ponto de referência sem musculatura (cotovelo), Figura 18. O Muscle Sensor v3 possui filtros, amplificadores e um retificador, sendo assim capaz de disponibilizar um sinal sem ruídos e pronto para ser utilizado num microcontrolador.

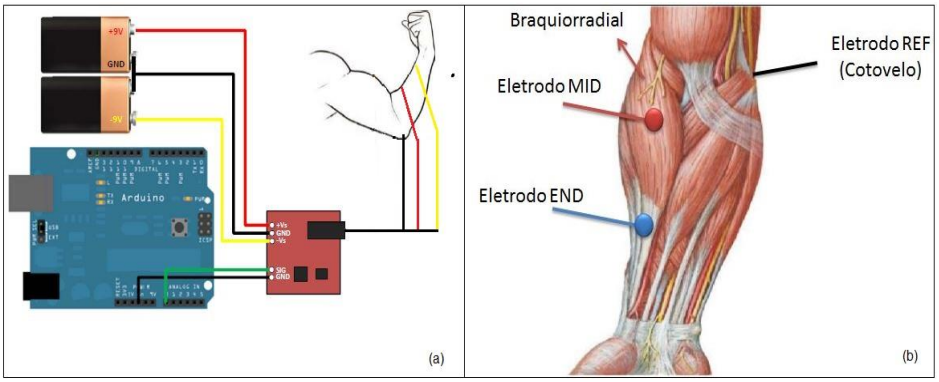

Figura 18. (a) conexões eletrônicas do sistema. (b) posicionamento dos sensores.

Existem vários plataformas de microcontroladores que podem ser realizar a tarefa de aquisição e processamento de dados do Muscle Sensor como Arduino, Raspberry Pi, Tiva, entre outras. Dentre essas plataformas, foi escolhido o Arduino pelos seguintes motivos:

- Facilidade de implementação – IDE, C++ e grande comunidade ativa de usuários;
- Módulo de motor - Motor Shield v3;
- Baixo Custo - um ATmega custa em torno de U$4;

- Placa de desenvolvimento pronta;
- Open-Source;

5.2 Extração de Características

Segundo Hsie-Jen (40), as principais características a serem analisadas em um sinal eletromiográfico são:

- Média Móvel;
- Variância Móvel;
- Frequência.

Neste trabalho o sinal EMG foi analisado a partir de sua média e variância, sendo calculados numa janela deslizante de 100 amostras.

O tamanho da janela foi definido empíricamente, e este valor foi definido por trazer resultados suficientes e não realizar uma forte filtragem no sinal, pois uma janela deslizante de média caracteriza um filtro de suavização (41).

De acordo com (41), pode-se expressar a média móvel como uma convolução entre o sinal e uma janela $w(n)$ de tamanho N, onde cada elemento tem valor $\frac{1}{N}$. Então, a equação da média móvel, expressada como convolução, resulta em

$$MM = (f * w)[k] = \sum_{m=-\infty}^{\infty} f[m]w[k-m]. \quad (20)$$

MM é a média móvel de mesmo tamanho de $f(m)$, que representa o sinal.

Já a variância é uma medida da dispersão de uma variável entorno de seu valor esperado (42). Quando estimando a variância da população usando n amostras aleatórias x_i onde $i = 1, 2, \ldots, n$, a fórmula é dada por

$$s^2 = \frac{1}{n-1}, \quad (21)$$

onde $\bar{x}$ é a média da amostra.

Os extratores foram implementados na linguagem Matlab, para o treino da rede neural, e em C++, para o Arduino, podendo, desta forma, realizar o controle da prótese. Os códigos estão nos apêndices C e D.

5.3 Classificador

A rede neural Perceptron de Multiplas Camadas foi utilizada para realizar a classificação dos sinais EMG. Em sua fase de treino, os sinais adquiridos foram pré-processados e inseridos numa rede MLP implementada em Matlab, já para o controle da prótese, a fase de propagação da rede foi implementada em C++, para ser utilizada no Arduino. Os códigos estão no Apêndice A e B, respectivamente.

No treinamento da rede MLP foi utilizado o algoritmo *backpropagation*, por ser um método bastante utilizado na literatura (43) (44) (45) (46). Um dos problemas em microcontroladores é a falta de memória embarcada, assim, o treinamento *Online* da rede neural é o mais indicado por utilizar uma quantidade menor deste recurso (30).

Para escolher os parâmetros da rede neural, testes empíricos foram realizados, comparando cada um dos resultados. Já a função de ativação de todos os neurônios foi fixada em tangente hiperbólica, pois com sua utilização só se faz necessário um neurônio na camada de saída, e assim já podemos controlar as duas direções do motor. As tangentes hiperbólicas possuem intervalo igual a [-1,1], dessa forma pode-se modelar o valor -1 representando o motor girando em uma direção e o valor 1, o motor girando em outra direção.

Algumas configurações de redes neurais foram testadas, afim de mostrar a sua eficiência alterando-se os parâmetros de taxa de aprendizado e número de neurônios na camada escondida. O resultado das combinações pode ser encontrado na Tabela 3, que demonstra o erro médio quadrático para cada uma das configurações testadas da rede em seu último ciclo.

Todos os testes foram conduzidos utilizando a tangente hiperbólica como função de ativação. Com este definiu-se que a configuração da rede neural a ser

utilizada possui 0,1 de taxa de aprendizado e 10 neurônios na camada escondida como ilustrado na Figura 19.

Tabela 3. Erro médio quadrático em diferentes configurações da rede neural permutando taxa de aprendizado (α) e número de neurônios na camada escondida (h).

