UnB - UNIVERSIDADE DE BRASÍLIA

FGA - FACULDADE UNB GAMA

PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA BIOMÉDICA

Desenvolvimento de um eletromiógrafo de superfície sem fio e comparação com o Delsys Bagnoli

Ithallo Junior Alves Guimarães

ORIENTADOR: Dr. Rinaldo André Mezzarane

DISSERTAÇÃO DE MESTRADO EM ENGENHARIA BIOMÉDICA

PUBLICAÇÃO: 138A/ 2021

BRASÍLIA: MARÇO – 2021

UnB - UNIVERSIDADE DE BRASÍLIA

FGA - FACULDADE UNB GAMA

PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA BIOMÉDICA

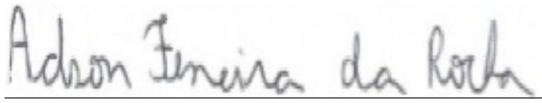
Desenvolvimento de um eletromiógrafo de superfície sem fio e comparação com o Delsys Bagnoli

Ithallo Junior Alves Guimarães

DISSERTAÇÃO DE MESTRADO SUBMETIDA AO PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA BIOMÉDICA DA FACULDADE GAMA DA UNIVERSIDADE DE BRASÍLIA, COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM ENGENHARIA BIOMÉDICA.

APROVADO POR:

Prof. Dr. Rinaldo André Mezzarane
(Orientador)



Prof. Dr. Adson Ferreira da Rocha
(Examinador Interno)

Prof. Dr. Jake Carvalho do Carmo
(Examinador Externo)

Brasília, 26 DE MARÇO DE 2021

FICHA CATALOGRÁFICA

Ithallo Junior Alves Guimarães
Desenvolvimento de um eletromiógrafo de superfície sem fio e comparação com o Delsys Bagnoli, [Distrito Federal] 2021.
138A. 124 p., 210 x 297 mm (FGA/UnB Gama, Mestre, Engenharia Biomédica, 2021).
Dissertação de Mestrado - Universidade de Brasília. Faculdade Gama. Programa de Pós-Graduação em Engenharia Biomédica.
1. Eletromiografia de Superfície. 2. Delsys Bagnoli-2 EMG System
3. Instrumentação Biomédica. 4. Instrumentação Eletrônica
I. FGA UnB Gama/ UnB. II. Desenvolvimento de um eletromiógrafo de superfície sem fio e comparação com o Delsys Bagnoli

REFERÊNCIA BIBLIOGRÁFICA

GUIMARÃES, I. J. A. (2021). Desenvolvimento de um eletromiógrafo de superfície sem fio e comparação com o Delsys Bagnoli. Dissertação de Mestrado em Engenharia Biomédica, Publicação 138A/2021, Programa de Pós-Graduação em Engenharia Biomédica, Faculdade Gama, Universidade de Brasília, Brasília, DF, 124 p.

CESSÃO DE DIREITOS

AUTOR: Ithallo Junior Alves Guimarães
TÍTULO: Desenvolvimento de um eletromiógrafo de superfície sem fio e comparação com o Delsys Bagnoli

GRAU: Mestre
ANO: 2021

---

ithallojunior@outlook.com
Brasília, DF – Brasil

DEDICATÓRIA

Dedico este trabalho à Professora Lourdes. Descanse em paz, queridinha.

AGRADECIMENTOS

Agradeço primeiramente à Deus, que, apesar das minhas constantes desistências, nunca desistiu de mim.

Agradeço aos meus pais por sempre me incentivarem a continuar, mesmo quando eu já queria desistir.

Agradeço à Dra. Lourdes Mattos Brasil (*in memoriam*), que me acolheu desde a graduação e que tanto me ensinou nesses muitos anos.

Agradeço ao meu orientador, Dr. Rinaldo André Mezzarane, que me recebeu de braços abertos em um momento de necessidade e me deu todo o suporte necessário a execução deste trabalho.

Agradeço ao Sandoval Tavares de Menezes, que me tirou inúmeras dúvidas sobre eletrônica e que sem o qual este trabalho não seria finalizado.

Agradeço ao Dr. Jake Carvalho do Carmo, cujos *insights* contribuíram imensamente para o meu trabalho.

Agradeço ao Laboratório de Processamento de Sinais e Controle Motor (LACOMOT), do qual faço parte, e também a todos seus membros pelo apoio e ajuda.

Agradeço aos extintos Laboratório de Engenharia e Inovação (LEI) e Laboratório de Informática em Saúde (LIS), dos quais tive a honra de fazer parte, bem como a todos seus participantes pelo suporte.

Por fim, agradeço a todos os muitos colegas e amigos que fiz durante a minha jornada dentro desta Universidade, os quais, de uma forma ou de outra, contribuíram na minha vida acadêmica, pessoal e profissional.

Muito obrigado!

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Código de Financiamento 001.

—Chame de intuição. É disso que se trata por enquanto. Mas pretendo chegar a uma conclusão sobre o assunto. Uma sequência de raciocínios válidos só pode levar à uma determinação da verdade e vou insistir até chegar lá.

Razão - Eu, robô (Isaac Asimov)

RESUMO

Desenvolvimento de um eletromiógrafo de superfície sem fio e comparação com o Delsys Bagnoli

Autor: Ithallo Junior Alves Guimarães
Orientador: Dr. Rinaldo André Mezzarane
Programa de Pós-Graduação em Engenharia Biomédica
Brasília, março de 2021

A eletromiografia de superfície é uma técnica amplamente utilizada nas ciências da saúde que permite o estudo de sinais advindos dos músculos de modo não invasivo. Tal técnica é extremamente promissora, visto que seu uso permite desde análises de movimento e auxílio ao diagnóstico de doenças até controle de próteses ativas. Por outro lado, também se trata de uma técnica que requer equipamentos pouco acessíveis, devido ao seu alto grau de refinamento e alto custo, fatos que aumentam a barreira de entrada para novos grupos que a desejem estudar. Assim, um dispositivo acessível, com o projeto disponibilizado publicamente, capaz de ser modificado (ou mesmo construído) pelos próprios pesquisadores e que seja capaz de coletar sinais de forma sem fio seria de grande valia para a comunidade acadêmica. Deste modo, este trabalho realizou a construção de tal dispositivo, bem como sua comparação com um dispositivo considerado como estado da arte para aquisição de sinais de eletromiografia de superfície, o *Delsys Bagnoli-2 EMG System*. Para esse fim, foram realizadas coletas de sinais, bem como a análise das interferências em seus respectivos espectros, das relações sinal-ruído e comparação de suas especificações, além da escrita de diversos programas para seu controle, configuração e uso e desenvolvimento de esquemático para padronizar seu projeto.

Palavras-chaves: Eletromiografia de Superfície, Delsys Bagnoli-2 EMG System, Instrumentação Biomédica, Instrumentação Eletrônica.

ABSTRACT

Development of a wireless surface electromyography device and comparison to the Delsys Bagnoli

Author: Ithallo Junior Alves Guimarães
Supervisor: Dr. Rinaldo André Mezzarane
Post-Graduation Program in Biomedical Engineering
Brasília, March of 2021.

Surface electromyography is a technique widely used in the health sciences which allows the study of signals coming from the muscles in a non-invasive way. Such technique is extremely promising, since its adoption allows uses from movement analysis and aided diagnosis of diseases to active prostheses control. On the other hand, it is also a technique that requires hard-to-obtain equipment, due to its high degree of refinement and high cost, facts that increase the entry barrier for new groups that want to study it. Thus, an accessible device, with a project made available publicly, capable of being modified (or even built) by the researchers themselves, and that is able to collect signals wirelessly would be of great value to the academic community. Hence, this work carried out the construction of such a device, as well as its comparison with a device considered as the state of the art for acquiring surface electromyography, the *Delsys Bagnoli-2 EMG System*. Along these lines, signal acquisitions were made, in addition to the analysis of the interferences in their respective spectra, the analysis of the signal-to-noise ratios and the comparison of their specifications, in addition to writing several programs for its control, configuration, and use, and a schematic to standardize the project.

Key-words: Surface Electromyography, Delsys Bagnoli-2 EMG System, Biomedical Instrumentation, Electronic Instrumentation.

SUMÁRIO

LISTA DE TABELAS

LISTA DE FIGURAS

LISTA DE ABREVIATURAS E SIGLAS

ADC	*Analog to Digital Converter* - Conversor Analógico-Digital
CI	Circuito Integrado
CMRR	*Common Mode Rejection Ratio* - Taxa de Rejeição do Modo Comum
CSV	*Comma-separated Values* - Valores Separados por Vírgula
DC	*Direct Current* - Corrente Direta
ECG	Eletrocardiograma
EMG	Eletromiografia
DAQ	*Data Acquisition Device* - Dispositivo de Aquisição de Dados
DFT	*Discrete Fourier Transform* - Transformada Discreta de Fourier
FFT	*Fast Fourier Transform* - Transformada Rápida de Fourier
IDE	*Integrated Development Environment* - Ambiente de Desenvolvimento Integrado
Kbps	*Kilo Bits per Second*
LED	*Light-emitting Diode* - Diodo Emissor de Luz
Mbps	*Mega Bits per Second*
MDF	*Median frequency* - Frequência Mediana
MNF	*Mean Frequency* - Frequência Média
RMS	*Root Mean Square* - Raiz do Valor Quadrático Médio ou Valor Eficaz
sEMG	*Surface Electromyography* - Eletromiografia de Superfície
SPI	*Serial Peripheral Interface*
PSRR	*Power Supply Rejection Ratio* - Taxa de Rejeição da Fonte de Alimentação
UART	*Universal Asynchronous Receiver/Transmitter*
UM	Unidade Motora
USB	*Universal Serial Bus* - Porta Serial Universal

LISTA DE SÍMBOLOS

A	Ampere
Ag	Prata
AgCl	Cloreto de prata
Ah	Ampere-hora
B	*Byte*
cm	Centímetro
dB	Decibel
g	Grama
Hz	Hertz
k	Kilo (10^3 ou 2^{10}, quando relacionado a computação)
M	Mega (10^6 ou 2^{20}, quando relacionado a computação)
μ	Micro (10^{-6})
m	Mili (10^{-3})
mm	Milímetro
Ω	Ohm
s	Segundo
V	Volt

1 INTRODUÇÃO

A Eletromiografia de Superfície (sEMG, de *Surface Electromyography*) é uma técnica amplamente utilizada nas ciências da saúde. O sinal de sEMG é definido pela soma dos potenciais de ação volume-conduzidos, quando adquiridos por eletrodos sobre a pele (acima do músculo), seja durante uma contração voluntária ou em resposta a um estímulo externo (mecânico ou elétrico) [1, 2].

De forma geral, a Eletromiografia (EMG) abrange técnicas de aquisição, amplificação, filtragem, conversão analógico-digital, análise e interpretação da atividade elétrica do músculo [2]. Ela é utilizada em estudos clínicos, neurofisiológicos e biomecânicos, uma vez que diversos aspectos do controle motor (executados pelo sistema nervoso central) podem ser analisados [3, 4, 5, 6]. A análise do padrão de interferência do sinal de sEMG, por exemplo, o qual representa a atividade assíncrona das unidades motoras (UM) recrutadas de forma voluntária, é bastante útil para desvendar o padrão da atividade muscular em diversos movimentos e atividades motoras [2].

Técnicas de processamento de sinais, tanto no domínio do tempo quanto no domínio da frequência, podem ser empregadas para se inferir a estratégia utilizada pelo sistema nervoso central em diversos contextos motores ou patologias [7]. Atualmente há duas formas para adquirir o sinal de EMG: por meio de eletrodos fixados à pele (sEMG) e a intramuscular, por meio de eletrodos de agulha. A primeira destas já foi apresentada anteriormente de forma geral. Já a segunda, representa um procedimento mais invasivo, utilizado para estudar o comportamento de uma única UM ou de um pequeno grupo de UMs [8, 9]. Deste modo, eletrodos intramusculares possibilitam o estudo de aspectos bastante específicos da atividade da UM, os quais não podem ser facilmente observados com a sEMG. Por outro lado, a sEMG possui a grande vantagem de permitir uma avaliação ampla da atividade muscular, além do fato de não ser invasiva. Isso permite uma compreensão mais completa da interação entre diferentes grupos musculares que são relevantes para a realização de movimentos coordenados [10].

1.1 JUSTIFICATIVA

Os altos custos de sistemas de EMG já consolidados no mercado impossibilitam a criação e expansão de grupos de pesquisa que tenham como foco esse tema. De forma especial, aqueles pesquisadores que vivem em países com poucas (ou nenhuma) empresas nacionais que produzam equipamentos de boa qualidade enfrentam grandes dificuldades em importar esses equipamentos, seja por causa dos custos de conversão de moeda e os impostos de importação envolvidos ou outro motivo qualquer. Mesmo quando considerada a pos-

sibilidade de aquisição de um bom equipamento de forma mais fácil, observa-se que uma versão sem fios é quase sempre muito mais cara.

Um sistema de EMG sem fios, com baixo nível de ruído, é de grande valia para pesquisas com o uso desses sinais durante a execução de movimentos (análise de marcha, por exemplo). Além disso, o uso de um aparelho pequeno, com pequenos eletrodos e cabos curtos é extremamente desejável para se reduzir a influência do sistema nas ações do voluntário. Por fim, é bem provável que um dispositivo de custo acessível seja bem aceito por pesquisadores em seus estudos da ação muscular ou mesmo para fins didáticos.

Pesquisas anteriores já propuseram sistemas de aquisição sem fio de EMG [11, 12, 13]. Contudo, estes estudos variam na forma de aquisição e transmissão do sinal. Também se acredita que o sistema aqui desenvolvido possui um custo mais acessível e maior facilidade de construção e programação, devido aos componentes e linguagem de programação escolhidos.

1.2 OBJETIVOS

1.2.1 Objetivos gerais

Este trabalho visa a continuação do trabalho de Guimarães [14], de desenvolvimento de um eletromiógrafo de superfície sem fios, objetivando a melhoria da qualidade dos sinais captados, da eletrônica do aparelho e a comparação com um equipamento já consolidado no mercado, o *Delsys Bagnoli-2 EMG System* (Delsys Bagnoli-2). Não se propõe para o momento a elaboração de um dispositivo de qualidade superior aquela do Delsys Bagnoli-2, mas sim que o use como referência. O presente trabalho também visa a disponibilização pública de esquemáticos e códigos aqui desenvolvidos, para se fomentar (e facilitar) a pesquisa com a sEMG.

1.2.2 Objetivos específicos

- Construir um eletromiógrafo de superfície sem fios;
- Elaborar esquemático do circuito em ferramenta de desenho assistido por computador;
- Realizar coletas de sinais em ambos dispositivos;
- Comparar os sinais obtidos entre o dispositivo e o Delsys Bagnoli-2 com base em quantificadores nos domínios do tempo e da frequência;
- Comparar as especificações dos dispositivos;
- Disponibilizar publicamente esquemáticos e códigos.

2 REFERENCIAL TEÓRICO

2.1 SEMG

O sinal de EMG (e por conseguinte o de sEMG) é gerado pela atividade elétrica das fibras musculares que se encontram ativas durante uma contração [1].

Considera-se neste trabalho a sEMG como a captação de sinais com o uso de um par de eletrodos de superfície e não a EMG de alta densidade, que se utiliza de uma matriz de eletrodos com múltiplos contatos [15].

2.1.1 *Crosstalk*, preparação do local e algumas fontes de interferências

Para casos específicos, a amplitude da EMG é proporcional à força empregada de forma voluntária. Essa relação é, até certo ponto, linear, apesar de depender fortemente do músculo em questão [1]. À medida que a força aplicada aumenta, a taxa de disparos e o número de UMs recrutadas aumenta. Contudo, em contrações dinâmicas (ou mesmo em contrações isométricas), o eletrodo de superfície pode estar com um posicionamento inadequado em relação às fibras musculares cuja atividade elétrica ele deveria medir. Isso, em conjunto com a presença de tecido adiposo abaixo dos eletrodos (o qual pode acabar atuando com um filtro passa-baixas) e o *crosstalk* entre os músculos adjacentes, afeta as características do sinal captado [16].

Por este motivo, a preparação do local em que os eletrodos seriam posicionados é crucial, um exemplo dessa preparação seria a limpeza da área com algodão embebido em álcool. Esse procedimento serve para reduzir a impedância entre os eletrodos e a pele. Em compensação, a impedância do pré-amplificador deve ser de 10 a 100 vezes maior que aquela entre o contato pele-eletrodo, ou seja, quanto maior a impedância do pré-amplificador, melhor. Indo além desses procedimentos também se deve considerar a tricotomia do local, especialmente em condições de alta umidade ou para pessoas que suam bastante [16, 17].

Uma fonte comum de ruído para a sEMG é o sinal de eletrocardiograma (ECG), já que este é um sinal bem conciso e com amplitude superior a dos músculos. Ele é claramente visível em quase todos os sinais captados no torso. Uma forma de se minimizar essa interferência é colocando os eletrodos próximos entre si e utilizar um filtro entre 100 e 200 Hz (rejeita-faixas). O sinal de ECG também pode ser reduzido com uma boa preparação da pele ou mudança do local do eletrodo de referência [16, 17].

Artefatos de movimentos também podem afetar a aquisição de sinais de sEMG. Eles podem ser observados em mudanças bruscas observadas nos potenciais do sinal de sEMG

que ocorrem quando os eletrodos se movem sobre a pele, gerando potenciais elétricos inconvenientes. Um filtro passa-altas é normalmente utilizado para evitar tais interferências, já que as suas harmônicas normalmente se encontram entre 0 e 20 Hz [1, 16].

Outra fonte comum de interferência é a frequência da rede elétrica, que gera sinais nas frequências de 50 Hz ou 60 Hz, além de seus harmônicos [1, 16]. Uma forma comum de tentar evitá-los é com o uso de um filtro *notch*, porém esta não é uma boa prática, já que pode retirar componentes importantes do sinal de sEMG [1]. Outra forma seria aterrar todos os dispositivos (especialmente aqueles com motores elétricos) do ambiente, mudar de tomada e evitar colocar múltiplos dispositivos na mesma tomada [17].

2.1.2 Banda e amplitude

De acordo com De Luca [18], o sinal de EMG varia entre 0 e 10mV (pico a pico) ou entre 0 e 1,5mV de valor eficaz (RMS, de *Root Mean Square*). Para a banda, é dito por Prutchi e Norris [19] que esse sinal possui componentes relevantes entre 2 e 500 Hz.

Já Konrad [17] considera que o sinal de sEMG tem uma amplitude de ± 5 mV (10 mV pico a pico), concordando com De Luca [18], porém acredita que esse valor é válido somente para atletas. Também é afirmado por Konrad [17] que, tipicamente, o sEMG possui frequências entre 6 e 500 Hz, tendo seu maior potencial entre cerca de 20 e 150 Hz.

Merletti e Parker [1] reportam que os filtros dos equipamentos utilizados para a aquisição de sEMG se encontram entre 10 - 20 Hz até 400 - 450 Hz, de modo a se evitar artefatos de movimentos, que normalmente possuem harmônicas entre 0 e 20 Hz.

2.1.3 Equivalência de sinais de sEMG

A comparação entre os sinais de sEMG, de forma geral, é algo difícil de ser feito. Uma das formas de fazer a equivalência entre sinais de sEMG entre diferentes dispositivos, embora não exata, pode ser vista na Tabela 1, que exemplifica como colocar equivalência entre sinais de diferentes dispositivos por meio de um fator de correção [16].

Os fatores de correção apresentados na Tabela 1 foram calculados com o uso de um sinal comum para todos dispositivos, nesse caso um ruído rosa (que possui potências iguais para cada década, em seu espectro [20]) com frequências dentro dos limites daquelas dos músculos e com amplitude de 5 μV ("músculo sintético"). Após isso, considerou-se um dispositivo como referência (para o caso da tabela, o J&J M-501) e foi feita a média entre 5 valores RMS de cada instrumento. Desta forma, foi possível estabelecer um padrão para equivalência dos sinais RMS entre diferentes dispositivos (Equação 1) [16].

$$Razão = \frac{V_{instrumento}[RMS]}{V_{referência}[RMS]} \tag{1}$$

Tabela 1 – Razões entre diferentes dispositivos, de modo a permitir a equivalência. Adaptado de Criswell [16].