	$\alpha = 0{,}01$	$\alpha = 0{,}1$	$\alpha = 0{,}5$
$h = 1$	$7{,}96\ x\ 10^{-2}$	$6{,}88\ x\ 10^{-3}$	$1{,}08\ x\ 10^{-4}$
$h = 2$	$2{,}02\ x\ 10^{-3}$	$8{,}33\ x\ 10^{-4}$	$1{,}99\ x\ 10^{-4}$
$h = 5$	$3{,}23\ x\ 10^{-4}$	$3{,}92\ x\ 10^{-4}$	$-2{,}79\ x\ 10^{-4}$
$h = 10$	$1{,}51\ x\ 10^{-4}$	$3{,}40\ x\ 10^{-5}$	$6{,}08\ x\ 10^{-4}$

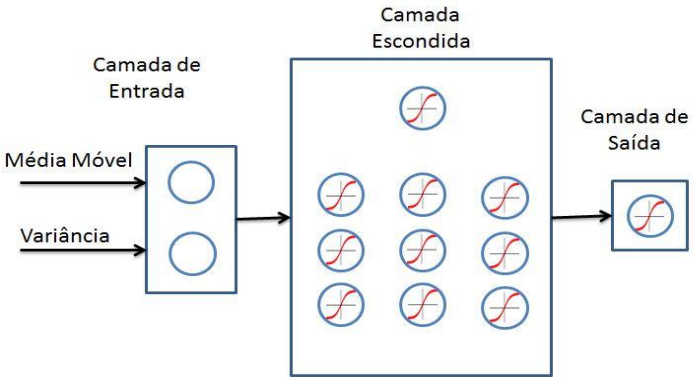

Figura 19. Arquitetura da Rede Neural

A fase de propagação do sinal de entrada da rede neural foi definida pelas Equações (1) e (2), que representam as entradas líquidas dos neurônios e sua função de ativação. Já para representar a fase de retropropagação do sinal, com os ajustes dos pesos sinápticos, as Equações (9) à (12).

5.4 Atuador

Para realizar o movimento de pinça com a prótese, foi utilizado um motor de corrente contínua (DC) com as seguintes propriedades:

- Velocidade de 15 rotações por minuto (RPM)
- Tensão de funcionamento de 12 V
- Redução para elevar o torque à 2 N.m

O motor DC gera energia mecânica rotacional continuamente ao ser aplicado uma tensão em seus terminais. Desta forma, como há um limite para a abertura e o fechamento da prótese foi necessário adicionar botões de fim de curso afim de limitar a trajetória da prótese movida pelo motor.

Foi utilizado um Arduino Uno (47) equipado com um Motor Shield V3 (37) para controlar o motor. O Motor Shield V3, ilustrado na Figura 20, foi necessário por possuir uma ponte-H, e ter uma interface facilitada com o Arduino, permitindo controlar a movimentação do motor para ambas as direções com facilidade.

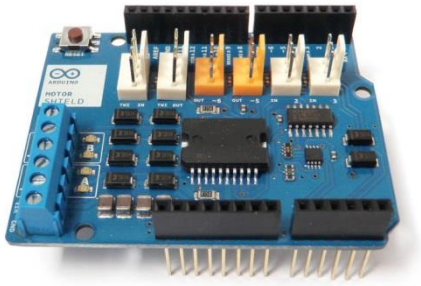

Figura 20. Motor Shield V3.

O controle de velocidade do motor é realizado pelo pino PWM 3 do Arduino conectado ao Motor Shield V3, entretanto, a ultima camada da rede neural possui a tangente hiperbólica como função de ativação, logo a sua saída consiste de valores entre -1 e 1. As saídas PWM do Arduino possuem um intervalo de funcionamento entre 0 e 255, representando 0 e 5V respectivamente, desta forma foi necessário

mapear a saída da rede neural para a saída do Arduino indicar se o valor é negativo para o pino de controle de direção. O código completo do atuador pode ser visualizado no apêndice E.

5.5 Testes

O objetivo deste trabalho é propôr um sistema de controle para prótese de mão. Os testes foram realizado na prótese que está em desenvolvimento no programa de mestrado de engenharia de sistemas vinculado a Universidade de Pernambuco. Esse projeto faz parte de um grupo de pesquisa que está se aprofundando em equipamentos tecnológicos desenvolvidos para área de saúde. A prótese atual deste grupo de pesquisa possui apenas um motor capaz de realizar o movimento de pinça. Entretanto, o sistema proposto é escalonável para outras próteses, servindo como um *background* a ser utilizado para o controlar vários motores e movimentar os dedos independentemente, no futuro.

A partir da implementação de todos os passos ateriores, o sistema foi conectado à protese, sendo ele testado por dois indivíduos. Para isto foi necessário realizar a coleta de dados de EMG dos dois indivíduos para treinar a rede neural para cada um dos casos.

Os movimentos que controlaram a protese são a contração e o relaxamento do músculo Braquiorradial. A coleta dos dados consistiu em conectar três eletrodos no antebraço e utilizar um osciloscópio para armazenar o sinal EMG emitido pelo Muscle Sensor. As posições dos eletrodos estão ilustradas na Figura 18b.

Os voluntários permaneceram sentados, com o braço estendido sobre à mesa. Foi requisitado que eles fechassem a mão, e permanecessem assim por 10 segundos, repetindo o experimento por 5 vezes, e logo após foi adquirido os dados referentes à posição de relaxamento da mão, também por 10 segundos e 5 repetições.

Foram realizadas 30 execuções da rede neural afim de verificar sua plausibilidade. Com isso pôde-se extrair os seguintes parâmetros estatísticos: taxa

de classificação, taxa de cobertura e taxa de precisão. Esses parâmetros são dados por

$$Taxa\ de\ Classificação = \frac{\#\ amostras\ corretamente\ classificadas}{total\ de\ amostras}, \tag{22}$$

$$Taxa\ de\ Cobertura = \frac{\#\ amostras\ de\ mão\ fechada\ corretas}{Amostras\ classificadas\ como\ mão\ fechada}, \tag{23}$$

$$Taxa\ de\ precisão = \frac{\#\ amostras\ de\ mão\ fechada\ corretas}{Total\ de\ amostras}, \tag{24}$$

Capítulo 6

Resultados

6.1 Aquisição de dados

Como pode-se observar na Figura 21 Dependendo do voluntário avaliado a amplitude do sinal de EMG varia, uma deles teve diferença muito mais notável entre as amostras de mão aberta e mão fechada, em torno de 700 mV contra 300 mV do segundo voluntário. Esta diferença se dá por diversos fatores, como capacidade de ativação neural, força ou preparação muscular e fadiga. Porém, é esperado que ainda assim a rede neural seja capaz de classificar as amostras do segundo voluntário.

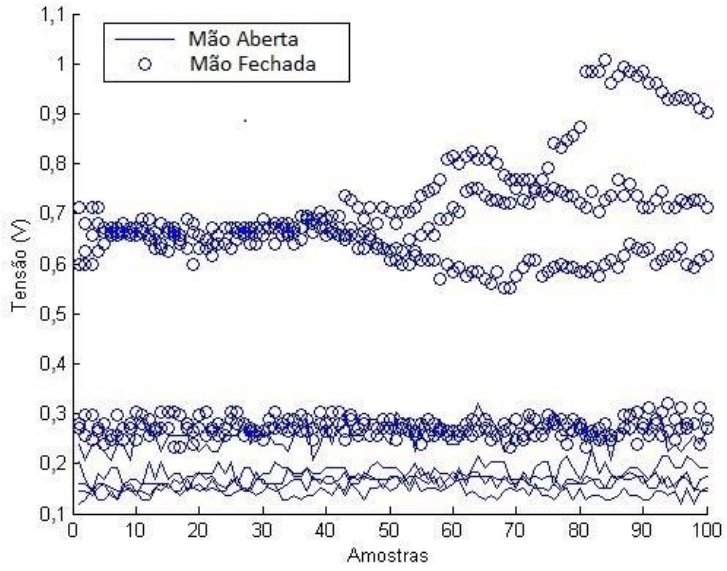

Figura 21. Dados coletados pelo Muscle Sensor v3 dos dois voluntários distribuídos aleatóriamente.

6.2 Extração de Características

As médias móveis podem ser observadas na Figura 22. Elas funcionam como filtros passa-baixa nos sinais, disponibilizando para a rede neural sinais que retratam o nível DC do sensor EMG.

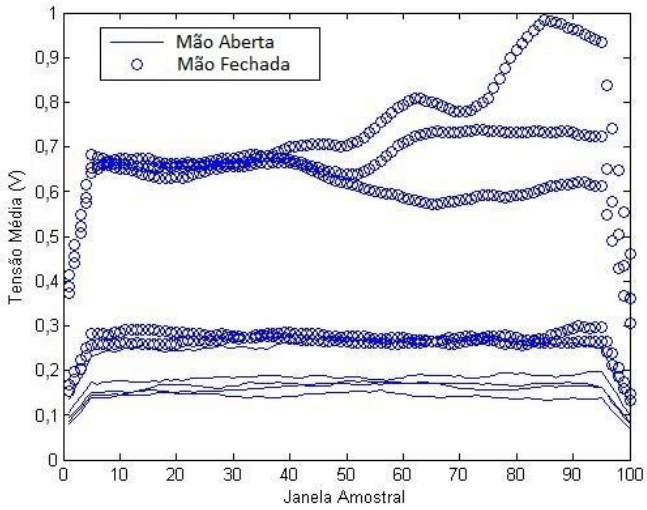

Figura 22. Médias Móveis dos dados dos dois voluntários.

Pelaa Figura 23 nota-se a diferença entre as variâncias dos sinais de mão fechada e mão aberta para os dois voluntário.

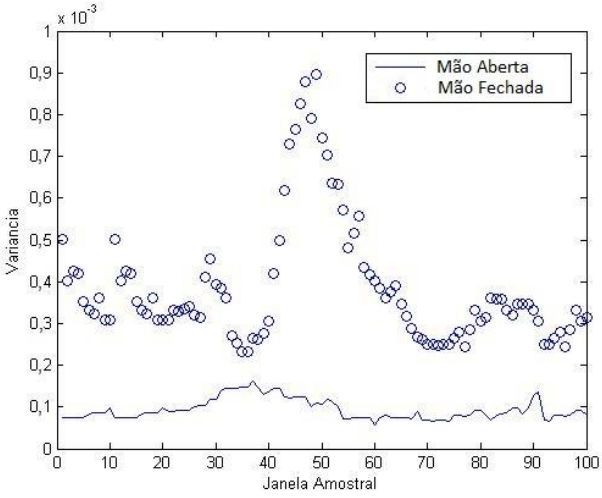

Figura 23. Variância dos dados de um dos voluntários.

6.3 Rede Neural MLP – Fase de treinamento

Como método de parada do treino, foi utilizado a quantidade de ciclos, que foi estipulado em 5, pois, como pode ser observado na Figura 24, a rede converge muito antes desse valor.

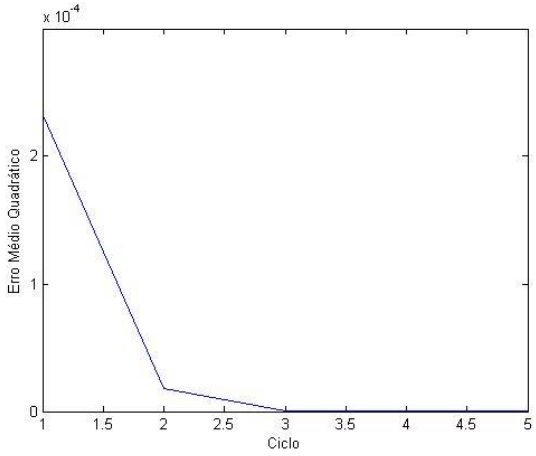

Figura 24. Erro Médio Quadrático para a rede neural durante 5 ciclos.

É possível observar o erro de classificação nos ciclos 1, e 5 do treinamento, pela Figura 25 As poucas amostras que ao fim foram classificadas errôneamente não interferem no desempenho do sistema, pois cada uma delas representa 1 μs de atuação no motor e seu momento de inércia não permite que sua rotação seja alterada em tão pouco tempo.

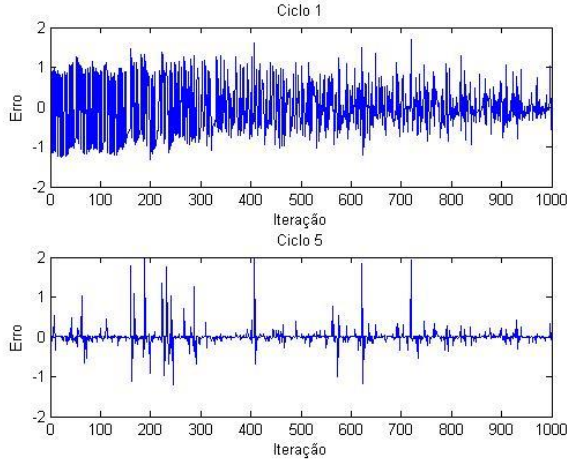

Figura 25. Erros nos ciclos 1, e 5 da rede neural.

6.4 Testes com a prótese

Após a integração de todas as partes do sistema, os testes foram executados com êxito na prótese. Na Figura 26 podemos observar todas as partes Físicas do sistema onde temos a prótese impressa em impressora 3D, com um motor DC acoplado, o arduino com o Motorshield, o kit Muscle Sensor, e os eletrodos de superfície utilizados.