Equipamento	Fator de correção
Referência	1,00
DMS 4000 (25–450 Hz)	1,17
Flexiplus (100–1000 Hz)	1,07
J&J M-53 (100–200 Hz)	1,00
J&J M-501 (100–200 Hz)	1,00
J&J M-501 (25–1000 Hz)	1,41
J&J I-410 Myoamp (controlado por *software*)	1,31
Norodyne 8000 (25–450 Hz)	1,67
Physiotech 4000 (25–450 Hz)	1,17
SRS: Orion (100–1000 Hz)	1,07
Thought Technology: Myotrac (100–200 Hz)	1,27
Thought Technology: Myotrac (20–500 Hz)	1,75
Verimed VStim 2 (20–1000 Hz)	1,26
Verimed Myo3 Dual (20–1000 Hz)	1,23

2.2 DECIBEL

Decibel (dB) representa uma unidade de medida para se comparar amplitudes relativas entre dois sinais distintos usando uma medida logarítmica. A medida logarítmica é utilizada uma vez que, para boa parte dos casos, se está lidando com razões da casa dos milhões. O decibel também representa um décimo de uma outra antiga unidade que caiu em desuso, conhecida como bel [20].

Por definição, a razão entre dois sinais em decibéis é dada pela Equação 2 [20]:

$$dB = 10log_{10}\frac{P_2}{P_1} \tag{2}$$

Onde P_1 e P_2 são as potências dos sinais.

Também há a possibilidade de se definir essa razão em termos da amplitude do sinal, o que pode ser mais usual, conforme a Equação 3 [20]:

$$dB = 20log_{10}\frac{A_2}{A_1} \tag{3}$$

Onde A_1 e A_2 representam as amplitudes dos sinais.

Uma forma prática de ser entender decibéis por meio da Equação 2 é observar que quando, um sinal dobra em potência um aumento de 3 dB é observado. Uma queda de 3 dB é observada quando ele cai para a metade da potência.

Já para Equação 3, uma prática útil é observar que tem-se o valor de 6 dB para os casos anteriores, de modo análogo. Note-se também que um ganho de 20 dB equivale a um

aumento de 10 vezes na amplitude do sinal, um ganho de 40 dB equivale a um aumento de 100 vezes e assim por diante. O inverso vale para as perdas e quedas dos sinais.

2.3 AMPLIFICADORES OPERACIONAIS

Amplificadores operacionais são elementos de circuito ativo, compostos internamente por resistores, capacitores e diodos. Eles possuem a habilidade de executar operações matemáticas (i.e. adição, subtração, multiplicação, divisão, diferenciação e integração), quando em conjunto com outros componentes externos (e.g. resistores e capacitores), sendo essa habilidade o motivo de seu nome [21].

Nesta seção são descritos os conceitos de CMRR e PSRR e alguns tipos de amplificadores necessários a este trabalho.

2.3.1 CMRR e PSRR

O valor da taxa de rejeição do modo comum (CMRR, de *Common Mode Rejection Ratio*) representa o quanto a tensão de saída varia com relação a tensões comuns a ambas entradas do circuito. Os valores de CMRR vão, tipicamente, de 70 dB (mínimo) até em torno de 130 dB para amplificadores de precisão [20].

A Figura 1 explica como funciona o CMRR na prática.

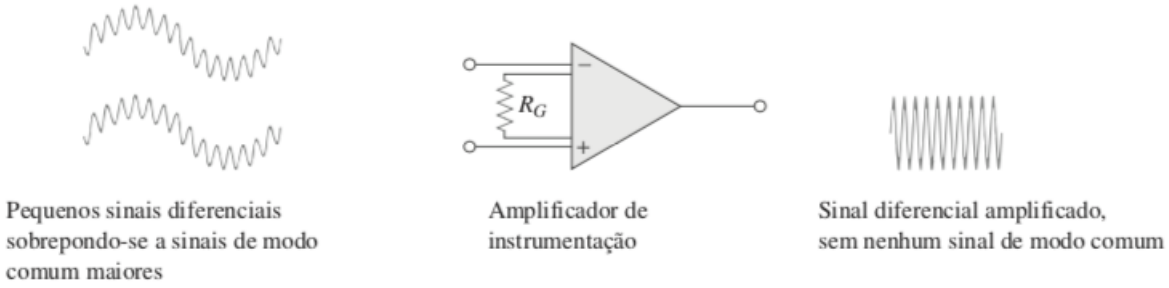


Figura 1 – Funcionamento do CMRR [21].

O valor da taxa de rejeição da fonte de alimentação (PSRR, de *Power Supply Rejection Ratio*) representa o quanto a tensão de saída varia com variações da alimentação. Seu valor se encontra tipicamente entre 60 e 80 dB, mas pode chegar a 130 dB para certos circuitos integrados (CI) [20].

De forma geral, o PSRR costuma ser muito pior em uma das linhas de alimentação que na outra (em alimentação simétrica). Uma forma de se defender de problemas relacionados a PSRR é adicionando um filtro resistor-capacitor às linhas de alimentação [20].

2.3.2 Seguidor unitário

O seguidor unitário ou *buffer* (Figura 2) possui como principal característica ter a saída igual a sua entrada. Esse é um circuito muito utilizado para acionar cargas de baixa impedância, em conversores analógico para digital (ADC, de *Analog to Digital Converter*) e *buffers* de tensões de referência [22].

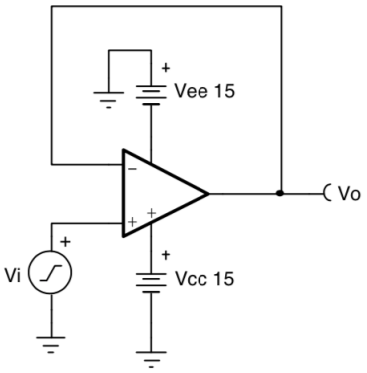


Figura 2 – Exemplo de seguidor unitário. Adaptado de Green e Wells [22].

2.3.3 Amplificador inversor

Um exemplo de amplificador inversor pode ser visto na Figura 3. Ele inverte a polaridade do sinal colocado em sua entrada ao mesmo tempo que o amplifica, como descrito pela Equação 4 [21].

$$v_o = -\frac{R_f}{R_1} v_i \tag{4}$$

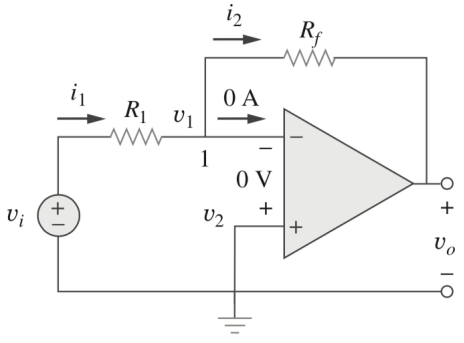


Figura 3 – Exemplo de amplificador inversor [21].

2.3.4 Amplificador de instrumentação

A Figura 4 mostra o arranjo dos amplificadores de instrumentação. Esses amplificadores recebem esse nome por seu largo uso em sistemas de medição, sendo um de seus usos típicos sistemas de aquisição de dados [21].

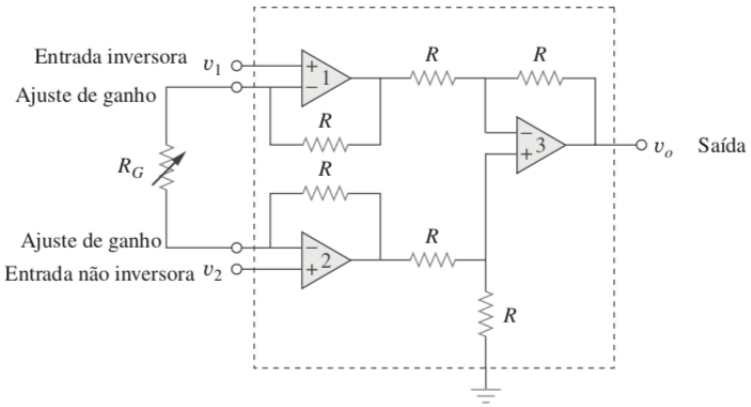


Figura 4 – Exemplo de amplificador de instrumentação. Adaptado de Alexander e Sadiku [21].

Ele é formado geralmente por 3 amplificadores operacionais e sete resistores. Caso todos seus resistores tenham o mesmo valor (R), exceto externo (ajuste de ganho, R_G), sua saída pode ser dada pelas Equações 5 e 6 [21].

$$v_o = A_v(v_2 - v_1) \tag{5}$$

$$A_v = 1 + \frac{2R}{R_G} \tag{6}$$

Assim, um amplificador de instrumentação tem seu ganho definido por um resistor externo e é capaz de amplificar pequenas tensões de forma diferencial e cancelar grandes sinais comuns as suas entradas (grande CMRR). Devido ao seu uso frequente, eles podem ser encontrados em um único CI [21].

2.4 TAXA DE AMOSTRAGEM

O critério de Nyquist estabelece que um sinal, ao ser digitalizado, não deve conter outros sinais com frequências que ultrapassem a metade da frequência de amostragem, caso isso não seja respeitado, pode ocorrer *aliasing* [20].

Isso é geralmente feito com um uso de um filtro passa-baixas (filtro *anti-aliasing*) em que a frequência de corte garanta uma atenuação para os sinais que ultrapassem o critério de Nyquist. A Figura 5 mostra como o *aliasing* atua em um sinal [20].

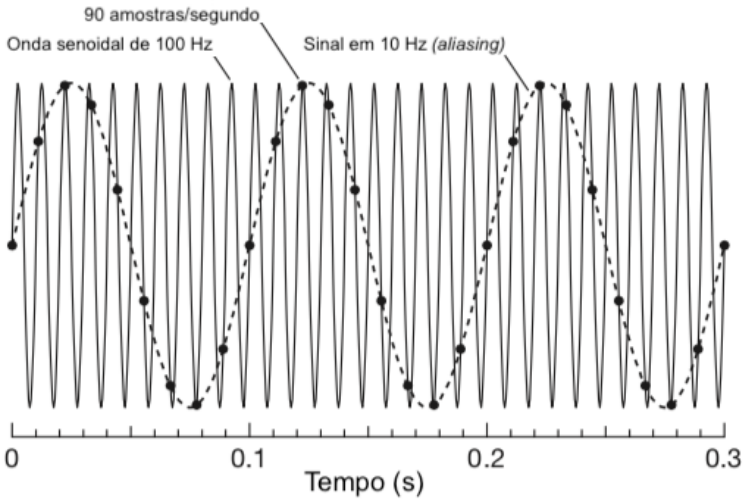


Figura 5 – Sinal com *aliasing*. Adaptado de Horowitz e Hill [20].

2.5 FILTROS

Filtros são definidos por Alexander e Sadiku [21] como circuitos que são projetados de modo a se deixar passar sinais com frequências desejadas e rejeitar ou atenuar outras componentes espectrais, sendo assim um dispositivo seletivo de frequências.

Os filtros podem ser passivos ou ativos. Um filtro passivo é composto exclusivamente por resistores, capacitores e indutores. Já um filtro ativo é aquele que possui algum elemento ativo (e.g. amplificador operacional ou transistor) [21].

Filtros ativos possuem certas vantagens com relação aos filtros passivos, sendo elas [21]:

- São normalmente menores e mais baratos por não precisarem de indutores, o que lhes permite serem construídos em CIs;
- São capazes de dar ganho ao amplificador, mesmo sendo capazes de fornecer a mesma resposta de frequência dos filtros passivos;
- Podem ser associados a *buffers* de modo a se isolar cada estágio do filtro dos efeitos de impedância de carga da fonte, permitindo também que cada estágio seja projetado de forma independente.

Apesar disso, filtros ativos são menos confiáveis e menos estáveis, tendo um limite prático para sua grande maioria em cerca de 100 kHz.

Além desses dois, também existem filtros digitais, que são aqueles que só conseguem operar com sinais digitalizados. Um filtro digital pode fazer tudo aquilo que um filtro físico pode além de poder ser simulado com um grau arbitrário de precisão [23].

Segundo Alexander e Sadiku [21] existem quatro tipos de filtros, sendo eles passa-baixas, passa-altas, passa-faixas e rejeita-faixas. Este trabalho somente se focará nos tipos passa-baixas e passa-altas, aqui utilizados.

Os filtros passa-baixas são aqueles projetados para deixar passar frequências abaixo de sua frequência de corte, já os filtros passa-altas são projetados para somente deixar passar frequências acima de sua frequência de corte [21].

2.6 CONCEITOS EM SINAIS E PROCESSAMENTO

2.7 VALOR RMS

O valor RMS adveio da necessidade de se medir a eficácia de uma fonte de tensão ou de corrente para a liberação de potência para uma carga resistiva [21].

Ele pode ser definido, para um sinal periódico, como a raiz do valor quadrático médio (Equação 7) [21].

$$X_{RMS} = \sqrt{\frac{1}{T}\int_0^T x^2 dt} \tag{7}$$

Outra forma de se calcular o valor RMS, considerando-se os pontos discretos e não contínuos, pode ser vista na Equação 8 [24]:

$$Q_x = \sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2} \tag{8}$$

2.8 SNR

O valor da relação sinal-ruído (SNR, de *Signal-to-noise Ratio*) pode ser medido em decibéis e simplesmente refere-se a razão entre a tensão RMS do ruído presente com relação ao sinal desejado [20].

Assim, o SNR pode ser definido pela Equação 9 [20]:

$$SNR = 20log_{10}\frac{V_{sinal}}{V_{ruído}} \tag{9}$$

2.9 FFT

A transformada rápida de Fourier (FFT, de *Fast Fourier Transform*) refere-se a um conjunto de algoritmos que realizam a implementação da transformada discreta de Fourier (DFT, de *Discrete Fourier Transform*) de forma computacionalmente eficiente [25].

2.10 FREQUÊNCIA MÉDIA E MEDIANA

A frequência média (MNF) e a frequência mediana (MDF) são os descritores espectrais mais comumente utilizados, de modo que pouquíssimos estudos se utilizam de momentos de ordem superior para descrever o espectro do sinal de EMG [1].

A MNF, como seu nome indica, é a frequência média, calculada como a soma do produto entre o espectro de potência do sinal de EMG (P) e a frequência (f), divididos pela soma do espectro de potência (Equação10). Já a MDF representa o valor de frequência para o qual o espectro de potência do sinal de EMG se divide em duas partes de igual amplitude (Equação 11) [26].

$$MNF = \frac{\sum_{j=1}^{M} f_j P_j}{\sum_{j=1}^{M} P_j} \tag{10}$$

$$\sum_{j=1}^{MDF} P_j = \sum_{j=MDF}^{M} P_j = \frac{1}{2} \sum_{j=MDF}^{M} P_j \tag{11}$$

3 PRINCIPAIS TECNOLOGIAS E COMPONENTES

Este capítulo aborda as principais tecnologias e componentes utilizados bem como certas modificações das mesmas, quando aplicável.

3.1 ARDUINO

Arduino é uma plataforma eletrônica que possui seu código aberto, sendo feita com *hardware* e *software* de fácil utilização, o que acelera o processo de desenvolvimento [27]. A Figura 6 mostra o ambiente de desenvolvimento integrado (IDE, de *Integrated Development Environment*) do Arduino no MacOS.

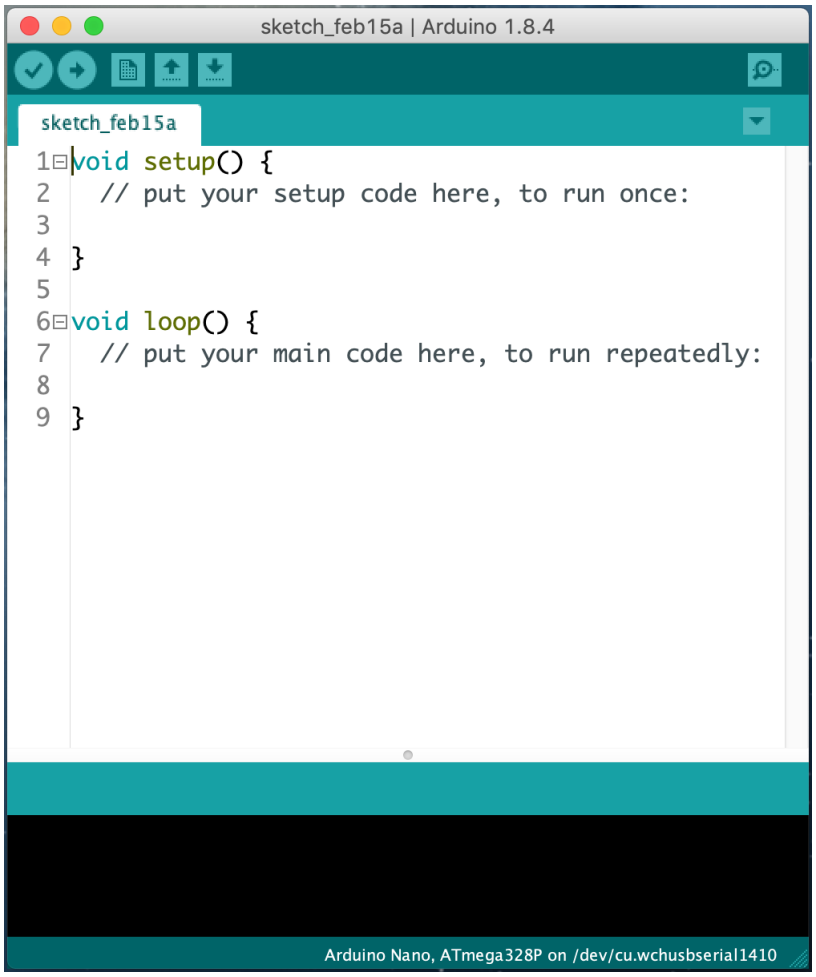


Figura 6 – IDE do Arduino.

A plataforma possui a vantagem de contar com uma grande comunidade de criadores ao redor do mundo, composta desde estudantes, artistas, programadores até profissionais. Assim, ela consegue disponibilizar publicamente uma grande quantidade de conhecimento, ajuda e exemplos de projetos através da sua comunidade *online* [27].

Há uma grande variedade de outros microcontroladores e plataformas de microcontroladores que poderiam ter sido utilizados, porém o Arduino (exemplo na Figura 7) oferece a vantagem, além de sua grande comunidade, de simplificar o processo de trabalho com microcontroladores. Além disso, há vantagens por ser de baixo custo (quando comparado com outras plataformas), de possuir um ambiente de trabalho simples, limpo e flexível e de ser multiplataforma, permitindo que se trabalhe com ele em ambientes Linux, Windows e MacOS, enquanto a maioria dos outros microcontroladores requer um ambiente Windows para desenvolvimento [27].

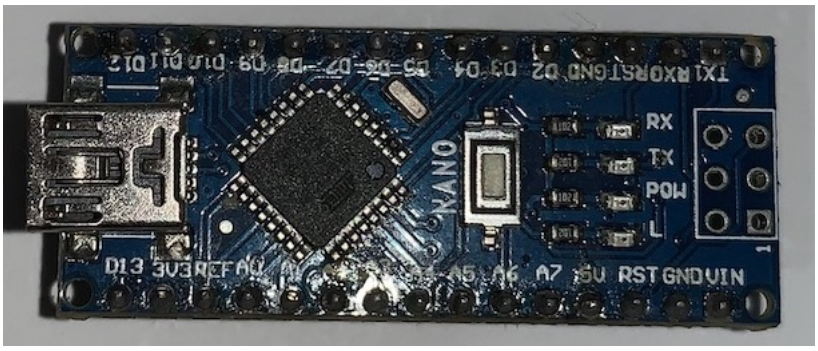


Figura 7 – Arduino Nano, modelo aqui utilizado.

3.2 PYTHON

Python é uma linguagem de programação orientada a objetos de código aberto gratuita, comparável a outras grandes e conhecidas linguagens, como Java ou Ruby. Ela possui a vantagem de ter uma sintaxe clara, elegante e simples, o que a faz ideal para um desenvolvimento rápido, além de simplificar o processo de prototipagem e de ajustes de código *on the fly*, sem comprometer a capacidade de manutenção do código [28].