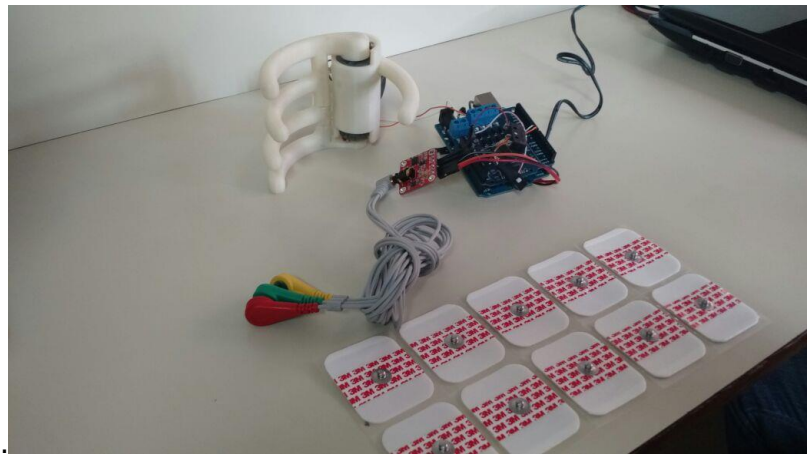

Figura 26. *Hardware* utilizado sistema.

Nas Figura 27 e Figura 28 podemos observar o sistema sendo utilizado. Na primeira imagem a mão do voluntário e a prótese estão abertas e na segunda , após o movimento de fechar a mão, a prótese também fechou. A mão esquerda do voluntário está segurando o botão de fim de curso, pois neste momento a prótese ainda não podia suportar tal dispositivo em sua estrutura.

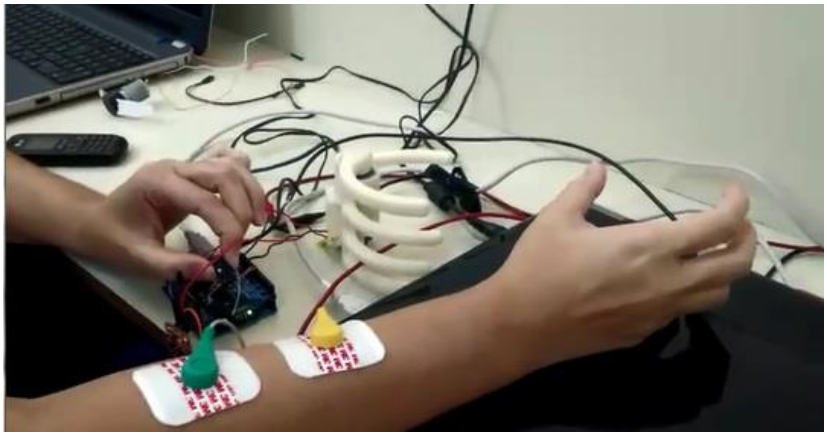

Figura 27. Sistema sendo utilizado. A Prótese e a mão estão na primeira posição: abertas.

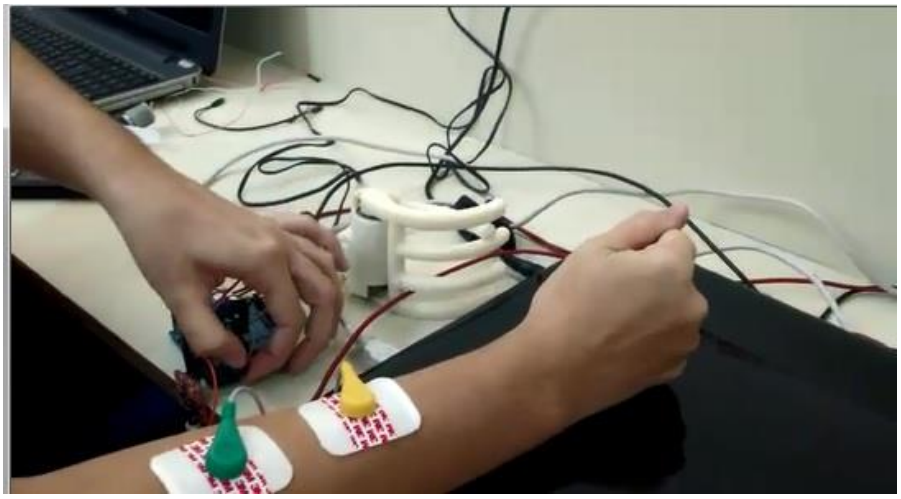

Figura 28. Sistema sendo utilizado. A Prótese e a mão estão na segunda posição: fechadas.

Após realizadas 30 execuções da rede neural, foram observadas os seguintes parâmetros estatísticos:

- Taxa de Classificação
    - Média: 0,9875
    - Desvio Padrão: 0,0014
- Taxa de Cobertura:
    - Média: 0,9751
    - Desvio Padrão: 0,0029
- Taxa de Precisão:
    - Média: 1
    - Desvio Padrão: 0

Capítulo 7

Conclusão e Trabalhos Futuros

Com este trabalho, nota-se que é possível controlar uma prótese de mão eletrônica, capaz de realizar o movimento de pinça, pelo sistema proposto. Tal sistema é composto por um Muscle Sensor v3 (um módulo de sensoriamento eletromiográfico), um conjunto de extratores de características (pré-processamento dos dados), uma rede neural MLP (classificador) e um Arduino munido de um Motor Shield v3 conectado ao motor da prótese. Com os testes sendo realizados para dois voluntários, o sistema apresentou boa confiabilidade, tendo uma taxa de erro mínima de $3.40 \text{ x}10^{-5}$.

Como o número de voluntários ao teste foi de apenas dois, trabalhos futuros podem ser realizados afim de testar e validar o sistema, elevando a quantidade de usuários.

Outro possível estudo futuro é o de investigar como melhorar o desempenho do sistema, utilizando outras formas de aquisição de dados, disponibilizando valores não filtrados para o pré-processamento, e, a partir daí, utilizar outros extratores de características como analisadores de frequência do sinal (transformadas de fourier, transformada wavelet, etc.), *zero-crossing*, entre outras. Também é possível investigar outras técnicas adaptativas para realizar a classificação do sinal, ou até mesmo utilizar outras configurações de redes neurais, potencializando o sistema e o deixando mais robusto.

Mesmo com o bom funcionamento do controle a partir do Arduino, outros microcontroladores podem baratear o custo do sistema, melhorar o desempenho e até mesmo gerar novas possibilidades para o sistema, como por exemplo Tiva C (48), e possui um DSP integrado em hardware, possibilitando a utilização de analisadores de frequência sem comprometer seu tempo de resposta.

Existem várias maneiras de melhorar o sistema proposto neste trabalho. Sua implantação tem impacto positivo na vida das pessoas com a ausência do membro,

pois acumula esforços afim de baratear e melhorar os sistemas de controle de próteses, podendo assim aumentar o número de pessoas que se propõem a utilizá-las e, com isso, devolver uma das funções deste membro aos deficientes que tanto necessitam, visando o dia em que um trabalho como este permitirá uma total recuperação.

Bibliografia

1. *Bionic Limbs: clinical reality and academic promises.* **Farina, D. and Aszmann, O.** 2014, Science Translational Medicine, Vol. 6, p. 12.

2. **Spichler, E.R.S., et al., et al.** Capture-recapture method to estimate lower extremity amputation rates in Rio de Janeiro, Brazil. *Revista Panamericana de Salud Pública.* 2001, pp. 334-340.

3. **Fonseca, M.C.R., et al., et al.** Traumas da mão: estudo retrospectivo. *Revista brasileira de ortopedia.* 2006, pp. 181-186.

4. **Cunha, F.L.** *Mão de são carlos, uma prótese multifunção para membros superiores: um estudo dos mecanismos, atuadores e sensores.* São Carlos : Escola de Engenharia de São Carlos, Universidade de São Paulo, 2002.

5. *Comparison of k-nearest neighbor, quadratic discriminant and linear discriminant analysis in classification of electromyogram signals based on the wrist-motion directions.* **KIM, Kang Soo et al.** 2011, Current applied physics, Vol. 11, pp. 740-745.

6. *Classification of EMG signals using PSO optimized SVM for diagnosis of neuromuscular disorders.* **SUBASI, Abdulhamit.** 2013, Computers in biology and medicine, Vol. 43, pp. 576-586.

7. *Intention detection using a neuro-fuzzy EMG classifier.* **HUSSEIN, Sherif E. and GRANAT, Malcolm H.** 2002, Engineering in Medicine and Biology Magazine, IEEE, Vol. 21, pp. 123-129.

8. *EMG pattern recognition based on artificial intelligence techniques.* **PARK, Sang-Hui and LEE, Seok-Pil.** 1998, Rehabilitation Engineering, IEEE Transactions on, Vol. 6, pp. 400-405.

9. *Reconhecimento de Padrões de Sinais de EMG para Controle de Prótese de Perna.* **FERREIRA, Renan Utida et al.** 2005, XI Congresso Brasileiro de Biomecânica, pp. 1-5.

10. *A real-time EMG pattern recognition system based on linear-nonlinear feature projection for a multifunction myoelectric hand.* **Chu, Jun-Uk, Inhyuk Moon, and Mu-Seong Mun.** 2006, Biomedical Engineering, IEEE Transactions on, pp. 2232-2239.

11. **Polis, J.E.** *Projeto e construção de parte estrutural de prótese de mão humana.* Campinas : Faculdade de Engenharia Mecanica, 2009.

12. **Favieiro, G.W.** *Controle de uma prótese experimental do segmento mão braço por sinais mioelétricos e redes neurais artificias.* s.l. : Universidade Federal do Rio Grande do Sul., 2009.

13. **Sono, T.S.P.** *Projeto de um sistema de controle sub-atuado para uma prótese de mão.* Rio de Janeiro : Instituto Militar de Engenharia, 2008.

14. *Effects of electrode location on myoelectric conduction velocity and median frequency estimates.* **Roy, S., De Luca, C. J. and Schneider, J.** 1986, J Appl Physiol, Vol. 61, pp. 1510-1517.

15. *A new bipolar indwelling electrode for electromyography.* **Basmajian, J. V. and Stecko, G.A.** 1962, J Appl Physiol, Vol. 17, p. 849.

16. **Wikipedia.** Wikipedia. [Online] [Cited: 06 15, 2015.] ttps://en.wikipedia.org/wiki/Electromyography#History.

17. **Portney, L. G. and Roy, S. H.** *Eletromiografia e testes de velocidade de condução nervosa. Fisioterapia avaliação e tratamento.* São Paulo: Manole : s.n., 2004. pp. 213-256.

18. **Forti, F.** *Surface Electromyography: Detection and Recording.* Universidade Metodista de Piracicaba. Piracicaba : s.n., 2005. Pós-Graduação.

19. **Burke, R.E.** Motor Units: Anatomy, Physiology, and Functional Organization. *Handbook of Physiology: The Nervous System Motor Control.* 1981, Vol. 2, pp. 345-422.

20. *A Method for positioning electrodes during surface EMG recordings in lower limb muscles.* **Rainoldi, A., Melchiorri, G. and Caruso, I.** 2004, Journal of Neurosciences Methods, Vol. 134, pp. 37-43.

21. *Linear and non-linear surface EMG/force relationships in human muscles.* **Woods, J. J. and Bigland-Ritchie, B.** 1983, Am J Phys Med, Vol. 62, pp. 287-298.

22. **Forti, F.** *Análise Do Sinal Eletromiográfico Em Diferentes Posicionamentos, Tipos De Eletrodos, Ângulos Articulares E Intensidades De Contração.* Piracicaba : Universidade Metodista de Piracicaba, 2005.

23. *Surface Electromyography: Detection and Recording.* **De Luca, C.J.** 2002, Delsys Incorporated.

24. **Sodeberg, G.L. and T.M., Cook.** Electromyography in Biomechanics. *Physical Therapy.* 1984, pp. 485-498.

25. **Hakin, S.** *Neural Networks a comprehensive foundation.* Singapore : Pearson Education, 2005.

26. *The human brain in numbers: a linearly scaled-up primate brain.* **HERCULANO-HOUZEL, S.** 2009, Frontiers in Human Neuroscience, pp. 3-31.

27. *Do we have brain to spare?* **Drachman, D.** 2005, Neurology.

28. *A logical calculus of the ideas immanent in nervous activity.* **McCulloch, W.S. and Pitts, W.** 1943, Bulletin of Mathematical Biophysics, Vol. 5, pp. 115-113.