Sua biblioteca padrão já é, por si só, bastante completa, tendo suporte nativo para atividades que vão desde se conectar a servidores *web*, buscas com expressões regulares e leitura/escrita de arquivos. Ademais, Python é capaz de ser executado em ambientes que vão de Windows, Linux e MacOS até Android e iOS, sendo verdadeiramente multiplataforma [28].

3.2.1 Principais bibliotecas e ferramentas utilizadas

Durante o desenvolvimento, algumas bibliotecas e ferramentas relativas ao Python se fizeram essenciais para consolidar tarefas, como comunicação, leitura, processamento de sinais, teste, validação e visualização de dados. Sendo elas:

- **Numpy** [29]: *arrays* de dados, FFTs, leitura/escrita de dados e processamento de sinais;

- **Scipy** [30]: elaboração de filtros digitais;
- **Matplotlib** [31]: visualização de dados em tempo real e geração de gráficos;
- **pySerial** [32]: comunicação entre o Arduino e o computador;
- **IPython** [33] e **Jupyter** [34]: testes rápidos de pedaços de código, processamento dos sinais e visualização de dados.

3.3 ATTINY85

O ATtiny85 é um microcontrolador de 8 *bits* de alto desempenho, baixo consumo, que é compatível com a plataforma Arduino. Ele possui um ADC de 10 *bits* de 4 canais, capaz de operar utilizando um sinal de *clock* interno [35]. A Figura 8 mostra o esquema de conexões para o ATtiny85.

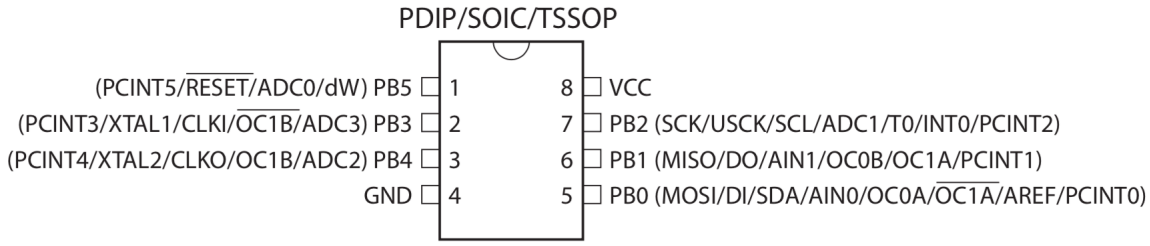


Figura 8 – Esquemático do ATtiny85. Adaptado de ATMEL [35].

Esse microcontrolador foi escolhido por seu tamanho, baixo consumo, capacidade de operar sem a necessidade de sinal de *clock* externo e por ser operável com segurança a 3,3 V e uma frequência de 8 MHz. Ele também é capaz de operar numa faixa de 1,8 V a 5,5 V com até 20 MHz em sua versão mais abrangente (Figura 9) [35].

Isso o fez ideal para um dispositivo que almejava ser leve, compacto e capaz de operar por longos períodos em baixas tensões conectado a pilhas ou baterias. Além disso, seu baixo custo e compatibilidade com a plataforma Arduino o tornam bastante atrativo, devido ao desenvolvimento facilitado.

O ATtiny85 também conta com 8 kB de memória para programas, capazes de suportar até 10.000 ciclos de escrita/reescrita, 512 *bytes* de memória não volátil, capazes de suportar até 100.000 ciclos de escrita/reescrita, e 512 *bytes* de memória volátil [35].

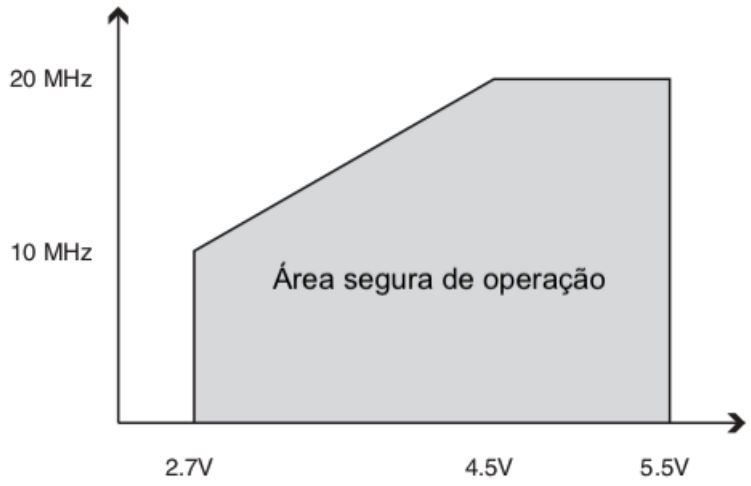


Figura 9 – Frequência máxima x tensão de alimentação no ATtiny85. Adaptado de ATMEL [35].

3.3.1 ADC

O ADC interno do ATtiny85 usa aproximações sucessivas para realizar conversão e, como dito anteriormente, possui 10 *bits* de resolução e 4 entradas. Ele possui tensões internas de referência selecionável entre 1,1 V e 2,56 V, além da própria tensão de alimentação ou outra referência externa [35]

Ademais, ele conta com um tempo de conversão entre 65 μs e 260 μs [35]. Isso o torna apropriado para a coleta de sinais de sEMG, já que a frequência máxima de 500 Hz do mesmo [19] resulta em um tempo entre amostras de 1000 μs, mais que suficiente para aquisição, ao se considerar o critério de Nyquist [20]. Essa taxa de amostragem seria até mesmo suficiente para uma taxa de 2 kHz (500 μs entre amostras).

3.3.2 Oscilador interno

Devido a quantidade reduzida de pinos disponíveis para o ATtiny85 (8 pinos totais, 5 disponíveis, após uso de alimentação e *RESET*) houve a necessidade de se utilizar o sinal de *clock* do oscilador interno. Ele vem calibrado de fábrica para utilizar um sinal de *clock* de 8,0 MHz, com uma acurácia de ± 10% a 25°C e 3 V de alimentação [35].

ATMEL [35] esclarece que há possibilidade de calibrar oscilador para ± 1% de acurácia. Essa variação pode ser irrelevante para grande parte das aplicações, porém, ela se fez necessária após verificada a variação com um osciloscópio.

Essa calibragem pode ser feita por meio da alteração do valor de um registrador interno do ATtiny85, chamado OSCCAL [35]. O ajuste do valor pode ser feito com o uso de uma frequência conhecida, um osciloscópio ligado ao ATtiny85 e outro Arduino ligado ao computador, sem código e com os pinos de sua porta serial conectados ao ATtiny85

(atuando com *bypass* para o ATtiny85).

Todo ATtiny85 aqui utilizado foi calibrado conforme descrito anteriormente. O código utilizado para essa ação pode ser visto no Apêndice B.1.

3.4 INA118

O INA118 é um amplificador operacional de instrumentação de precisão de propósito geral, que apresenta baixo consumo, facilidade de ajuste de ganho, grande faixa de alimentação (de $\pm 1,35$ V até ± 18 V), alto CMRR (110 dB, mínimo) e grande banda de operação (70kHz, com um ganho de 100) [36].

O ganho no INA118 pode ser ajustado com apenas um resistor entre seus pinos 1 e 8, conforme pode ser visto na Figura 10, que também apresenta seu esquemático [36]. A fórmula para cálculo de seu ganho encontra-se replicada na Equação 12, que é similar a Equação 6, discutida anteriormente.

$$G = 1 + \frac{50k\Omega}{R_G} \tag{12}$$

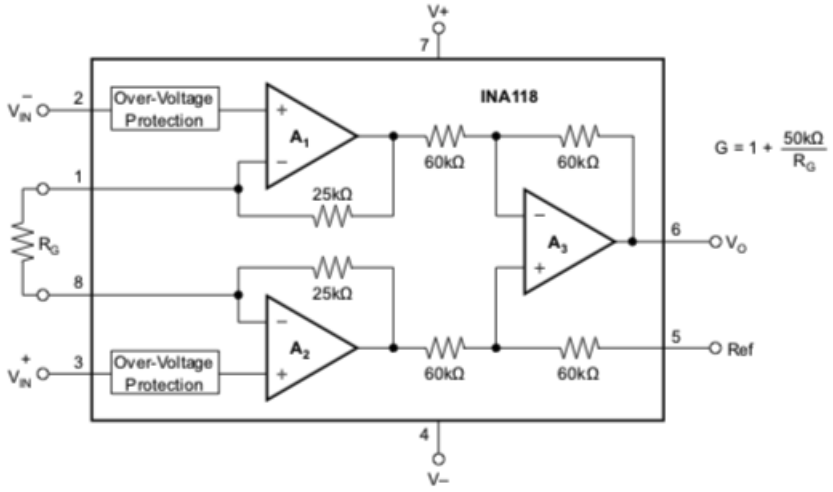


Figura 10 – Esquemático simplificado e ganho do INA118 [36].

O INA118 foi escolhido devido a sua disponibilidade, uma vez que se trata de um amplificador de precisão que possui um custo mais elevado e uma certa dificuldade de aquisição.

3.5 NRF24L01+

O NRF24L01+ é um transceptor de baixo consumo que opera na faixa de 2,4 GHz capaz de transmitir dados com uma taxa de até 2 Mbps, operável entre 1,9 V e 3,6 V e que é utilizado desde *mouses* e teclados a *headsets* de VoIP [37].

O transceptor pode ser operado com o uso de um microcontrolador via protocolo SPI, além de possuir entradas capazes de tolerar 5 V, o que facilita a conexão no receptor, sem necessidade de conversão de nível de tensão (de 5 V para 3,3 V e vice-versa) [37].

Ele opera com um protocolo interno proprietário, que permite o uso dinâmico de pacotes com tamanhos entre 1 e 32 *bytes*, controle automático de transação (retransmissão automática e confirmação de recebimento) e de pacotes. No mais, o NRF24L01+, com seu protocolo proprietário, permite o uso de uma topologia estrela, com recepção de dados de até 6 dispositivos paralelamente [37].

A Figura 11 mostra dois módulos NRF24L01+, um com uma antena de longo alcance e outro sem.

Figura 11 – Exemplos de NRF24L01+.

4 ESTADO DA ARTE

Por uma questão de disponibilidade e reconhecimento no mercado, considera-se o Delsys Bagnoli-2 como o exemplo de estado da arte, ou mesmo de padrão ouro para a sEMG. Isso ocorre de tal modo que a comparação com os sinais captados por ele é um dos principais focos deste trabalho.

Assim, este capítulo se dedica a abordar características do Delsys Bagnoli-2 e dispositivo associado necessário para a execução da coleta dos sinais de EMG.

4.1 DELSYS BAGNOLI-2

O Delsys Bagnoli-2 é um sistema projetado para coleta de sinais de EMG de forma confiável e sem complicações com dois canais independentes. Seus eletrodos foram especificamente projetados para realizar a sEMG de forma ótima, ao mesmo tempo que rejeita interferências de movimentos dos seus cabos [38].

A Tabela 2 traz informações técnicas a respeito do dispositivo utilizado. As Figuras 12, 13 e 14 mostram o dispositivo em questão em ângulos diferentes.

Tabela 2 – Especificações do Delsys Bagnoli-2. Adaptado de DELSYS [38].

Característica	Valor
Ganho máximo por canal:	100, 1.000 ou 10.000
Banda de operação dos canais:	20 ± 5 Hz a 450 ± 50 Hz
Ordem dos filtros:	4^a ordem (80 dB/década)
Alimentação:	bateria de 9 V
Tensão máxima da saída:	± 5 V

Figura 12 – Entradas do Delsys Bagnoli-2.

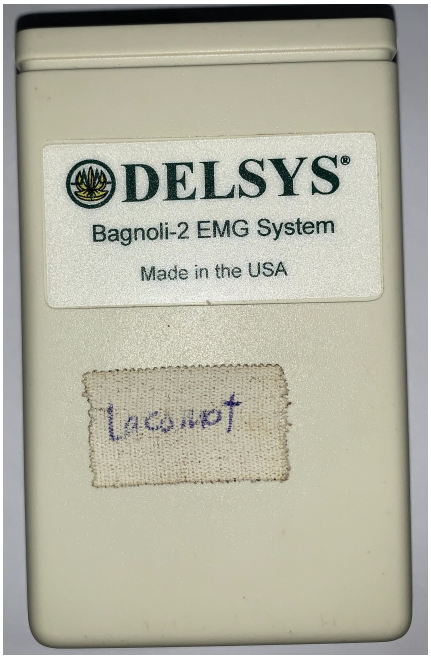


Figura 13 – Vista frontal do Delsys Bagnoli-2.

Figura 14 – Ajustes de ganho e chave liga/desliga do Delsys Bagnoli-2.

4.1.1 Eletrodos

Esse dispositivo conta com eletrodos de barras paralelas de prata pura (99,9% de pureza) com uma distância fixa entre contatos de 10 milímetros. Ele é feito de policarbonato, com um formato para maximizar o contato com a pele e selado completamente, além de ser blindado internamente para rejeitar ruído [38].

A Figura 15 mostra as dimensões dos eletrodos , já a Figura 16 mostra o eletrodo em questão e seu cabo.

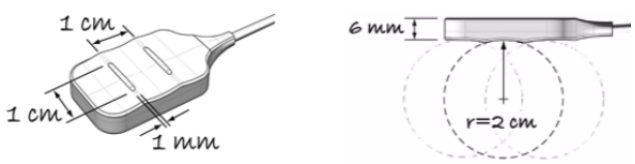


Figura 15 – Dimensões e geometria do eletrodo do Delsys Bagnoli-2. Adaptado de DELSYS [38].

Figura 16 – Eletrodo do Delsys Bagnoli-2.

4.2 NATIONAL INSTRUMENTS NI USB-6002

O Delsys Bagnoli-2 não digitaliza os sinais captados por si só, ele apenas os disponibiliza através de um cabo, para que eles sejam digitalizados por meio de um dispositivo de aquisição de dados (DAQ, de *Data Acquisition Device*) externo.

Nesse contexto, surge o NI USB-6002 da National Instruments (Figura 17), que é um DAQ com resolução de 16 *bits* em seu ADC, capaz de ler até 50.000 amostras por segundo e com entradas para sinais na faixa -10 V a 10 V [39]. DELSYS [38] recomenda o NI USB-6009, porém ambos pertencem à mesma família de dispositivos.

Figura 17 – NI USB-6002.

Para seu uso, basta se conectar via USB, que serve tanto para alimentação quanto para transmissão e controle, a um computador com sistema operacional Windows e utilizar

um dos *softwares* compatíveis. Neste caso, foi utilizado o NI Measurement & Automation Explorer (NI MAX) para coletar sinais, bem como para se ajustar os seus parâmetros. A Figura 18 mostra um exemplo de tela desse *software*.

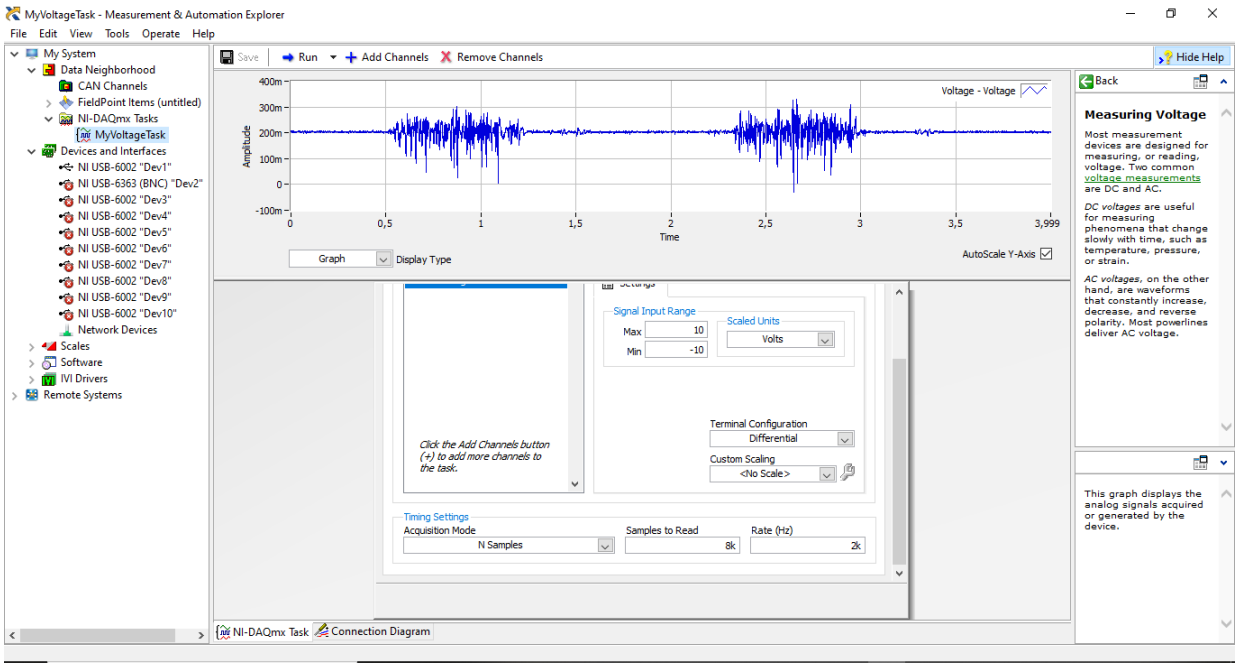


Figura 18 – Tela do NI MAX com o NI USB-6002 conectado em teste de coleta.

5 DISPOSITIVO CONSTRUÍDO

A partir do que foi especificado, foi construído um dispositivo compacto, leve, operado por bateria, com cabos curtos, de modo a se evitar interferências [16], e capaz de transmissão sem fio (Figura 19).

Esse dispositivo pode ser afixado ao corpo por meio de velcro (Figura 20) e se conecta aos eletrodos por meio de conectores do tipo jacaré parafusado a um conector na placa. O tipo de conector e sua forma de conexão foram escolhidos para dar facilidade em caso de uma eventual troca, removendo a necessidade de solda.

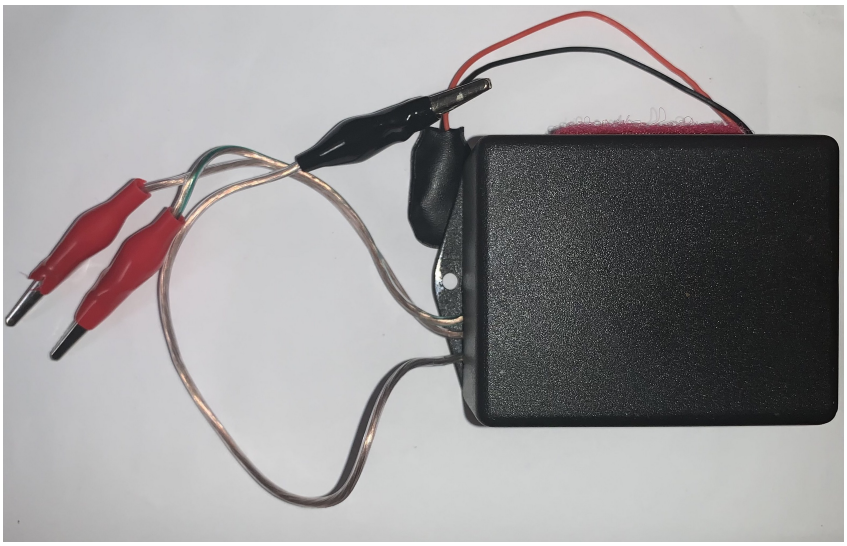


Figura 19 – Dispositivo construído.

O dispositivo foi chamado dev5 devido ao fato de ser a quinta iteração feita. Ele conta com um LED para indicar seu funcionamento (Figura 21). Ele conta com um ajuste de ganho de entrada (pré-amplificação) de aproximadamente 100, ajustado por meio de um potenciômetro e um ganho total de 1.000, também ajustado por meio de outro potenciômetro. O valor definido para o ganho de entrada tem por objetivo não amplificar em demasiado o sinal antes que o mesmo seja filtrado, reduzindo assim sinais indesejados. A Equação 13 mostra como se calcular o ganho total. Ambos potenciômetros podem ser vistos na Figura 22, que apresenta o dispositivo aberto.

$$G_{total} = G_{entrada} \cdot \frac{R_G + 1k\Omega}{1k\Omega} \tag{13}$$

O esquemático do dev5 pode ser visto no Apêndice A.1. Se desejado, há a possibilidade de se ajustar os ganhos com o auxílio de um multímetro na função de medir resistência e fazendo o ajuste nos devidos potenciômetros.