29. **Heeb, D.O.** *The organization of behavior: A neuropsychological theory.* New York: Wiley : s.n.

30. **Valença, M.** *Fundamentos das Redes Neurais: exemplos em java.* [ed.] Tarcísio Pereira. 2. Recife : Livro Rápido - Elógica, 2013.

31. *Polynomial theroy of complex systems.* **Ivakhnenko, A. G.** 1971, IEE Transaction on Systems, Man, Cybernetics, Vol. 1, pp. 364-378.

32. **Valença, M.** Aplicando Redes Neurais: um guia completo. Recife : Livro Rápido, 2005, p. 284.

33. **Braga, A. P., Carvalho, A. P. L. F. and Ludemir, T. B.** *Redes Neurais Artificiais: Teoria e Aplicações.* Rio de Janeiro : Livros Técnicos e Científicos S.A. - LTC, 2000.

34. **Million, E.** *The Hadamard Product.* 2007.

35. **Arduino.** Arduino Boards. [Online] [Cited: 05 31, 2015.] http://www.arduino.cc/en/Main/Boards.

36. **STMicroelectronics.** *L298 Dual full-bridge driver.* s.l. : STMicroelectronics, 2000.

37. **Arduino.** Motor Shield v3. [Online] [Cited: 05 31, 2015.] http://www.arduino.cc/en/uploads/Main/arduino_MotorShield_Rev3-schematic.pdf.

38. **ADVANCER TECHNOLOGIES.** *Muscle Sensor v3 User Manual.* 2013.

39. **Analog Devices.** *AD8221 Precision Instrumentation Amplifier.* s.l. : One Technology Way, 2003.

40. *An Adaptive Upper-Arm EMG-Based Robot.* **Hsie-Jen and Kuu-Young.** 2010, International Journal of Fuzzy Systems,.

41. **Smith, Steven W.** *The Scientist and Engineer's Guide to.* California : Analog Devices, 1997.

42. **Evans, Lawrence C.** *An introduction to stochastic differential equations Version 1.2.* 2012.

43. *Removal of EMG Interference from Electrocardiogram Using Back Propagation.* **Singh, G. and Kaur, R.** 6, 2013, International Journal of Innovative Research in Computer and Communication Engineering, Vol. 1, pp. 1300-1305.

44. *INTERNATIONAL JOURNAL OF ADAPTIVE CONTROL AND SIGNAL PROCESSING.* **Tsuji, T., et al., et al.** 2000, Pattern classi"cation of time-series EMG signals using, Vol. 14, pp. 829-848.

45. *Neural Network Models in EMG Diagnosis .* **Pattichis, Constantinos s., Schizas, Christos N. and Middleton, Lefkos T.** 5, 1995, IEEE TRANSACTIONS ON BIOMEDICAL ENGINEERING, Vol. 42, pp. 486-496.

46. *Classification of Myoelectric Signals Using Multilayer Perceptron Neural Network with Back Propagation Algorithm in a Wireless Surface Myoelectric Prosthesis.* **Manalo, Kevin D., Linsangan, Noel B. and Torres, Jumelyn L.** 9, 2016, International Journal of Information and Education Technology, Vol. 6, pp. 686-690.

47. **Arduino.** Arduino Uno. [Online] Arduino. [Cited: 05 31, 2015.] http://www.arduino.cc/en/Main/ArduinoBoardUno.

48. **Texas Instruments.** *Tiva™ TM4C123GH6PM Microcontroller Datasheet.* Austin : Texas Instruments, 2007.

49. *Additions to the article* On a New Class of Theorems In elimination between quadratic functions.* **SYLVESTER, J. J.** 1850, Philosophical, pp. 363-370.

50. *An Introduction to Neural Computing.* **ALEKSANDER, I. and MORTON, H.** 1995, International Thomson Computer Press, p. 490.

Anexo A

Matrizes

O desenvolvimento das matrizes ocorreu a partir do século XIX, apesar de se ter representações de números semelhantes as matrizes modernas desde a Era Cristã, com matemáticos como Arthur Cayley, William Rowan Hamilton e Augustin-Louis Cauchy, sendo o último quem possívelmente foi o primeiro a batizar essa representação, com o nome de *tableau (*tabela).

Contudo, apenas em 1850, com um artigo de Sylvester, J.J. (49), o termo matriz foi utilizado pela primeira vez:

"[...]Para isso, devemos iniciar, não com um quadrado, mas com um arranjo retangular de termos que consistem, suponho, de m linhas e n colunas. Isso não vai, por si só representam um determinante, mas é, por assim dizer, uma **matriz** a partir da qual se podem formar vários sistemas de determinantes, fixando-se um número p, e selecionando p linhas e p colunas, os quadrados correspondem ao que pode ser chamado de determinantes de ordem p. Temos, então, a seguinte proposição. O número de determinantes que constituem um sistema da ordem de P derivado a partir de uma dada matriz, n termos gerais e m termos de profundidade, pode ser igual, mas pode nunca exceder este número."

Recentemente, com as planilhas eletrônicas de computador, cálculos que antes eram realizados à mão de forma cansativa de repetitiva agora têm sua solução obtida com mais velocidade e precisão. Essas planilhas, em geral, são formadas por tabelas que armazenam os dados utilizados no problema a ser resolvido. Somos muito acostumados com essa realidade, mas nem sempre isto foi uma verdade.

Conceito

Portanto, como Sylvester (49) definiu, uma matriz é um arranjo de termos formado por m linhas e n colunas. Esses termos são símbolos de um conjunto e são

representados dentro de um quadro. Uma matriz com m linhas e n colunas é chamada de matriz de ordem m por n(m x n). como pode-se observar na Figura 29.

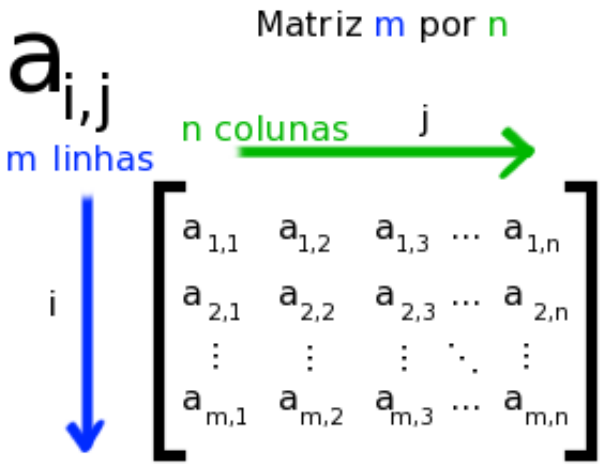

Figura 29. Representação matricial.

Cada elemento da matriz pode ser escrito como uma variável com dois subscritos, onde o primeiro representa a linha cujo o elemento pertence, e o segundo representa a coluna.

Algumas Operações e Propriedades Envolvendo Matrizes

Multiplicação de um número real por uma matriz

Para multiplicar um número *k,* pertencente ao conjunto dos números reais, por uma matriz $A_{m,n}$, basta multiplicar cada elemento da matriz por k. Assim a matriz resultante terá a mesma ordem da matriz original.

Adição e subtração entre as matrizes

Dado as matrizes $A_{m,n}$ e $B_{m,n}$, a sua soma A+B é uma matriz de mesma ordem computada somando os elementos de mesmo índice subscrito.

Multiplicação entre as matrizes

Apenas ocorre quando o número de colunas da matriz da esquerda é igual ao número de linhas da matriz da direita, desta forma a multiplicação entre matrizes não é comutativa, e gera uma matriz que possui o número de linhas da matriz da esquerda e o número de colunas da matriz da direita. Se A é uma matriz m-por-n e B é uma matriz n-por-p, então seu produto é uma matriz m-por-p chamada de AB. O produto é dado por

$$(AB)_{ij} = \sum_{r=1}^{n} a_{ir} b_{rj}. \qquad (25)$$

Matriz Transposta

A matriz transposta de uma matriz $A_{m,n}$ é a matriz $A_{n,m}^{T}$ em que os elementos $a_{ij}^{T} = a_{ji}$, ou seja, todos os elementos da linha n se tornarão elementos da coluna n.

Matriz de Permutação

É uma matriz quadrada binária que tem o poder de gerar uma permutação específica em um vetor ou entre linhas ou colunas de uma dada matriz. A matriz de Permutação é uma transformação linear e é obtida através da troca das linhas (ou colunas) da matriz identidade de mesma ordem.

Um exemplo de permutação onde P: N → N, onde o conjunto N = {1,2,3,4,5} e Mp é a matriz permutação, pode ser representada da seguinte forma:

$$Mp = \begin{pmatrix} 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \end{pmatrix}, N = \begin{pmatrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{pmatrix},$$

$$P = Mp \,.\, N \qquad (26)$$

Logo,

$$P = \begin{pmatrix} 4 \\ 5 \\ 2 \\ 3 \\ 1 \end{pmatrix}.$$

Com a matriz de permutação, é possível realizar a mesma permutação em vários vetores, ou ainda em várias linhas ou colunas de uma matriz.

Produto de Hadamard

É uma operação binária entre duas matrizes de mesma dimensão. É dado pela multiplicação entre cada elemento ij da primeira matriz, pelo seu respectivo elemento de mesma coordenada na segunda matriz. Esta operação também é chamada de *element-wise multiplication.*

Por exemplo, o produto de Hadamard entre as matrizes A_{3x3}e B_{3x3} é

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} \circ \begin{pmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ b_{31} & b_{32} & b_{33} \end{pmatrix} = \begin{pmatrix} a_{11}b_{11} & a_{12}b_{12} & a_{13}b_{13} \\ a_{21}b_{21} & a_{22}b_{22} & a_{23}b_{23} \\ a_{31}b_{31} & a_{32}b_{32} & a_{33}b_{33} \end{pmatrix}. \quad (27)$$

Apêndice A

MLP – Matlab

```
%%%%%%%%%%%%%%%%%%%%%%%%%%
%       in – input vector                       %
%       output – output vector                  %
%       coeff – learning coefficient                    %
%%%%%%%%%%%%%%%%%%%%%%%%%%
%       y2 – calculated output at last cycle    %
%       weights_1 – optimum layer 1 weights     %
%       weights_2 – optimum layer 2 weights     %
%       ERR – mean squared error                %
%       err – errors in each cycle              %
%%%%%%%%%%%%%%%%%%%%%%%%%%

function [y2,weights_1,weights_2,ERR,err] = RNA_completo(in,output,coeff)

  %first layer bias
  S = size(in);
  input = ones(S(1)+1,S(2));
  input(2:S(1)+1,:) = in;

  % ouput neurons
  S_O = size(output);
  o_n = S_O(1);

  % hidden layer neurons
  h_l_n = 10;
  h_l_n = h_l_n +1; %bias

  % Number of learning cycles
  cicles = 50;

  % Calculate weights randomly using seed.
  rand('state',sum(100*clock));
  weights_1 = -1 +2.*rand(h_l_n,length(input(:,1)));
  weights_2 = -1 +2.*rand(o_n,h_l_n);

  %examples
  numIn = length (input(1,:));

  for it = 1:cicles
    for(i=1:numIn)
      %Compute hidden layer output
      NET_1(:,i) = weights_1 * input(:,i);
      y1(:,i) = tanh(NET_1(:,i));
      y1(1,i) = 1;

      %Compute last layer output (system output)
      NET_2(:,i) = weights_2 * y1(:,i);
      y2(:,i) = tanh(NET_2(:,i));
```

```
        %Backpropagation:::
        %Compute Errors and propagate through the layers
        Sigma_2(:,i) = (sech(y2(:,i)).^2).*(output(:,i)-y2(:,i));
        Sigma_1(:,i) = (sech(y1(:,i)).^2).*((weights_2')*Sigma_2(:,i));

       %UPDATE WEIGHTS
       weights_2 = weights_2 + coeff*(Sigma_2(:,i)*y1(:,i)');
       weights_1 = weights_1 + coeff*(Sigma_1(:,i)*input(:,i)');
     end
     err(it,:) = y2-output;
     ERR(it) = sum(((err(it)).^2))/length(output);
   end
end
```