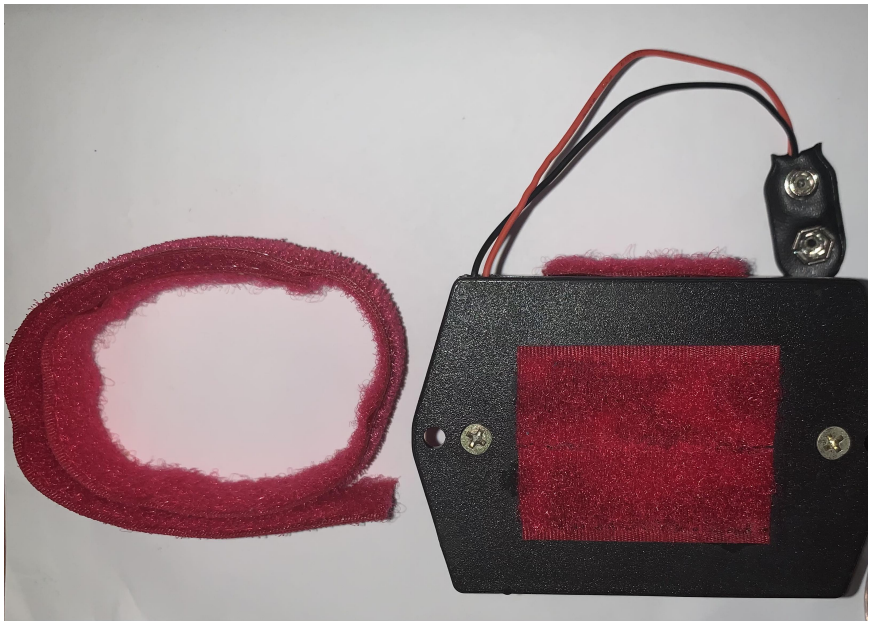


Figura 20 – Velcro utilizado para fixar o dispositivo ao corpo.

Figura 21 – LED sinalizador de funcionamento.

O ganho de entrada pode ser ajustado calculando-se a resistência necessária (R_G) através da Equação 12, medindo a resistência entre os pinos 1 e 8 do INA118 e fazendo o ajuste requerido no potenciômetro. Já para o ganho total, a resistência (R_G) pode ser calculada tendo-se o ganho de entrada ($G_{entrada}$) a partir da Equação 13 e o ajuste se dá após medida a resistência entre os pinos 8 e 9 do amplificador operacional TL074 ou equivalente (e.g. LM324) do circuito.

O dev5 também conta com eletrodos de superfície, que contam com seu próprio eletrólito, conforme a Figura 23 (eletrodos de Ag/AgCl).

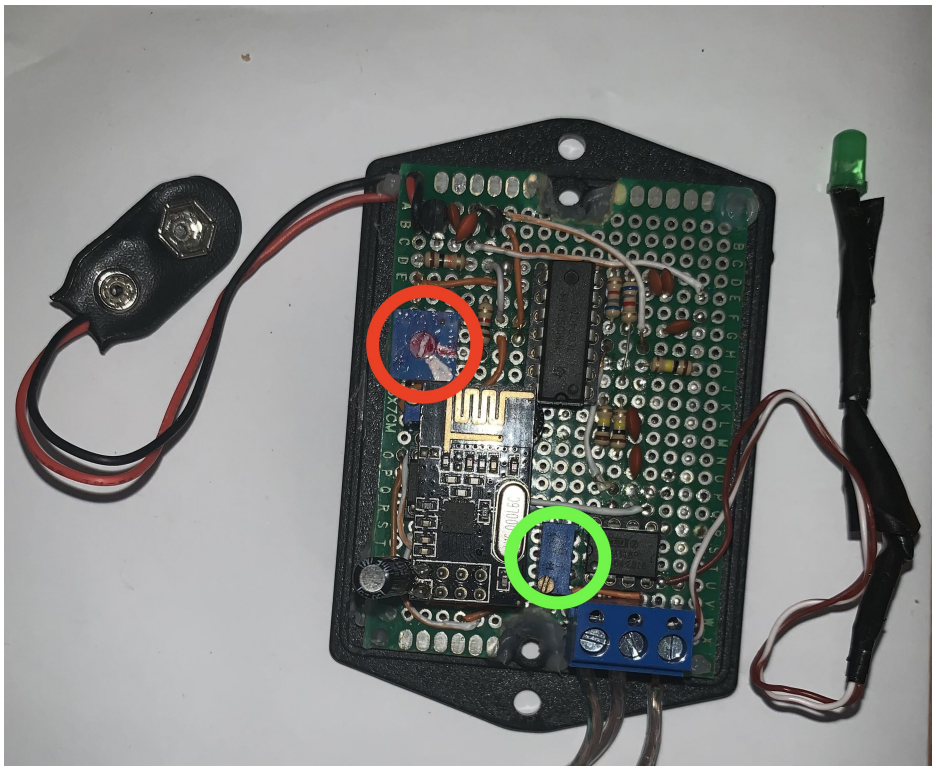


Figura 22 – Dispositivo aberto com os potenciômetros de ajuste de ganho de entrada (em verde) e total (em vermelho) visíveis.

Figura 23 – Eletrodos utilizados.

5.1 ESPECIFICAÇÕES

A Tabela 3 consolida as especificações para o dev5.

Tabela 3 – Especificações do dev5.

Característica	Valor
Ganho de entrada:	100
Ganho total:	1.000
Banda de operação:	22,5 Hz a 501,5 Hz
Ordem dos filtros:	2^a ordem (40 dB/década)
Resolução:	10 *bits*
Taxa de amostragem:	2kHz
Alimentação:	bateria de 9 V
CMRR mínimo aproximado*	116 dB
PSRR mínimo aproximado*	119 dB
Consumo medido:	17 mA

*Valores após o INA118.

5.1.1 CMRR e PSRR após INA118

As Figuras 24, 25 e 26 mostram, respectivamente, os gráficos para verificação aproximada dos valores de referência do CMRR, do PSRR para alimentação positiva e do PSRR para alimentação negativa.

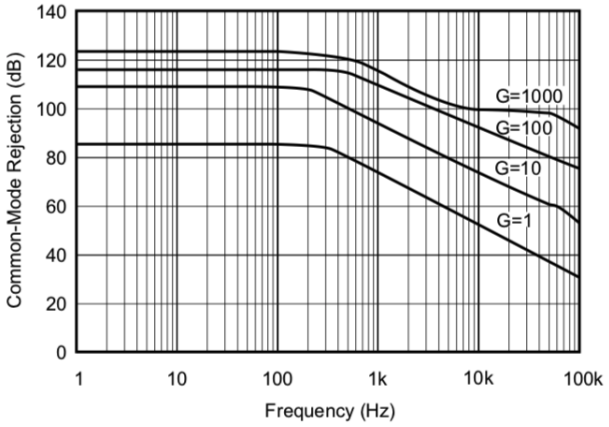


Figura 24 – CMRR para o INA118 [36].

Ao se considerar o ganho de entrada, já mencionado, de 100, a frequência máxima do sinal de 500 Hz [18] e a frequência de corte definida de 501,5 Hz (Tabela 3), pode-se estimar os valores mínimos de CMRR e PSRR.

Assim, o valor mínimo de CMRR foi estimado em 116 dB, o valor do PSRR para alimentação positiva em 119 dB e o PSRR para alimentação negativa em 127 dB. Uma vez que o sistema permite a mudança, há a possibilidade de se alterar esses valores com a mudança do ganho de entrada.

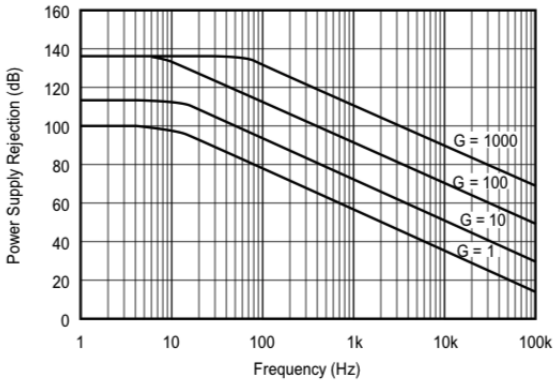


Figura 25 – PSRR para alimentação positiva para o INA118 [36].

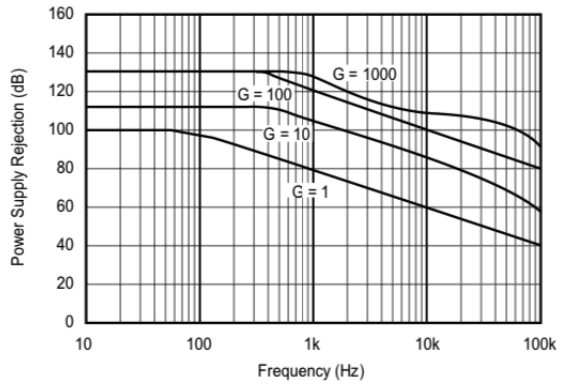


Figura 26 – PSRR para alimentação negativa para o INA118 [36].

5.1.2 Filtros utilizados

O dev5 conta com dois filtros, um passa-altas logo após a pré-amplificação (saída do INA118) e, após ele, um passa-baixas. Ambos são filtros ativos de segunda ordem com topologia Sallen-Key (Figura 27), o fato de serem de segunda ordem provoca uma atenuação de 40 dB por década no sinal, significando que a amplitude do sinal cai 100 vezes cada vez que a frequência muda por um fator de 10. Essa ordem de filtro é comum em dispositivos de coleta de sEMG [1].

Ambos os filtros tiveram seus valores de componentes calculados a partir de uma ferramenta japonesa *online* de projeto de filtros [40]. A topologia de filtros, os valores dos componentes e ordem foram escolhidos tendo-se em mente a quantidade de componentes necessários, resultados e a complexidade, uma vez que deseja obter um dispositivo de fácil construção e boa reprodutibilidade.

Além disso, frequências de corte foram delimitadas tendo-se por base a banda do sinal de EMG (entre 2 Hz e 500 Hz [19] ou entre 6 Hz a 500 Hz [17]), a banda do Delsys Bagnoli-2 (20 ± 5 Hz a 450 ± 50 Hz [38]), a faixa geral mencionada por [1] (entre 10-20 Hz e 400-450 Hz), facilidade de valores de componentes e para se evitar artefatos de movimento, os quais chegam até cerca de 20 Hz [1].

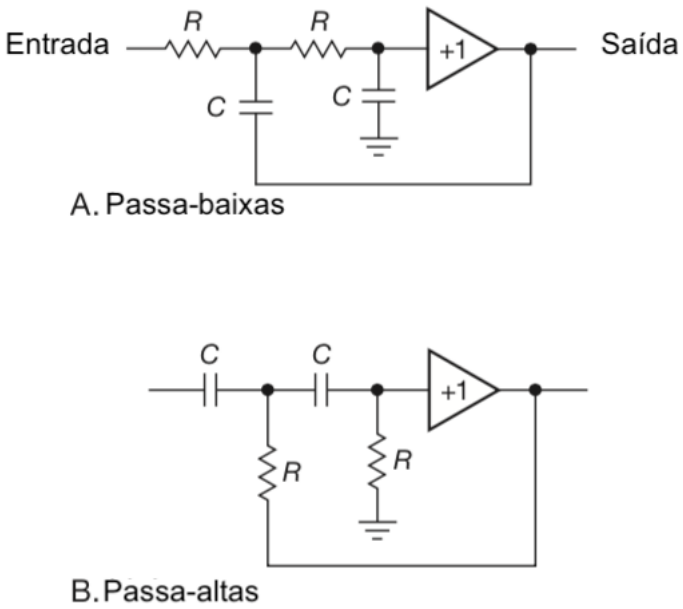


Figura 27 – Filtros ativos de 2^a ordem passa-baixas (A) e passa-altas (B) de topologia Sallen-Key, com amplificador operacional como *buffer*. Adaptado de Horowitz e Hill [20].

As Figuras 28 e 29 apresentam as respostas dos filtros calculadas com a ferramenta [40]. A banda de operação dos filtros pode ser vista na Tabela 3.

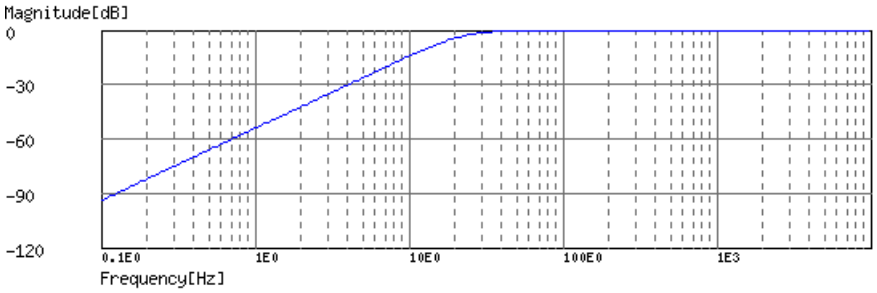


Figura 28 – Resposta do passa-altas, frequência de corte em 22,5 Hz.

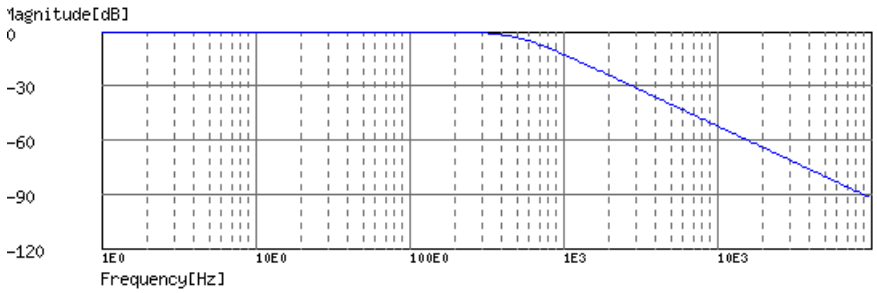


Figura 29 – Resposta do passa-baixas, frequência de corte em 501,5 Hz.

Os detalhamentos dos valores de cada filtro e suas respostas podem ser encon-

trados nos Anexos A.1 e A.2. Os valores dos componentes também podem ser vistos no esquemático (Apêndice A.1).

5.1.3 Consumo e alimentação

O dispositivo construído foi feito para ser alimentado a partir de uma bateria de 9 V comum. Isso também permite seu uso por meio de baterias de 9 V recarregáveis, o que reduz o lixo produzido após seu uso.

O tempo máximo de funcionamento do dispositivo pode ser estimado com base na capacidade da bateria e no consumo do dispositivo. Foi registrado, como já mostrado na Tabela 3, um consumo aproximado 17 mA pelo dispositivo.

Ao se considerar que uma bateria alcalina comum possui aproximadamente 600 mAh [41], pode-se estimar um tempo máximo de operação de 35 horas. Para o caso de baterias recarregáveis, considerando uma bateria recarregável com 450 mAh como a da Figura 30, obtém-se um tempo máximo estimado de 26 horas. As estimativas desprezam fatores externos, tais como grau de uso e/ou idade da bateria.

Figura 30 – Bateria recarregável utilizada.

5.1.4 ADC e taxa de amostragem

O dev5 foi configurado para trabalhar com taxa de amostragem de 2 kHz, que é mais que o suficiente ao se considerar que a banda do sinal do sinal de EMG vai até 500 Hz [19, 17]. Assim, esse valor satisfaz o critério de Nyquist [20] de pelo menos 1 kHz.

Devido ao uso do ATtiny85, o dev5 possui uma resolução de 10 *bits* [35]. Este fato poderia limitar seus usos, no entanto, o ATtiny85 pode ter sua tensão de referência ajustada [35], que resulta em uma sobrevida do mesmo e amplia sua gama de usos.

De modo analgo a Merletti e Parker [1] e considerando que 10 *bits* equivalem a 1024 diferentes níveis de tensão (sendo 1023 o nível máximo que se igualaria a 1,1 V) e um ganho total de 1.000, pode-se chegar a Tabela 4.

Tabela 4 – Relação entre as diferentes tensões máximas de leitura do ATtiny85, valores mínimos de leitura e valores mínimos de leitura, considerando o ganho total.

Tensão máxima de leitura (referência)	Leitura mínima	Leitura mínima, considerando ganho total de 1.000
1,1 V	1,1 mV	1,1 μV
2,56 V	2,5 mV	2,5 μV
3,3 V (alimentação)	3,2 mV	3,2 μV

Como pode ser visto na Tabela 4, a referência interna de 1,1 V resulta numa entrada sensível a sinais de pequena magnitude, chegando a ser capaz de ler sinais com 1,1 μV. Por este motivo, a tensão interna foi ajustada para 1,1 V.

Assim, observa-se ser possível captar sinais que se encontram dentro da faixa esperada para o sinal de EMG, considerado aqui entre 0 e 10 mV pico a pico [18], sem dificuldades.

5.2 TRANSMISSÃO, RECEPÇÃO E COMUNICAÇÃO COM O COMPUTADOR

A recepção dos sinais é feita por meio de um Arduino Nano conectado a um módulo transceptor NRF24L01+, como pode ser visto na Figura 31. A comunicação entre emissor (dev5) e receptor é feita por meio de protocolo proprietário dos transceptores NRF24L01+. Em testes informais foi observado um *range* de 10 a 20 metros entre o emissor e o receptor com transmissões ininterruptas.

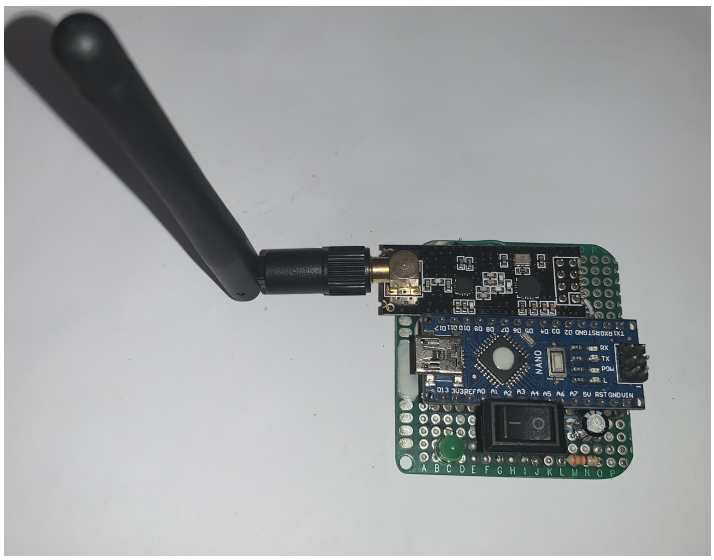


Figura 31 – Receptor com antena de longo alcance.

Internamente, para se manter uma taxa de transmissão constante entre o receptor e o computador, os valores recebidos de 10 *bits* do transmissor do ADC são divididos em

duas partes. Uma parte contém o resultado inteiro da divisão por 256 e a outra o resto dessa divisão.

Para a transmissão entre o receptor e o computador foi feito o uso do protocolo UART (*Universal Asynchronous Receiver/Transmitter*, serial) através de um cabo USB, uma vez que se trata de um protocolo de fácil uso e já suportado nativamente pelo Arduino. Dentro do protocolo são enviados pacotes de 10 *bits*, tendo sido definidos 8 *bits* para a mensagem e 2 *bits* para controle [20].

Essa escolha permite obter uma taxa constante de transmissão, uma vez que ela garante que cada parte tenha exatamente 8 *bits*. Por causa disso, cada valor é transmitido constantemente com o uso de dois pacotes pelo protocolo serial.

Isso permitiu que fosse definida uma taxa de transmissão (*baud rate*) adequada para o tipo de sinal transmitido entre o receptor e o computador. Considerando-se uma taxa de amostragem de 2 KHz e que cada valor ocupa constantemente 16 *bits* (10 *bits* do ADC, separados em duas partes de 8 *bits*), tem-se uma taxa de 32 Kbps para o sinal. Ao se considerar o formato do protocolo de transmissão, há a necessidade de um (*baud rate*) de pelo menos 40 Kbps.

Por este motivo, foi escolhido o *baud rate* de 115200 (115,2 Kbps), que seria suficiente para, inclusive, se adicionar outro canal.

5.3 *SOFTWARE*

Os programas desenvolvidos para a transmissão e recepção dos sinais se encontram nos Apêndices B.2 e B.3, respectivamente.

Foram também elaborados programas para permitir a visualização em tempo real dos sinais (Apêndice C.1), coleta apenas (Apêndice C.2), visualização apenas (Apêndice C.3) e visualização do espectro do sinal em tempo real (Apêndice C.4).

A Figura 32 exemplifica a tela de visualização dos sinais em tempo real (*software* do Apêndice C.1).

Todos esses códigos dependem do módulo de ferramentas (Apêndice C.5) e do módulo de configuração (Apêndice C.6). O módulo de configuração permite o ajuste dos parâmetros para visualização, indo desde o tamanho da janela em segundos até filtros digitais.

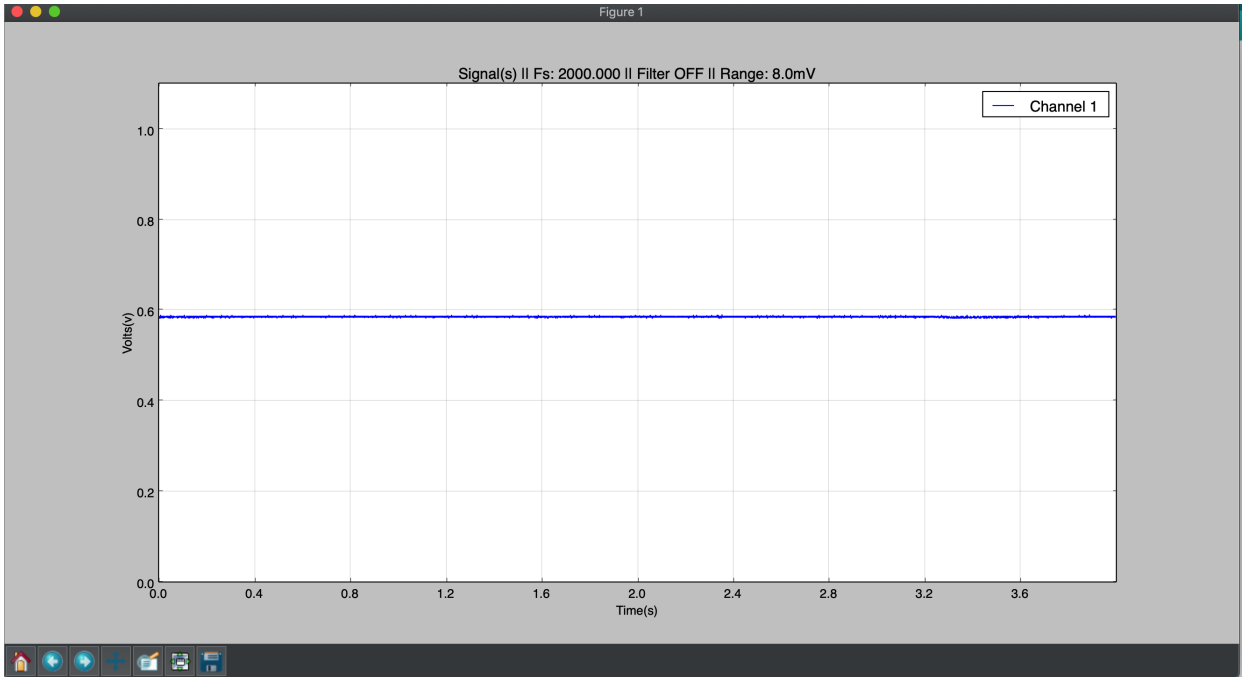


Figura 32 – Tela para exibição de sinais em tempo real.

Os códigos foram feitos de modo a facilitar a sua leitura e compreensão, contendo comentários detalhados do que foi feito. Isso teve por objetivo dar flexibilidade a quem desejasse os modificar. Assim, eles atuam mais como uma base do que pode ser feito do que algo fechado e imutável. A linguagem escolhida, Python, permite essa robustez e flexibilidade, sem que seja perdida a legibilidade do código.

Para a parte embarcada, conta-se com os mesmos princípios de sua contraparte em Python, permitindo, inclusive, que se ajuste a frequência de amostragem. Também foi incorporada a possibilidade de uso mais de um canal, porém, seu uso funcional carece de mais ajustes.

6 METODOLOGIA

6.1 DELIMITAÇÃO DO TRABALHO

O presente trabalho visa comparar um dispositivo construído (dev5) com o estado da arte (Delsys Bagnoli-2). A comparação será feita com sinais coletados na modalidade de ensaio de bancada, utilizando-se somente de sinais coletados do pesquisador e removendo a necessidade de sua aprovação por um comitê de ética.

Ademais, não é visado por este trabalho substituir o estado da arte, mas sim se obter um dispositivo que seja bom o suficiente para facilitar o uso de sEMG pela comunidade acadêmica.

6.2 PREPARAÇÃO PARA A COLETA

A coleta foi precedida dos procedimentos necessários à minimização de interferências, tais como: tricotomia da área, abrasão vigorosa da pele e limpeza com álcool. Optou-se, também, por desligar todos os equipamentos elétricos presentes das tomadas, de modo a se evitar a interferência da rede elétrica nos sinais captados [16, 17].

6.3 LOCAL E POSICIONAMENTO DOS ELETRODOS

O bíceps braquial (Figura 33) foi escolhido para a realização da coleta dos sinais. Esse músculo foi escolhido pela sua localização e relativa facilidade de obtenção de sinais.

As coletas foram realizada de forma separada, porém mantendo-se o mesmo posicionamento dos eletrodos. Uma caneta foi utilizada para marcar e manter o posicionamento entre coletas.

É importante ressaltar que as barras do eletrodo do Delsys Bagnoli-2 e os eletrodos de Ag/AgCl do dev5 foram posicionados de forma perpendicular ao músculo [38], conforme a Figura 34, e logo acima de seu ventre. As referências foram posicionadas no cotovelo, a crista óssea mais próxima.

Figura 33 – Posição do bíceps braquial. Adaptado de THOUGHT TECNLOLOGY LTD. [42].

Figura 34 – Posicionamento perpendicular das barras do eletrodo do Delsys Bagnoli-2 ao sentido das fibras musculares (reta no mesmo sentido). Adaptado de DELSYS [38].

6.4 CONFIGURAÇÃO DOS DISPOSITIVOS

O sinal do Delsys Bagnoli-2 foi adquirido com o auxílio do dispositivo NI USB-6002 e do *software* NI MAX, ambos da National Instruments. Já o dev5 foi adquirido por *software*

próprio, desenvolvido durante a execução deste trabalho.

A taxa de amostragem do *software* e ganho do Delsys Bagnoli-2 foram ajustados para os mesmos do dev5, ou seja, uma taxa de amostragem de 2 kHz e um ganho de 1.000.

Ambos sinais foram captados sem a utilização de filtros digitais e/ou qualquer alteração por meio de *software*.

6.5 PROTOCOLOS DE COLETA

6.5.1 Coleta em repouso

Durante essa coleta, ambos sinais foram adquiridos sem que fosse executado movimento algum, em repouso e com o braço em sua posição natural durante um período de 4 segundos. Os sinais aqui obtidos foram considerados como ruído para os fins de sua utilização, mesmo sabendo-se da existência do tônus muscular.

6.5.2 Coleta durante contrações isométricas

Os sinais foram coletados separadamente durante um intervalo fixo de 10 segundos, utilizando-se uma referência de 5 kg a ser levantada.

Cada uma das coletas se iniciava com o braço em repouso e a referência em um apoio, logo em seguida a referência era levantada, flexionando-se o braço. Ela era mantida erguida por pelo menos 5 segundos, após isso o braço retornava ao repouso e a referência ao seu apoio.

6.6 FERRAMENTAS UTILIZADAS

A preparação e análise dos sinais foi feita em Python e se utilizou das ferramentas Jupyter Notebook [34] e das bibliotecas Numpy [29] e Matplolib [31].

6.7 PREPARAÇÃO DOS SINAIS COLETADOS

O *software* NI MAX conta com a funcionalidade de exportar seus sinais coletados em formato CSV (*Comma-separated Values* - Valores Separados por Vírgula), o que facilita seu uso posterior em outras ferramentas.

Entretanto, ele também faz a adição de certos caracteres e certas mudanças nos números que dificultam a sua importação posterior e compreensão por linguagens de programação. Assim, foi feito um código em Python que automatiza o processo de limpeza dos dados (Apêndice C.7) antes de sua utilização. Por utilizar *software* próprio, os dados exportados pelo dev5 já se encontravam no formato esperado.

Após isso, os sinais tiveram seus níveis DC removidos de modo a ficarem ao entorno de zero, já que ambos dispositivos desprezam frequências abaixo de 20 Hz.

6.8 ANÁLISE E COMPARAÇÃO

Os transitórios do inicio e fim das contrações isométricas foram removidos de modo que se obtivesse um sinal em regime estacionário para as análises subsequentes.

Os sinais dos dispositivos foram comparados em termos de valor médio retificado, da área sob a envoltória linear, dos valores RMS, da MNF e da MDF.

A envoltória linear vem da retificação do sinal de EMG (seja por obtenção do valor absoluto ou pelo quadrados do sinal) e posterior filtragem por um passa-baixas [1]. Um filtro Butterworth passa-baixas de quarta ordem e frequência de corte em 20 Hz foi utilizado, de modo análogo a George, Sivanandan e Mohandas [43].

Os dispositivos também foram comparados em termos de suas especificações técnicas e dos valores SNR.

As análises de sinais foram exportadas com um *Notebook* do Jupyter [34] e se encontram no Apêndice D.

7 RESULTADOS E DISCUSSÃO

As coletas foram realizadas conforme descrito na metodologia. As Figuras 35 e 36 mostram respectivamente os sinais adquiridos durante contrações isométricas e em repouso. Os sinais após a remoção dos transitórios podem ser vistos na Figura 37.

É importante destacar que apenas o sinal estacionário foi utilizado para as análises dos sinais coletados em contração isométrica.

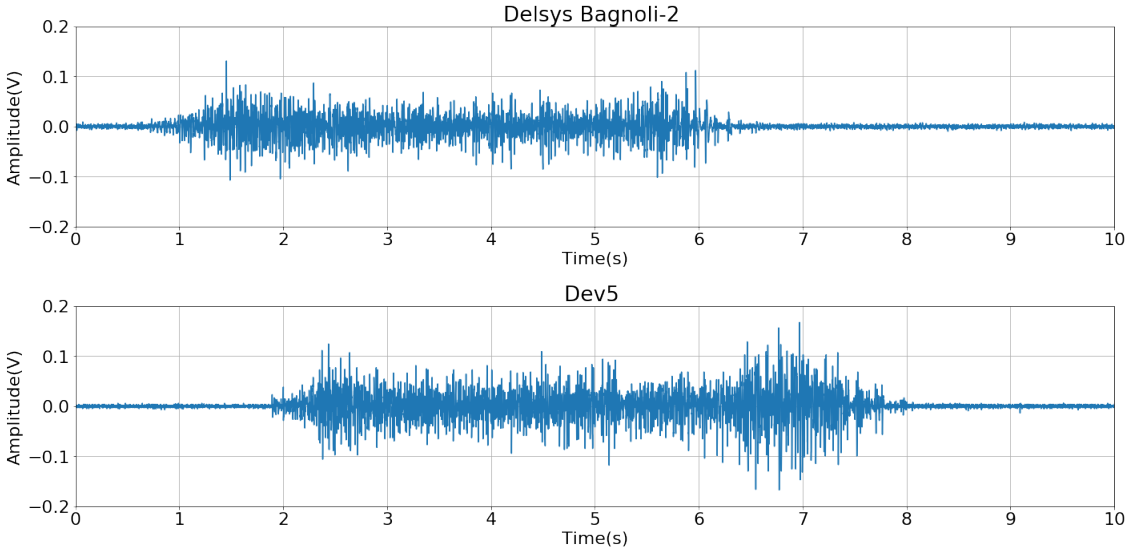


Figura 35 – Sinais adquiridos durante contrações isométricas. Delsys Bagnoli-2 na parte superior e dev5 na parte inferior.

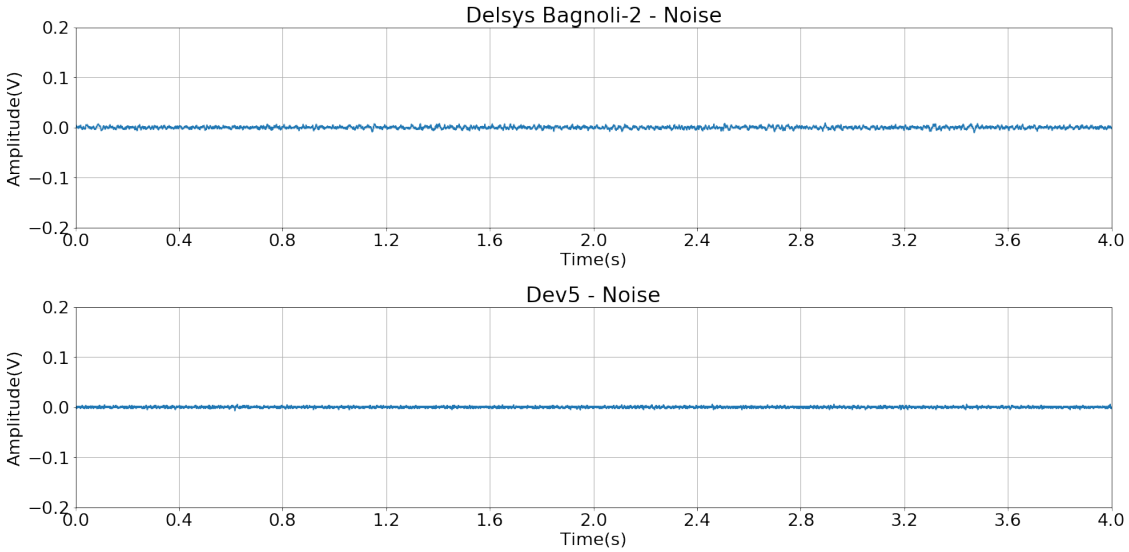


Figura 36 – Sinais adquiridos em repouso. Delsys Bagnoli-2 na parte superior e dev5 na parte inferior.

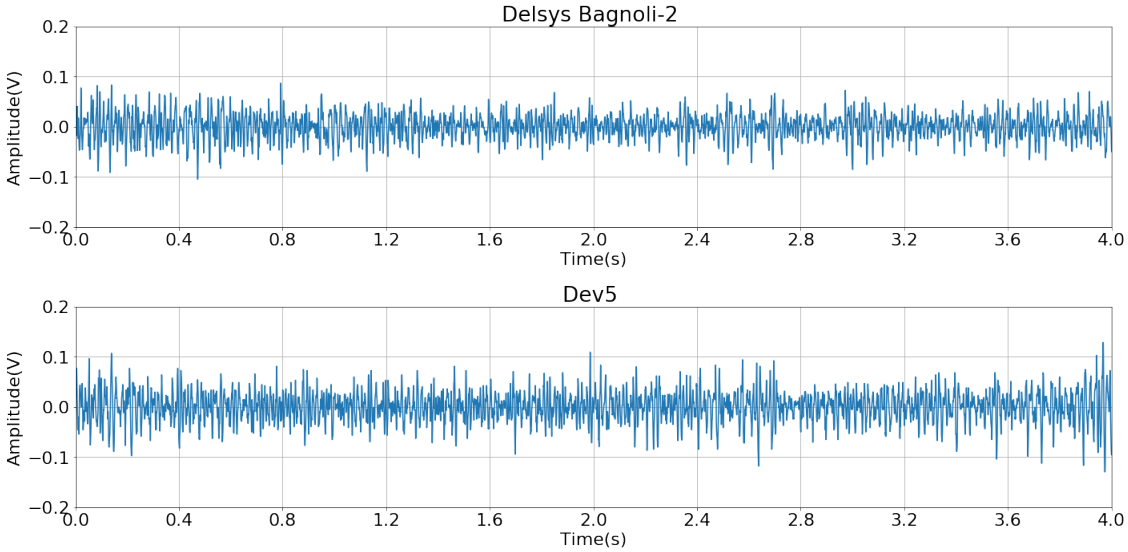


Figura 37 – Sinais adquiridos durante contrações isométricas, após remoção de transitórios. Delsys Bagnoli-2 na parte superior e dev5 na parte inferior.

7.1 VALOR MÉDIO RETIFICADO

Os valores médios para os sinais de EMG retificados coletados em contração isométrica podem ser vistos na Tabela 5. Os sinais retificados são observáveis na Figura 38.

Tabela 5 – Valor médio retificado.

Dispositivo	Valor médio
Delsys Bagnoli-2	19,29 mV
dev5	23,46 mV

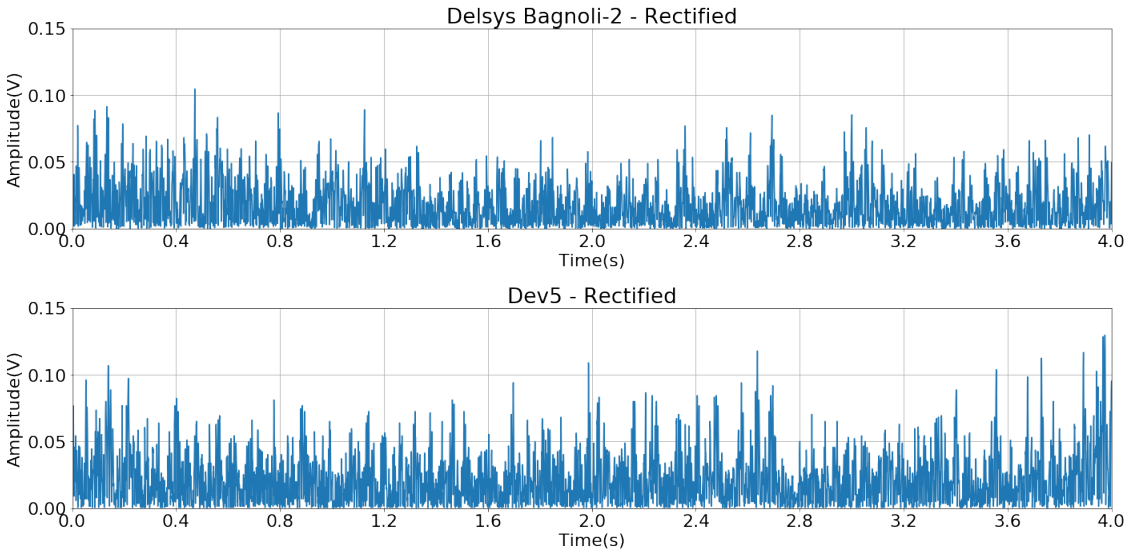


Figura 38 – Sinais retificados. Delsys Bagnoli-2 na parte superior e dev5 na parte inferior.

7.2 ÁREA SOB A ENVOLTÓRIA LINEAR

A Tabela 6 mostra os valores da área sob a envoltória linear para os sinais coletados em contração isométrica. A Figura 39 mostra a envoltória em conjunto com o sinal retificado.

Tabela 6 – Área sob a envoltória linear.