Apêndice B

MLP – Arduino

```
#include "Arduino.h"
#include "RNA.h"
#include "math.h"
void RNA::run()
{
	float weightsLayer1[11][3] = {{-0.83,0.59,-0.48},{1.44,-4.19,-2.44},{1.00,-2.99,-1.95},{-
	0.28,1.66,0.17},{0.00,0.66,0.57},{-1.14,4.28,0.77},{0.33,-0.54,0.10},{-
	0.61,2.13,1.23},{1.24,-3.26,-2.42},{0.16,-1.03,-0.54},{1.82,-5.25,-2.81}};
	float weightsLayer2[1][11] = {0.17,2.21,1.61,-0.73,-0.34,-1.97,0.33,-1.09,1.82,0.57,2.77};
	//Variables Definition
	float netLayer1[11] = {0,0,0};
	float yLayer1[11] = {0,0,0};
	// Pointers Definition
	float *p_input = input;
	float *p_weightsLayer1 = weightsLayer1[0];
	float *p_netLayer1 = netLayer1;
	float *p_yLayer1 = yLayer1;
	//Compute NET1
	net(p_input,p_weightsLayer1,
	,3,p_netLayer1);
	actvationFunction(p_netLayer1,11,p_yLayer1);
	yLayer1[0] = 1;
	//Variables Definition
	float netLayer2[1] = {0};
	// Pointers Definition
	float *p_weightsLayer2 = weightsLayer2[0];
	float *p_output = output;
	float *p_netLayer2 = netLayer2;
	//Compute NET2
	net(p_yLayer1,p_weightsLayer2,1,11,p_netLayer2);
	actvationFunction(p_netLayer2,1,p_output);
}
```

```
void RNA::net(float *in, const float *weights, int weightsRows, int weightsColumns, float *out)
{
	for(int j=0; j<weightsRows; j++)
	{
		for(int i=0; i<weightsColumns; i++)
		{
			//treats the weights matrix like an array
			out[j] += weights[(weightsColumns*j)+i] * in[i];
		}
	}
}
void RNA::actvationFunction(float *netValues, int layerSize, float *out)
{
	float ePositive[layerSize];
	float eNegative[layerSize];
	for(int i=0; i<layerSize; i++)
	{
		float e = 2.72;
		ePositive[i] = pow(e,netValues[i]);
		eNegative[i] = 1 / ePositive[i];
		out[i] = (ePositive[i] - eNegative[i]) / (eNegative[i] + ePositive[i]);
	}
}
```

Apêndice C

Extratores de Características – Arduino

```
#include "Arduino.h"
#include "FeaturesExtractor.h"
#include "math.h"
FeaturesExtractor::FeaturesExtractor()
{
	lastIndex = 0;
	size = 10;
}
float FeaturesExtractor::mean(float windowM[])
{
	float m = 0;
	for(int i=0; i<size;i++)
	{
		m += windowM[i];
	}
	return m/size;
}
float FeaturesExtractor::var(float windowV[], float mean)
{
	float v = 0;
	float temp = 0;
	for(int i=0; i<size;i++)
	{
		temp = windowV[i] - mean;
		temp = pow(temp,2);
		v+=temp;
	}
	v = v/(size-1);
	return v;
```

```
}
void FeaturesExtractor::add(float input)
{
	if(lastIndex == (size-1))
	{
		lastIndex = 0;
	}
	else
	{
		lastIndex = lastIndex + 1;
	}
	window[lastIndex] = input;
	output[0] = 1;
	output[1] = mean(window);
	output[2] = var(window,output[1]);}
```

Apêndice D

Extratores de Características – Matlab

```
W(1:100) = 1/100;
W = W';
A = csvread('MuscleSensor_Databank\Clarinha\Mao_Aberta (1).csv',0,4,[0 4 1000 4]);
a_min = min(A);
a_max = max(A);
M = conv(A,W,'same'); %Media móvel
V = movingWindowVar(A,100); %Variancia Mível

function R = movingWindowVar(M, window)
L = length(M);

for i = window:L-window
   i_menos = i - window + 1;
   i_mais = i + window;
   R(i_menos) = var(M(i_menos:i_mais));
end

R = R';
R = vertcat(R(1:window),R);
L2 = length(R);
R = vertcat(R,R(i_menos-(L-L2-1):i_menos));
end
```

Apêndice E

Controle do Motor - Arduino

```
#include "Arduino.h"
#include "Motor.h"
Motor::Motor(){
	motorSpeedPin = 3;
	motorDirectionPin = 4;
	snapActionPin_1 = 9;
	snapActionPin_2 = 10;
	pinMode(motorDirectionPin, OUTPUT);
	pinMode(snapActionPin_1, INPUT);
	pinMode(snapActionPin_2, INPUT);
}
void Motor::act(){
	snapActionValue_1 = digitalRead(snapActionPin_1);
	snapActionValue_2 = digitalRead(snapActionPin_2);
	if(output > -0.1 && output < 0.1){ // limiar de parada do motor;
		analogWrite(motorSpeedPin,0);
	}
	else if (output >= 0.1){
		digitalWrite(motorDirectionPin, HIGH);
		output = mapFloat(output, 0, 1, 0, 255);


		//SNAP ACTION BUTTON
		if (snapActionValue_1==LOW){
			analogWrite(motorSpeedPin, output);
		}else{
			analogWrite(motorSpeedPin, 0);
		}
	}else if (output <= -0.1){
		digitalWrite(motorDirectionPin, LOW);
		output *= -1;
		output = mapFloat(output, 0, 1, 0, 255);
```

```
                //SNAP ACTION BUTTON
                if (snapActionValue_2==LOW){
                        analogWrite(motorSpeedPin, output);
                }else{
                        analogWrite(motorSpeedPin, 0);
                }
        }
}
int Motor::mapFloat(float value, float in_min, float in_max, float out_min, float out_max)
{
        return (int) ((value - in_min) * (out_max - out_min) / (in_max - in_min)) + out_min;
}
```

Apêndice F

Main – Arduino

```
#include <Arduino.h>
#include <RNA.h>
#include <FeaturesExtractor.h>
#include <Motor.h>
Motor motor;
RNA rna;
FeaturesExtractor featuresExtractor;
void setup(){ }
float mapFloat(float value, float in_min, float in_max, float out_min, float out_max){
        return ((value - in_min) * (out_max - out_min) / (in_max - in_min)) + out_min; }
void loop(){
        float inputSignal = analogRead(5);
        float inputSignalRemapeado = mapFloat(inputSignal,0,1024,0,1);
        featuresExtractor.add(inputSignalRemapeado);
        rna.input[0] = featuresExtractor.output[0];
        rna.input[1] = featuresExtractor.output[1];
        rna.input[2] = featuresExtractor.output[2];
        rna.run();
        motor.output = rna.output[0];
        motor.act();
}
```