Dispositivo	Área
Delsys Bagnoli-2	76,63 mV.s
dev5	92,95 mV.s

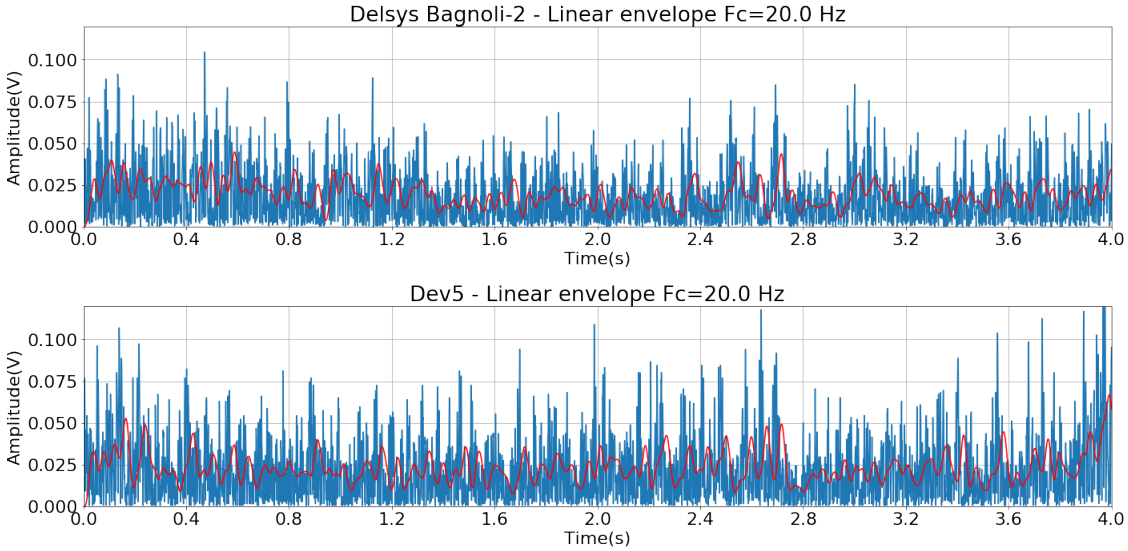


Figura 39 – Envoltória linear e sinal retificado. Delsys Bagnoli-2 na parte superior e dev5 na parte inferior.

7.3 VALORES DE RMS

A Tabela 7 traz os valores de RMS calculados para os sinais obtidos de ambos dispositivos durante em contração isométrica

Tabela 7 – Valores calculados de RMS para as coletas.

Dispositivo	Coleta durante contração isométrica	Coleta em repouso
Delsys Bagnoli-2	24,84 mV(RMS)	2,48 mV(RMS)
dev5	30,24 mV(RMS)	1,55 mV(RMS)

7.4 MNF E MDF

A Tabela 8 mostra os valores estimados para MNF e MDF observados em cada um dos dispositivos para os sinais adquiridos em contração isométrica. O cálculo do espectro de

potência do sinal foi feito com o uso do método de Welch. A Figura 40 mostra os sinais no domínio da frequência.

Tabela 8 – MMF e MDF

Dispositivo	MNF	MDF
Delsys Bagnoli-2	82,06 Hz	70,31 Hz
dev5	84,04 Hz	70,31 Hz

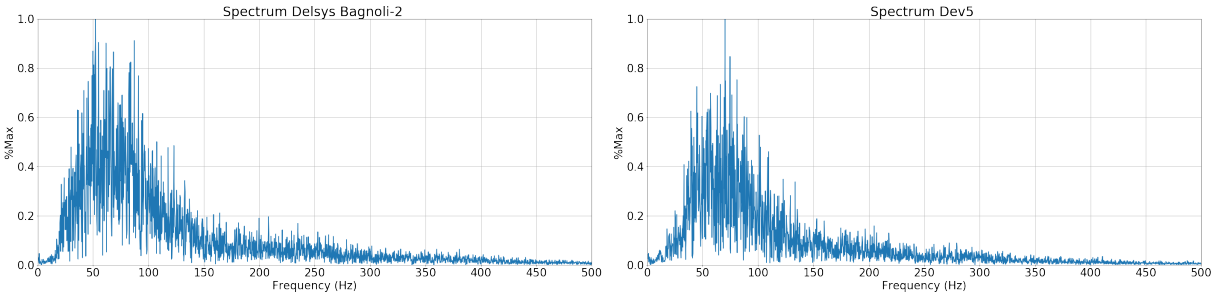


Figura 40 – Espectro para os sinais adquiridos durante contrações isométricas. Delsys Bagnoli-2 à esquerda e dev5 à direita.

7.5 VALORES DE SNR

Os valores de SNR foram calculados utilizando-se a Equação 8, para cálculo dos valores de tensão RMS, e a Equação 9, para cálculo efetivo dos valores de SNR de cada coleta.

O sinal coletado em repouso foi considerado como ruído para fins de cálculo. Os valores calculados podem ser observados na Tabela 9.

Tabela 9 – Valores calculados de SNR para as coletas.

Dispositivo	SNR
Delsys Bagnoli-2	20,01 dB
dev5	25, 81 dB

7.6 COMPARAÇÃO ENTRE OS DISPOSITIVOS

Os parâmetros específicos de cada dispositivo já foram discutidos nos capítulos 4 e 5, bem como nas Tabelas 2 e 3.

É observado que ambos contam similaridades, tais como o uso de baterias de 9 V, a possibilidade de se trabalhar com ganhos de 1.000, a taxa de amostragem ajustada em 2 kHz e mesmo suas bandas de operação são semelhantes, entre 22,5 - 501,5 Hz (dev5) e 20 ± 5 - 450 ± 5 Hz (Delsys Bagnoli-2).

Por outro lado, é inegável que o Delsys Bagnoli-2 é um dispositivo muito mais versátil e prático, devido a sua facilidade de uso, filtros com ordem superior e ajuste fácil dos ganhos entre 3 possibilidades, além de já ser consolidado no mercado.

O dev5 permite os ajustes dos ganhos e, via alteração em código, da sua taxa de amostragem, fatos que reduzem sua praticidade de uso fora dos parâmetros estabelecidos inicialmente. Porém, trata-se de um dispositivo sem fios, ao contrário do Delsys Bagnoli-2, que remove a necessidade de cabos atrelados ao paciente durante coletas, bem como diminui sua suscetibilidade a artefatos de movimento.

O Delsys Bagnoli-2 possui a necessidade de ser utilizado em conjunto com um DAQ, visto que não possui ADC em sua construção. Isso lhe permite uma maior flexibilidade quanto a resolução e a taxa de amostragem dos seus sinais. Entretanto, também se torna mais um equipamento a ser adquirido em conjunto para que ele possa operar, acarretando em mais custos e o deixando menos acessível. Enquanto o dev5 encapsula tudo isso em sua construção.

É impossível não se observar a diferença de resolução entre os dispositivos, visto que o conjunto Delsys Bagnoli-2 e NI USB-6002 consegue uma resolução de 16 *bits* numa faixa ±10 V, resultando em um valor mínimo de leitura (considerando-se o ganho em 1.000) de 0,30 μV. O dev5 consegue 1,1 μV (Tabela 3). Essa diferença de resolução pode ser observada na prática ao se ampliar um sinal de baixa amplitude (Figura 41), como o sinal de ruído, por exemplo.

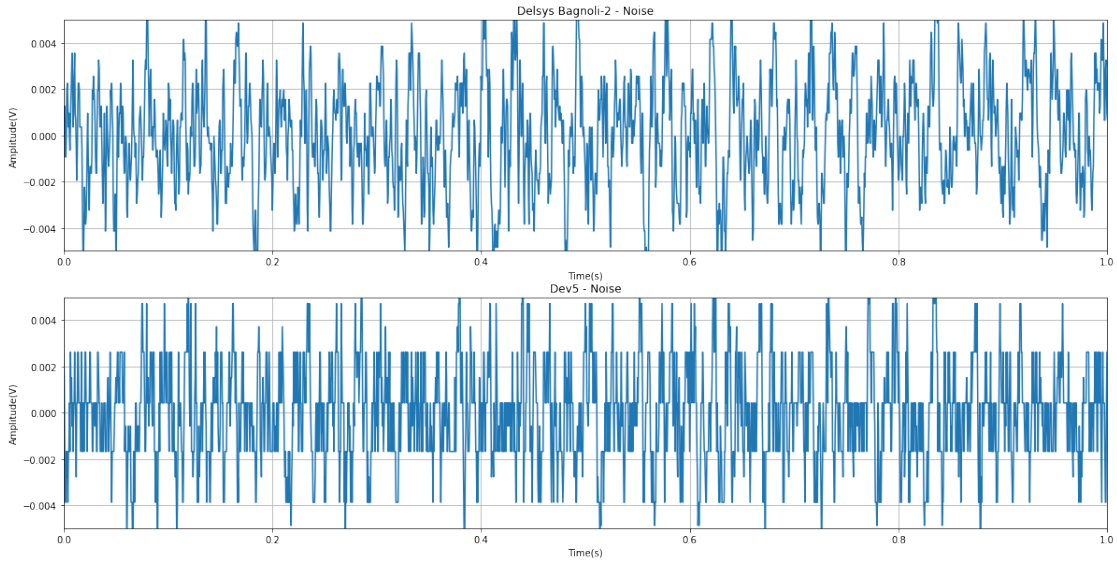


Figura 41 – Diferenças de resolução observadas entre os dispositivos. Delsys Bagnoli-2 na parte superior e dev5 na parte inferior. O sinal captado pelo Delsys Bagnoli-2 é claramente mais suave, enquanto os níveis de quantização já podem ser observados para o dev5.

Por outro lado, deve-se lembrar que se objetiva por coletar um sinal que se considera

estar até 10 mV [18], sendo o próprio valor mínimo do dev5, mesmo sendo inferior, cerca 9.000 vezes menor que essa referência.

Pode-se também observar que para as coletas o dev possui melhor SNR, maior valor RMS para o sinal, menor valor de RMS em sinal em repouso (aqui considerado ruído), maior valor médio retificado e maior área da envoltória linear. Além disso, ele também tem valores de MNF e MNF comparáveis aos do Delsys Bagnoli-2. Todos esses, fatos notáveis para dev5.

7.7 DISPONIBILIZAÇÃO

Os códigos e esquemáticos desenvolvidos durante este trabalho se encontram nos Apêndices A, B, C e D e foram disponibilizados em:

https://github.com/ithallojunior/sEMG-device.

8 CONCLUSÃO

Foi desenvolvido um dispositivo para a coleta de sinais sEMG sem fios que possui boas características.

Esse dispositivo possui filtros de 2^a ordem ativos, que atenuam componentes espectrais indesejadas a uma taxa de 100 vezes para cada vez cada vez que a frequência se altera num fator de 10.

O dev5 permite uma taxa de amostragem confiável, uma vez que seu oscilado foi ajustado em 1 % por meio do OSCCAL. Além disso, traz facilidade e praticidade para seu uso ao contar com um *case* protetor e uso de baterias de 9 V, em adição a permitir uma mais fácil substituição dos cabos, não soldados a placa. Ele também permite a fixação do dispositivo ao corpo por meio do velcro já embutido.

Quanto a sua comparação ao Delsys Bagnoli-2, percebe-se que se trata de um dispositivo ainda longe da qualidade técnica e de construção do mesmo, porém que é capaz de cumprir o seu propósito de realizar coletas de sinais de forma confiável e de funcionar dentro das suas limitações. Sendo assim, pode-se dizer que é um dispositivo que consegue funcionar e ser acessível, mesmo diante do trunfo de engenharia que é o Delsys Bagnoli-2.

O dispositivo, no entanto, ainda falha quanto a sua facilidade de uso. Ele ainda exige um certo nível de conhecimento em programação para configuração de seu uso, falta-lhe uma disponibilidade efetiva de mais canais para uso e seu ajuste de ganho é possível, porém trabalhoso. Apesar de tudo, uma vez que seu projeto foi disponibilizado de forma pública, abre-se a possibilidade para que outros pesquisadores e pessoas interessados também contribuam com seu avanço e melhorias.

9 TRABALHOS FUTUROS E PUBLICAÇÕES

9.1 TRABALHOS FUTUROS

Com o desenvolvimento deste trabalho se faz necessário prosseguir no aprimoramento de seus componentes e de seu *software*, de modo a se ter algo que, de fato, seja útil à comunidade acadêmica. Assim, os seguintes pontos podem ser elencados:

- Busca e adoção de outro amplificador operacional de instrumentação, que melhore as características do dispositivo;
- Substituição do ATtiny85 por outro microcontrolador com pelo menos 12 *bits* de resolução;
- Adoção de baterias de íons de lítio internas, que podem facilitar a recarga e adoção do dispositivo, porém acarretando em aumento de custos;
- Estudo de utilização de um transmissor Bluetooth, visando eliminar a necessidade de uso de um módulo de recepção;
- Melhoria na interface homem-máquina do sistema, visando facilitar seu uso.
- Migração da versão do código de Python 2 para Python 3, visando longevidade de suporte e compatibilidade do código;
- Melhoria no suporte a mais de um canal simultaneamente;
- Adicionar um indicador de nível de bateria;
- Facilitar o ajuste do ganho do dispositivo.

9.2 PUBLICAÇÕES

A elaboração deste trabalho resultou nas seguintes publicações até o momento:

GUIMARÃES, I. J. A.; LOPES, R. M. ; SILVA, J. L. F. J.; SOUSA, B. S.; MARAES, V. R. F. S.; BRASIL, L. M. **Predicting Knee Angles from Video: An Initial Experiment with Machine Learning**. In: Costa-Felix R.; Machado J.; Alvarenga A.. (eds). XXVI Brazilian Congress on Biomedical Engineering. IFMBE Proceedings. 1ed.: Springer, Singapore, 2019, v. 70/2.

GUIMARÃES, I. J. A.; LIMA, R. A. ; MARÃES, V. R. F. S. ; BRASIL, L. M. **HUMAN KNEE SIMULATION USING MULTILAYER PERCEPTRON ARTIFICIAL NEURAL NETWORK**. In: Christiane Trevisan Slivinski. (Org.). Impactos das Tecnologias nas Ciências Biológicas e da Saúde. 1ed.Ponta Grossa (PR): Atena, 2019, v. 2, p. 194-201.cl

As publicações encontram-se disponibilizadas, respectivamente, nos Apêndices E.1 e E.2.

REFERÊNCIAS BIBLIOGRÁFICAS

1 MERLETTI, R.; PARKER, P. A. *Electromyography: physiology, engineering, and non-invasive applications*. [S.l.]: John Wiley & Sons, 2004. v. 11. 16, 18, 19, 26, 41, 43, 50

2 MEZZARANE, R. A. et al. Experimental and simulated emg responses in the study of the human spinal cord. In: *Electrodiagnosis in new frontiers of clinical research*. [S.l.]: IntechOpen, 2013. 16

3 NAKAJIMA, T. et al. Reassessment of non-monosynaptic excitation from the motor cortex to motoneurons in single motor units of the human biceps brachii. *Frontiers in human neuroscience*, Frontiers, v. 11, p. 19, 2017. 16

4 GAZENDAM, M. G.; HOF, A. L. Averaged emg profiles in jogging and running at different speeds. *Gait & posture*, Elsevier, v. 25, n. 4, p. 604–614, 2007. 16

5 GERVASIO, S. et al. The effect of crossed reflex responses on dynamic stability during locomotion. *Journal of neurophysiology*, American Physiological Society Bethesda, MD, v. 114, n. 2, p. 1034–1040, 2015. 16

6 CRONIN, N. J. et al. Mechanical and neural stretch responses of the human soleus muscle at different walking speeds. *The Journal of physiology*, Wiley Online Library, v. 587, n. 13, p. 3375–3382, 2009. 16

7 SARTORIO, G. et al. Análise tempo-frequência do emg de diferentes porções do músculo tríceps braquial nas fases excêntricas e concêntricas dos exercícios testa e pulley. In: *Anais do XXVI Congresso Brasileiro de Engenharia Biomédica*. [s.n.], 2018. Disponível em: <http://media.cbeb.org.br/media/uploads/s/Analise_tempo-frequencia_do_EMG_de_diferentes_porcoes_do_musculo_triceps_braquial_nas_fases_excentricas_e_concentricas_nos_exercicios_testa_e_pulley.pdf>. Acesso em: 20 outubro 2019. 16

8 de Carvalho, M.; SWASH, M. Motor unit recruitment in myopathy: The myopathic emg reconsidered. *Journal of Electromyography and Kinesiology*, Elsevier, v. 45, p. 41–45, 2019. 16

9 PIERROT-DESEILLIGNY, E.; BURKE, D. *The circuitry of the human spinal cord: its role in motor control and movement disorders*. [S.l.]: Cambridge University Press, 2005. 16

10 ENOKA, R. M. *Neuromechanics of Human Movement*. [S.l.]: Human kinetics, 2008. 16

11 CHANG, K.-M.; LIU, S.-H.; WU, X.-H. A wireless semg recording system and its application to muscle fatigue detection. *Sensors*, Molecular Diversity Preservation International, v. 12, n. 1, p. 489–499, 2012. 17

12 KOBAYASHI, H. Emg/ecg acquisition system with online adjustable parameters using zigbee wireless technology. *Electronics and Communications in Japan*, Wiley Online Library, v. 96, n. 5, p. 1–10, 2013. 17

13 BIAGETTI, G. et al. Wireless surface electromyograph and electrocardiograph system on 802.15. 4. *IEEE Transactions on Consumer Electronics*, IEEE, v. 62, n. 3, p. 258–266, 2016. 17

14 GUIMARÃES, I. J. A. *Desenvolvimento tecnológico de um dispositivo de coleta de sinais sEMG aplicado ao diagnóstico da doença de Parkinson*. Monografia (Engenharia eletrônica) — Universidade de Brasília, Gama, 2017. 17

15 FARINA, D.; HOLOBAR, A. Characterization of human motor units from surface emg decomposition. *Proceedings of the IEEE*, IEEE, v. 104, n. 2, p. 353–373, 2016. 18

16 CRISWELL, E. *Cram's introduction to surface electromyography*. [S.l.]: Jones & Bartlett Publishers, 2010. 11, 18, 19, 20, 37, 47

17 KONRAD, P. *The ABC of EMG*. [S.l.]: Noraxon INC., 2005. 18, 19, 41, 43, 47

18 De Luca, C. J. Surface electromyography: Detection and recording. *DelSys Incorporated*, v. 10, n. 2, p. 1–10, 2002. Disponível em: <https://www.delsys.com/downloads/TUTORIAL/semg-detection-and-recording.pdf>. Acesso em: 20 outubro 2019. 19, 40, 44, 56

19 PRUTCHI, D.; NORRIS, M. *Design and Development of Medical Electronic Instrumentation: A Practical Perspective of the Design, Construction, and Test of Medical Devices*. [S.l.]: Jhon Wiley & Sons, 2005. 19, 30, 41, 43

20 HOROWITZ, P.; HILL, W. *The Art of Electronics*. 3. ed. [S.l.]: Cambridge University Press, 2016. 12, 19, 20, 21, 23, 24, 25, 30, 42, 43, 45

21 ALEXANDER, C. K.; SADIKU, M. N. O. *FUNDAMENTOS DE CIRCUITOS ELEÉTRICOS*. 5. ed. [S.l.]: The McGraw-Hill Companies, Inc, 2013. 12, 21, 22, 23, 24, 25

22 GREEN, P. S. T.; WELLS, C. (Ed.). *Analog Engineer's Circuit Cookbook: Amplifiers*. 2. ed. [S.l.]: Texas Instruments, 2019. 12, 22

23 III, J. O. S. *INTRODUCTION TO DIGITAL FILTERS WITH AUDIO APPLICATIONS*. Stanford University, 2007. Disponível em: <https://ccrma.stanford.edu/~jos/filters/>. Acesso em: 11 fevereiro 2021. 24

24 JONES, A. *Probability, Statistics and Other Frightening Stuff*. Taylor & Francis, 2018. (Working Guides to Estimating & Forecasting). ISBN 9781351661386. Disponível em: <https://books.google.com.br/books?id=OvtsDwAAQBAJ>. 25

25 OPPENHEIM, R. W. S. A. V.; BUCK, J. R. *Discrete-Time Signal Processing*. 2. ed. [S.l.]: Prentice-Hall Inc., 1998. ISBN 0137549202. 25

26 THONGPANJA, S. et al. Mean and median frequency of emg signal to determine muscle force based on time-dependent power spectrum. *Elektronika ir Elektrotechnika*, v. 19, n. 3, p. 51–56, Mar. 2013. Disponível em: <https://eejournal.ktu.lt/index.php/elt/article/view/3697>. 26

27 ARDUINO. *What is Arduino?* 2021. Disponível em: <https://www.arduino.cc/en/Guide/Introduction#>. Acesso em: 15 janeiro 2021. 27, 28

28 PYHTON SOFTWARE FOUNDATION. *BeginnersGuide Overview*. 2020. Disponível em: <https://wiki.python.org/moin/BeginnersGuide/Overview>. Acesso em: 17 dezembro 2020. 28

29 HARRIS, C. R. et al. Array programming with NumPy. *Nature*, Springer Science and Business Media LLC, v. 585, n. 7825, p. 357–362, set. 2020. Disponível em: <https://doi.org/10.1038/s41586-020-2649-2>. 28, 49

30 VIRTANEN, P. et al. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, v. 17, p. 261–272, 2020. 29

31 Hunter, J. D. Matplotlib: A 2d graphics environment. *Computing in Science Engineering*, v. 9, n. 3, p. 90–95, 2007. 29, 49

32 LIECHTI, C. *pySerial*. 2021. Disponível em: <https://github.com/pyserial/pyserial>. Acesso em: 19 janeiro 2021. 29

33 PÉREZ, F.; GRANGER, B. E. IPython: a system for interactive scientific computing. *Computing in Science and Engineering*, IEEE Computer Society, v. 9, n. 3, p. 21–29, maio 2007. ISSN 1521-9615. Disponível em: <https://ipython.org>. 29

34 KLUYVER, T. et al. Jupyter notebooks – a publishing format for reproducible computational workflows. In: LOIZIDES, F.; SCHMIDT, B. (Ed.). *Positioning and Power in Academic Publishing: Players, Agents and Agendas*. [S.l.], 2016. p. 87 – 90. 29, 49, 50

35 ATMEL. *Atmel 8-bit AVR Microcontroller with 2/4/8K Bytes In-System Programmable Flash: ATtiny25/V / ATtiny45/V / ATtiny85/V*. [S.l.], 2021. Disponível em: <http://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-2586-AVR-8-bit-Microcontroller-ATtiny25-ATtiny45-ATtiny85_Datasheet.pdf>. Acesso em: 6 janeiro 2021. 12, 29, 30, 43

36 TEXAS INSTRUMENTS. *INA118 Precision, Low-Power Instrumentation Amplifier*. [S.l.], 2020. Disponível em: <https://www.ti.com/lit/ds/symlink/ina118.pdf>. Acesso em: 6 dezembro 2020. 12, 31, 40, 41

37 NORDIC SEMICONDUCTOR. *nRF24L01+ Single Chip 2.4GHz Transceiver: Preliminary Product Specification v1.0*. [S.l.], 2021. Disponível em: <https://www.sparkfun.com/datasheets/Components/SMD/nRF24L01Pluss_Preliminary_Product_Specification_v1_0.pdf>. Acesso em: 14 janeiro 2021. 31, 32

38 DELSYS. *Bagnoli-2 EMG System: User's Guide*. 2021. Disponível em: <https://silo.tips/download/bagnoli-2-emg-system-user-s-guide>. Acesso em: 12 fevereiro 2021. 11, 12, 13, 33, 34, 35, 41, 47, 48

39 NATIONAL INSTRUMENTS. *USER GUIDE: NI USB-6001/6002/6003 Low-Cost DAQ USB Device*. 2021. Disponível em: <https://www.ni.com/pdf/manuals/374259a.pdf>. Acesso em: 13 fevereiro 2021. 35

40 OKAWA ELECTRIC DESIGN. *Filter Design and Analysis*. 2021. Disponível em: <http://sim.okawa-denshi.jp/en/Fkeisan.htm>. Acesso em: 12 janeiro 2021. 41, 42, 65

41 ENERGIZER. *Product datasheet: ENERGIZER 522*. 2021. Disponível em: <https://data.energizer.com/pdfs/522.pdf>. Acesso em: 19 janeiro 2021. 43

42 THOUGHT TECNLOLOGY LTD. *Basics of Surface Electromyography Applied to Physical Rehabilitation and Biomechanics*. [S.l.]: THOUGHT TECNLOLOGY LTD., 2010. 13, 48

43 GEORGE, K. S.; SIVANANDAN, K.; MOHANDAS, K. Estimation of elbow angle using surface electromyographic signals. *Journal of Intelligent & Fuzzy Systems*, IOS Press, v. 34, n. 6, p. 4191–4201, 2018. 50

Anexos

ANEXO A – FILTROS

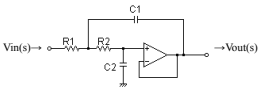
Cálculos de componentes dos filtros por meio da ferramenta de [40].

A.1 CÁLCULO DO FILTRO PASSA-BAIXAS E REPOSTA

Sallen-Key Low-pass Filter Design Tool - Result -

Calculated the Transfer Function for the Sallen-Key Low-pass filter, displayed on graphs, showing Bode diagram, Nyquist diagram, Impulse response and Step response

Sallen-Key Low-pass Filter



Transfer Function:

$$G(s)=\frac{9929429.5582436}{s^2+4456.3279857397s+9929429.5582436}$$

R1= 6.8k Ω C1= 66n F
R2= 6.8k Ω C2= 33n F
p:pico, n:nano, u:micro, k:kilo, M:mega

Frequency analysis

- [x] Bode diagram
  - (•) Phase ( ) Group delay
- [x] Nyquist diagram
- [x] Pole, zero
- [x] Phase margin
- [x] Oscillation analysis

Analysis on frequency range:
f1= –f2= [Hz] (optional)

Transient analysis

- [x] Step response
- [ ] Impulse response
- [x] Overshoot
- [x] Final value of the step response

Analysis on time range:
0– [sec] (optional)

Calculate

Cut-off frequency

fc = 501.51309946363[Hz]

Quality factor

Q = 0.70710678118655

Damping ratio

ζ = 0.70710678118655

Pole(s)

p = -354.62331348462 +354.62331348462i[Hz]
$|p|$= 501.51309946363[Hz]
p = -354.62331348462-354.62331348462i[Hz]
$|p|$= 501.51309946363[Hz]

Phase margin

pm= INF[deg] (f=0[Hz])

Oscillation frequency

f = 354.62331348462[Hz]

Overshoot (in absolute value)

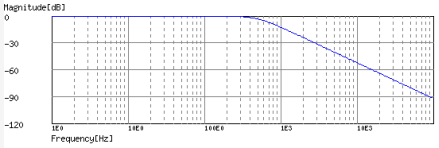
The 1st peak g_{pk} = 1.04 (t =0.0014[sec])
The 2nd peak g_{pk} = 1 (t =0.0028[sec])
The 3rd peak g_{pk} = 1 (t =0.0042[sec])

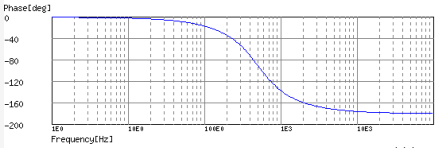
Final value of the step response (on the condition that the system converged when t goes to infinity)

$g(\infty)$ = 1

Frequency analysis

BodeDiagram





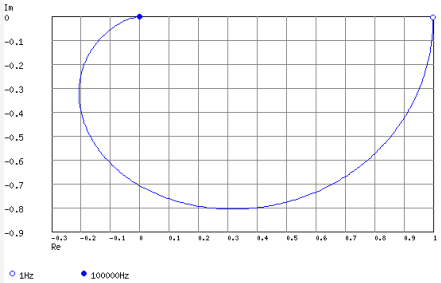
(c)okawa-denshi.jp

Gain characteristics at the Bode Diagram (provides up to 1 minute)

Phase characteristics at the Bode Diagram (provides up to 1 minute)

Bode Diagram text data (provides up to 1 minute)

NyquistDiagram

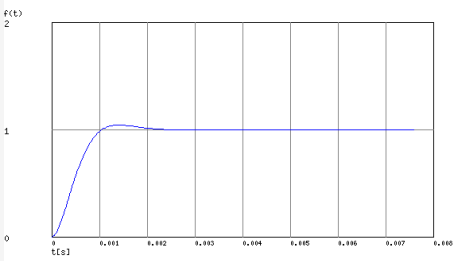


(c)okawa-denshi.jp

Nyquist Diagram text data (provides up to 1 minute)

Transient analysis

StepResponse



(c)okawa-denshi.jp

Step Response text data (provides up to 1 minute)

Suggestion box

We'll use your suggestion to improve site quality in future.

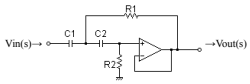
Post Comment

A.2 CÁLCULO DO FILTRO PASSA-ALTAS E RESPOSTA

Sallen-Key High-pass Filter Design Tool - Result -

Calculated the Transfer Function for the Sallen-Key High-pass filter, displayed on graphs, showing Bode diagram, Nyquist diagram, Impulse response and Step response

Sallen-Key High-pass Filter



Transfer Function:

$$G(s)=\frac{s^2}{s^2+200s+20000}$$

Cut-off frequency

fc = 22.507907903928[Hz]

Quality factor

Q = 0.70710678118655

Damping ratio

ζ = 0.70710678118655

Pole(s)

p = -15.91549430919 +15.91549430919i[Hz]
|*p*|= 22.507907903928[Hz]
p = -15.91549430919-15.91549430919i[Hz]
|*p*|= 22.507907903928[Hz]

Zero(s)

z = 0[Hz]
|*z*|= 0[Hz]
z = -0[Hz]
|*z*|= 0[Hz]

Phase margin

pm= INF[deg] (*f* =0[Hz])

Oscillation frequency

f = 15.91549430919[Hz]

Overshoot (in absolute value)

The 1st peak g_{pk} = -0.21 (*t* =0.016[sec])
The 2nd peak g_{pk} = 0.009 (*t* =0.047[sec])
The 3rd peak g_{pk} = -0.00039 (*t* =0.079[sec])

Final value of the step response (on the condition that the system converged when t goes to infinity)

$g(\infty) = 0$

R1= 50k Ω C1= 0.1u F
R2= 100k Ω C2= 0.1u F
p:pico, n:nano, u:micro, k:kilo, M:mega

Frequency analysis

- [x] Bode diagram
  - (•) Phase ( ) Group delay
- [x] Nyquist diagram
- [x] Pole, zero
- [x] Phase margin
- [x] Oscillation analysis

Analysis on frequency range:
f1= –f2= [Hz] (optional)

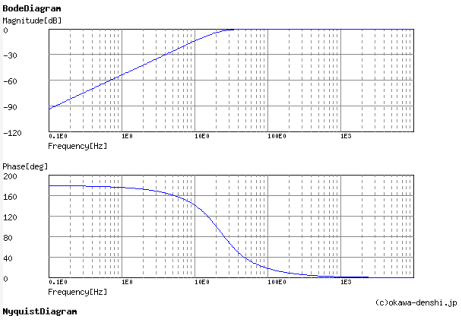
Transient analysis

- [x] Step response
- [ ] Impulse response
- [x] Overshoot
- [x] Final value of the step response

Analysis on time range:
0– [sec] (optional)

Calculate

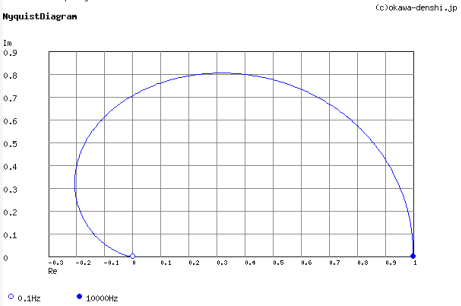
Frequency analysis



Gain characteristics at the Bode Diagram (provides up to 1 minute)

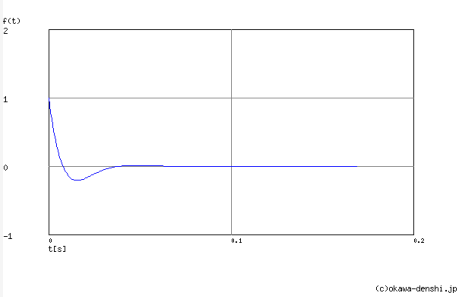
Phase characteristics at the Bode Diagram (provides up to 1 minute)

Bode Diagram text data (provides up to 1 minute)



Nyquist Diagram text data (provides up to 1 minute)

Transient analysis



Step Response text data (provides up to 1 minute)

Suggestion box

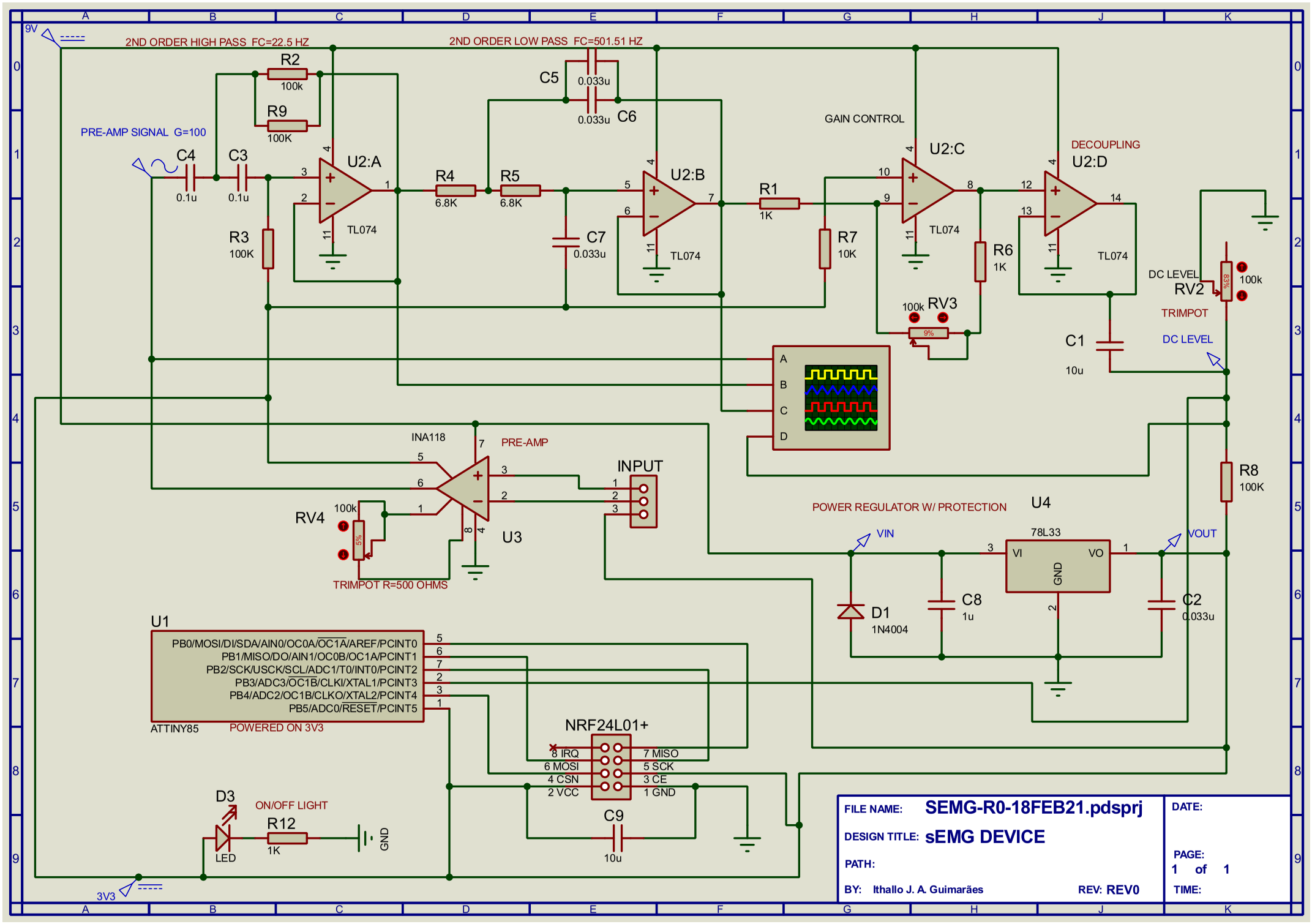
We'll use your suggestion to improve site quality in future.

Post Comment

Apêndices

APÊNDICE A – ESQUEMÁTICO

A.1 ESQUEMÁTICO DO DISPOSITIVO.


9V
2ND ORDER HIGH PASS FC=22.5 HZ
2ND ORDER LOW PASS FC=501.51 HZ
PRE-AMP SIGNAL G=100
GAIN CONTROL
DECOUPLING
R2
100k
R9
100K
C5
0.033u
C6
0.033u
C4
0.1u
C3
0.1u
U2:A
U2:B
U2:C
U2:D
TL074
R4
6.8K
R5
6.8K
R1
1K
R3
100K
C7
0.033u
R7
10K
R6
1K
100k RV3
9%
DC LEVEL
100k
RV2
83%
TRIMPOT
DC LEVEL
C1
10u
INA118
PRE-AMP
INPUT
RV4
100k
5%
U3
TRIMPOT R=500 OHMS
POWER REGULATOR W/ PROTECTION
U4
78L33
VIN
VOUT
VI
VO
GND
D1
1N4004
C8
1u
C2
0.033u
R8
100K
U1
PB0/MOSI/DI/SDA/AIN0/OC0A/OC1A/AREF/PCINT0
PB1/MISO/DO/AIN1/OC0B/OC1A/PCINT1
PB2/SCK/USCK/SCL/ADC1/T0/INT0/PCINT2
PB3/ADC3/OC1B/CLKI/XTAL1/PCINT3
PB4/ADC2/OC1B/CLKO/XTAL2/PCINT4
PB5/ADC0/RESET/PCINT5
ATTINY85
POWERED ON 3V3
NRF24L01+
8 IRQ
6 MOSI
4 CSN
2 VCC
7 MISO
5 SCK
3 CE
1 GND
C9
10u
D3
LED
ON/OFF LIGHT
R12
1K
GND
3V3
FILE NAME: SEMG-R0-18FEB21.pdsprj
DESIGN TITLE: sEMG DEVICE
PATH:
BY: Ithallo J. A. Guimarães
REV: REV0
DATE:
PAGE: 1 of 1
TIME:

APÊNDICE B – CÓDIGOS EMBARCADOS

B.1 CALIBRAGEM DO SINAL DE *CLOCK*

Arquivo attiny85_serial_calibrator.ino.

```
/*
A calibrator code that uses a serial connection
between the ATtiny85 and other Arduino
to adjust the internal oscillator of the tiny.

Use the bare minimum sketch on other Arduino to read the
ATtiny85's software serial and to set the values as
the codes show.

Remember that to keep the EEPROM data after reprogramming
the fuses must be set before using this code.

The fuses to be set on the ATtiny85 are:
-U lfuse:w:0xe2:m -U hfuse:w:0xf7:m -U efuse:w:0xff:m

It writes the callibration factor to the EEPROM, starting
at 0.
*/

#include <EEPROM.h>
#include<SoftwareSerial.h>

#define PIN 0
#define HALF_CYCLE 500 //useconds, 1kHz

const byte RX=4;
const byte TX=3;
unsigned long int t=0;
bool initial_state=0;
char response='g';
unsigned char value='0';

SoftwareSerial mySerial(RX, TX);
```

```
void setup() {
  pinMode(PIN, OUTPUT);
  pinMode(RX, INPUT);
  pinMode(TX, OUTPUT);

  mySerial.begin(9600);
  delay(1000);
  // press b to begin
  mySerial.println("Started");
  mySerial.println("disable line endings");
  mySerial.println("d to start");
  mySerial.println("(i)ncrease OSCCAL");
  mySerial.println("(d)ecrease OSCCAL");
  mySerial.println("(R)ead/(W)rite OSCCAL from/to EEPROM");

  mySerial.print("OSCCAL at beginning: ");
  mySerial.println(OSCCAL);

  delay(1000);

}


void loop(void){

  while (mySerial.available() > 0) {

    response = mySerial.read();
    mySerial.print("echo: ");
    mySerial.println(response);

    if (response=='b'){
      mySerial.println("signal begin");
      break;
    }
    else if(response=='i'){

      mySerial.print("OSCCAL before: ");
      mySerial.println(OSCCAL);

      OSCCAL += 1;
      response = 'b';
```

```
  }

  else if(response=='d'){

    OSCCAL -= 1;
    response = 'b';
  }

  else if (response=='R'){

    EEPROM.get(0, value);

    mySerial.print("read from EEPROM: ");
    mySerial.println( (int) value );

    response = 'b';
  }

  else if (response=='W'){

    EEPROM.update(0, (unsigned char) OSCCAL);

    mySerial.println("written to EEPROM");
    response = 'b';
  }
  else {
     mySerial.println("signal stop");
  }

  mySerial.print("OSCCAL now: ");
  mySerial.println(OSCCAL);

}


if (response=='b'){

  initial_state ^= 1;
  digitalWrite(PIN, initial_state);

  while( (micros() - t) <= HALF_CYCLE);
  t += HALF_CYCLE;
```

```
    }
}
```

B.2 TRANSMISSÃO

Arquivo transmitter.ino.

```
/*
SENDER which uses the pre-calibrated OSCCAL from EEPROM.

This code is for multiple channel transmission and reception. In order
to differentiate the devices they are marked with white dots, so
they are called 2SEMG, 1SEMG, (this naming follows historical reasons)
and so on. This is because of the 40 bit nature
of the pipe name (5 Bytes). They can just differ on the first byte.
*/

#include <EEPROM.h>
#include "nRF24L01.h"
#include "RF24.h"

#define DEVICE1 // if commented it activates the second device

#define CSN_PIN 4
#define CE_PIN 5 //reset pin
#define ANALOG_PIN 3 //PB3, pin 2
#define CHANNEL 76//0x4c

#define BUFFER_SIZE 1 //max 255, real buffer size

#define SAMPLING_FREQUENCY 2000 //HZ
#define DELTA 1000000/SAMPLING_FREQUENCY


// setting the name of the transmitter
#ifdef DEVICE1
  const unsigned char pipe[6] = "2SEMG";
#else
  const unsigned char pipe[6] = "1SEMG";
#endif

uint16_t data[BUFFER_SIZE];
```

```
unsigned int i=0;
unsigned long int t=0;
unsigned char value;

RF24 radio(CE_PIN, CSN_PIN);

void setup() {

  //loading calibrated OSCCAL
  EEPROM.get(0, value);
  OSCCAL = value;

  radio.begin();
  radio.setChannel(CHANNEL);
  radio.setAddressWidth(5);

  radio.setAutoAck(false);
  radio.setPayloadSize(sizeof(data)); // 2 x buffer size
  radio.setRetries(0,0);
  radio.setDataRate(RF24_2MBPS); //more than enough

  radio.stopListening();
  radio.openWritingPipe(pipe);

  analogReference(INTERNAL); // sets attiny85 reference voltage to 1v1

}


void loop(void){

    // fill the buffer
    for(i=0;i < BUFFER_SIZE;i++){

      data[i] = analogRead(ANALOG_PIN);

      while( (micros() - t) <= DELTA);
      t += DELTA;

    }

 //send data
```

```
 radio.writeFast(&data, sizeof(data) );

 }
```

B.3 RECEPÇÃO

Arquivo receiver.ino.

```
 /*
 * Receiver
 */
#include <SPI.h>
#include "nRF24L01.h"
#include "RF24.h"

#define CSN_PIN 7
#define CE_PIN 8
#define CHANNEL 76//0x4c
#define BUFFER_SIZE 1 //abs max per transmission 32 Bytes
#define COMPUTER_BUFFER_SIZE 1
#define DEVICES 1

//#define WAIT_ALL_DEVICES // if enabled, waits for signals from all devices


RF24 radio(CE_PIN, CSN_PIN);

const unsigned char pipes[][6] = {"2SEMG", "1SEMG"};
uint16_t receiver_buffer[DEVICES][COMPUTER_BUFFER_SIZE] = { 0 };

uint8_t pipe = 0;
uint8_t previous_pipe = 0;
uint8_t pipe_count = 0;

void receiver() {

  pipe_count = 0;
  previous_pipe = DEVICES + 1; // in order to be different for only one device
  while(pipe_count < DEVICES){

    if (radio.available(&pipe)){
```

```
      radio.read(&receiver_buffer[pipe-1], sizeof(uint16_t));

      #ifdef WAIT_ALL_DEVICES
        if (pipe != previous_pipe){
          pipe_count += 1;
          previous_pipe = pipe;
        }
      #else
        pipe_count += 1;
        previous_pipe = pipe;
      #endif
    }

  }

  for (int i=0;i<COMPUTER_BUFFER_SIZE;i++){

    for (int j=0; j<DEVICES; j++){

      Serial.write(receiver_buffer[j][i]/256);
      Serial.write(receiver_buffer[j][i]%256);
    }

  }

}


void setup() {

  Serial.begin(115200);
  delay(1000);

  radio.begin();
  radio.setChannel(CHANNEL);
  radio.setAddressWidth(5);
  //radio.enableAckPayload();

  radio.setAutoAck(false);
  radio.setRetries(0,0);
  //radio.setCRCLength(RF24_CRC_16);
  radio.setDataRate(RF24_2MBPS); //more than enough
```

```
  radio.setPayloadSize(BUFFER_SIZE * sizeof(uint16_t)); // 2 x buffer size


  for (int i=0; i<DEVICES; i++){

    radio.openReadingPipe(i+1, pipes[i]);
    delay(250);
  }

  //start
  radio.startListening();

}


void loop(void){
  receiver();
}
```

APÊNDICE C – CÓDIGOS EM PYTHON

C.1 VISUALIZAÇÃO EM TEMPO REAL

Arquivo real_time_plot.py

```
# -*- coding: utf-8 -*-
"""
Real time plot
It must be set with the settings.py file
Author: Ithallo Junior Alves Guimaraes
"""

import matplotlib.pyplot as plt
import numpy as np
from scipy.signal import lfilter
import time
import os

import settings
import modules


def plotter():
    """Plots the acquired signal along the time axis."""

    os.system("clear")

    p = modules.serial_port()

    print("Starting plotter...")

    p.flush()

    time.sleep(0.5)

    plt.ion()
    plt.figure(figsize=(18, 9))

    grid_spacing = np.arange(
        0,
```

```
    settings.time_window_to_show,
    settings.time_window_to_show/settings.xticks
)
samples = int(
    settings.sampling_frequency * settings.time_window
)
buffer_length = int(
    settings.sampling_frequency * settings.time_window_to_show
)

t = np.arange(
    0., settings.time_window_to_show,
    1./settings.sampling_frequency
)

y = np.zeros((samples, settings.number_of_channels))
plot_buffer = np.zeros((buffer_length, settings.number_of_channels))

# filter, getting the coefficients one time for all, as it is kinda slow
if(settings.use_filter):
    b, a = modules.get_filter_constants()

while(1):
    try:
        plt.clf()
        plt.ylabel("Volts(v)")
        plt.xlabel("Time(s)")
        plt.xlim(0, settings.time_window_to_show)

        if settings.remove_mean:
            plt.ylim(-0.6, 0.6)
        else:
            plt.ylim(0., settings.voltage_range)

        i = 0
        while i < samples:

            for j in range(settings.number_of_channels):
                v1 = p.read()
                v2 = p.read()
                y[i, j] = modules.convert_input(v1, v2)
                i += 1
```

```
        # circular buffer, simple now
        plot_buffer = plot_buffer[samples:, :]
        plot_buffer = np.vstack((plot_buffer, y[:buffer_length, :]))

        for i in range(settings.number_of_channels):

            # removing DC
            if (settings.remove_mean):
                yn = plot_buffer[:, i] - plot_buffer[:, i].mean()
            else:
                yn = plot_buffer[:, i]
            # checking if there is signal
            signal_flag = ''
            if not np.any(yn):
                signal_flag = '- no signal'

            # filtering signal
            if (settings.use_filter):
                yn = lfilter(b, a, yn)

            plt.plot(
                t, yn, c=settings.colors[i],
                label="Channel %s %s" % (i+1, signal_flag)
            )

            plt.xticks(grid_spacing)
            plt.grid(color='k', linestyle='-', linewidth=.1)
            plt.legend(loc="upper right")
            plt.title("Signal(s) || Fs: %.3f || Filter %s || Range: %smV" %
                (
                      settings.sampling_frequency,
                      'ON' if settings.use_filter else 'OFF',
                      round(yn.max() - yn.min(), 3)*1000.)
                      )

            plt.pause(1.0/60.0)

    except KeyboardInterrupt:
        plt.close()
        break
```

```
    p.close()


if (__name__ == "__main__"):
    plotter()
```

C.2 COLETA APENAS

Arquivo sampler.py

```
"""
This code works as a sampler for the EMG system (Arduino).
It must be configured via the settings.py file.
After sampling, it calls the plotter, if the argument
'--plot' was passed. Other possible arguments may be passed,
as detailed in the 'plot.py file'.

Author: Ithallo Junior Alves Guimaraes

"""

import os
import datetime
import time
from sys import argv

import serial
import numpy as np
from scipy.signal import lfilter

import modules
import settings
from plot import plot


# the run of the code
def sampler():
    """Samples the sEMG signal as declared on the settings file."""

    # filter, getting the coefficients one time for all, as it is kinda slow
    if (settings.use_filter):
        b, a = modules.get_filter_constants()
```

```
os.system("clear")
raw_input("Press enter to start...")

p = serial.Serial(port=settings.device, baudrate=settings.baud_rate,
                  bytesize=serial.EIGHTBITS, parity=serial.PARITY_NONE,
                  timeout=settings.timeout)
# timeout to be blocking

time.sleep(0.1) # settle time
p.flush() # deletes buffers

# control variables and max
total_samples = int(settings.sampling_frequency * settings.total_time)
delta_time = 1./settings.sampling_frequency
broke_out = False

# allocating
X = np.zeros((total_samples, 2))

# sampling
for i in xrange(total_samples):
    try:
        # reading value
        v1 = p.read()
        v2 = p.read()

        # converting to
        if (v1 == '') or (v2 == ''):
            print("\nNo data, check your circuits and run again.\n")
            broke_out = True
            break
        else:
            X[i, 0] = i * delta_time
            X[i, 1] = modules.convert_input(v1, v2)

        # loading screen
        percent = 100. * float(i)/total_samples
        print("Running --- %.3f%%" % percent)

    except KeyboardInterrupt:
        print("\nInterrupted\n")
```

```
            time.sleep(.5)
            broke_out = True
            break

    if (not broke_out):
        # saving to file
        filepath = settings.default_path + "data_%s.txt" % (
            str(datetime.datetime.now())[:-7]
        ).replace(" ", "_").replace(":", "-")

        # filtering signal
        if (settings.use_filter):
            X[:, 1] = lfilter(b, a, X[:, 1])

        np.savetxt(filepath, X, fmt=settings.format)
        print ("file saved to %s " % filepath)

    else:
        filepath = None

    # closing port
    p.flush()
    p.close

    return filepath


if (__name__ == "__main__"):

    filepath = sampler()
    if filepath is not None:
        plot(filepath, *argv)
```

C.3 APENAS VISUALIZAÇÃO/GERAR IMAGEM

Arquivo sampler.py

```
"""
This code works as a sampler for the EMG system (Arduino).
It must be configured via the settings.py file.
After sampling, it calls the plotter, if the argument
'--plot' was passed. Other possible arguments may be passed,
```

```
as detailed in the 'plot.py file'.

Author: Ithallo Junior Alves Guimaraes

"""

import os
import datetime
import time
from sys import argv

import serial
import numpy as np
from scipy.signal import lfilter

import modules
import settings
from plot import plot


# the run of the code
def sampler():
    """Samples the sEMG signal as declared on the settings file."""

    # filter, getting the coefficients one time for all, as it is kinda slow
    if (settings.use_filter):
        b, a = modules.get_filter_constants()

    os.system("clear")
    raw_input("Press enter to start...")

    p = serial.Serial(port=settings.device, baudrate=settings.baud_rate,
                      bytesize=serial.EIGHTBITS, parity=serial.PARITY_NONE,
                      timeout=settings.timeout)
    # timeout to be blocking

    time.sleep(0.1) # settle time
    p.flush() # deletes buffers

    # control variables and max
    total_samples = int(settings.sampling_frequency * settings.total_time)
    delta_time = 1./settings.sampling_frequency
```

```
broke_out = False

# allocating
X = np.zeros((total_samples, 2))

# sampling
for i in xrange(total_samples):
    try:
        # reading value
        v1 = p.read()
        v2 = p.read()

        # converting to
        if (v1 == '') or (v2 == ''):
            print("\nNo data, check your circuits and run again.\n")
            broke_out = True
            break
        else:
            X[i, 0] = i * delta_time
            X[i, 1] = modules.convert_input(v1, v2)

        # loading screen
        percent = 100. * float(i)/total_samples
        print("Running --- %.3f%%" % percent)

    except KeyboardInterrupt:
        print("\nInterrupted\n")
        time.sleep(.5)
        broke_out = True
        break

if (not broke_out):
    # saving to file
    filepath = settings.default_path + "data_%s.txt" % (
        str(datetime.datetime.now())[:-7]
    ).replace(" ", "_").replace(":", "-")

    # filtering signal
    if (settings.use_filter):
        X[:, 1] = lfilter(b, a, X[:, 1])

    np.savetxt(filepath, X, fmt=settings.format)
```

```
        print ("file saved to %s " % filepath)

    else:
        filepath = None

    # closing port
    p.flush()
    p.close

    return filepath


if (__name__ == "__main__"):

    filepath = sampler()
    if filepath is not None:
        plot(filepath, *argv)
```

C.4 ESPECTRO EM TEMPO REAL

Arquivo spectrum.py

```
# -*- coding: utf-8 -*-
"""
Spectrum plotter, it must be set with the settings.py file
Author: Ithallo Junior Alves Guimaraes
"""

import matplotlib.pyplot as plt
import numpy as np
from scipy.signal import lfilter
import time
import os

import settings
import modules


def spectrum_plotter():
    """Plots the signal's frequency spectrum."""

    os.system("clear")
```

```
p = modules.serial_port()
print("Starting spectrum plotter...")
p.flush()

time.sleep(0.5)

plt.ion()
plt.figure(figsize=(18, 9))

grid_spacing = np.arange(
    0, settings.max_expected_frequency_to_show,
    settings.max_expected_frequency_to_show/settings.xticks
)
samples = settings.frequency_window
frequencies = np.fft.fftfreq(samples)
f = abs(frequencies * settings.sampling_frequency)
y = np.zeros((samples, settings.number_of_channels))

# filter, getting the coefficients one time for all, as it is kinda slow
if(settings.use_filter):
    b, a = modules.get_filter_constants()

while(1):
    try:
        plt.clf()
        plt.xlabel("Frequency(Hz)")
        plt.xlim(0, settings.max_expected_frequency_to_show)
        if settings.normalize_spectrum:
            plt.ylabel("%Max")

        i = 0
        while i < samples:
            for j in range(settings.number_of_channels):

                v1 = p.read()
                v2 = p.read()
                y[i, j] = modules.convert_input(v1, v2)
                i = i + 1

        for i in range(settings.number_of_channels):

            # removing DC
```

```
            if (settings.remove_mean
                    or settings.always_use_remove_mean_for_spectrum):
                yn = y[:, i] - y[:, i].mean()
            else:
                yn = y[:, i]

            # filtering signal
            if (settings.use_filter):
                yn = lfilter(b, a, yn)

            Y = np.abs(np.fft.fft(yn))

            freq_hz = round(
                abs(frequencies[Y.argmax()] * settings.sampling_frequency),
                2
            )

            if settings.normalize_spectrum:
                y_max = Y.max()
                y_max = 1. if y_max==0 else y_max
                Y = Y/y_max

            plt.plot(
                f, Y, c=settings.colors[i],
                label="Channel %s, Fp: %s Hz" % (j+1, freq_hz)
            )

        plt.xticks(grid_spacing)
        plt.grid(color='k', linestyle='-', linewidth=0.1)
        plt.legend(loc="upper right")
        plt.title("Frequency domain || Fs: %.3f || Filter %s" % (
                  settings.sampling_frequency,
                  'ON' if settings.use_filter else 'OFF')
                  )
        plt.pause(1.0/60.0)
    except KeyboardInterrupt:
        plt.close()
        break

p.close()
```

```
if (__name__ == "__main__"):
    spectrum_plotter()
```

C.5 MÓDULO DE FERRAMENTAS

Arquivo modules.py

```
# -*- coding: utf-8 -*-
"""
Modules file for the plotters, which will be used for multiple codes.
Author: Ithallo Junior Alves Guimaraes
"""

import serial
from sys import exit
from scipy.signal import butter

import settings


def convert_input(a, b, raw=False):
    """Converts the two bytes into a single number."""

    if (a == '') or (b == ''):
        return 0. # if it was hanging defaults to 0

    value = ord(a[0]) * 256 + ord(b[0])

    if raw:
        return value

    return (settings.voltage_range * (value/1023.)) - settings.offset


def serial_port():
    """Encapsulates the opening of the serial port."""

    try:
        p = serial.Serial(
            port=settings.device, baudrate=settings.baud_rate,
            bytesize=serial.EIGHTBITS, parity=serial.PARITY_NONE,
            timeout=settings.timeout
```

```
        )
    except serial.SerialException:
        print("Device <%s> not found. Check your circuits." % settings.device)
        exit()

    return p


def get_filter_constants(analog=False):
    """Encapsulates the calculation of the filter constants."""

    b, a = butter(
        settings.order, settings.fc,
        fs=settings.sampling_frequency,
        btype=settings.filter_type, analog=analog
    )

    return b, a
```

C.6 MÓDULO DE CONFIGURAÇÕES

Arquivo settings.py

```
# -*- coding: utf-8 -*-
"""
Settings file for the plotters
Author: Ithallo Junior Alves Guimaraes
"""
# serial settings
device = "/dev/cu.wchusbserial1410" # change accordingly
baud_rate = 115200 # depends on your arduino code
timeout = 5. # seconds, in order to prevent the code from holding

# frequency settings
sampling_frequency = 2000. # Hz
time_window = 500e-3 # second(s),fixed to avoid performance issues on display
time_window_to_show = 4000e-3
frequency_window = 2048 # samples
total_time = 4. # *60. # seconds, to be used to get the total number of samples
max_expected_frequency = 500. # Hz
max_expected_frequency_to_show = 2. * max_expected_frequency # Hz
normalize_spectrum = True # normalizes to the max
```

```
# signal settings
voltage_range = 1.1 # max selected in the attiny85 sampler
offset = 0. # voltage offset to correct signal
remove_mean = False # removes mean for the window
always_use_remove_mean_for_spectrum = True # forces removing mean for spectrum


# data saving settings
format = "%.4f" # I think 4 values after the point is enough
default_path = 'acquisitions/' # leave it empty to be on the same folder

# transmitter settings
number_of_channels = 1 # number of channels to be displayed/sampled

# plot settings
colors = ["b", "r", "g", "y"]
xticks = 10

# Filter settings, here a Butterworth will be used.
use_filter = False # whether to use or not
filter_type = "bandpass" # lowpass, highpass or bandpass
fc = [2., 500.] # cutoff frequencies, change depending on the type
order = 4 # order of the filter
```

C.7 CORREÇÃO DO CSV DO NI MAX

Arquivo cleaner.py

```
"""
Cleans the files that come from the acquisitions with a NI device.
Dots must not be in the filepath, except for the extension.
Spaces are used as delimiters.
Author: Ithallo Junior Alves Guimaraes
"""

from sys import argv


def cleaner(filepath, delimiter=' '):
    """
```

```
    Cleans data from an NI device to a format readable by numpy.
    It uses spaces as delimiters.
    """

    with open(filepath, 'r') as f:
        lines = f.readlines()[1:] # ignoring the header

    new_lines = ''
    for line in lines:

        line = line.replace(',', '.') # data uses commas for decimals
        line = line.replace('\r', '')
        line = line.replace('\n', '')
        line = line.replace('\t', delimiter)
        line = line.replace(';', delimiter)

        if 'm' in line:
            split_line = line.split(delimiter)
            voltage = str(float(split_line[1][:-1])/1000.)
            line = ('%s' % delimiter).join([split_line[0], voltage])

            new_lines += (line + '\n')

    split_filepath = filepath.split('.')
    new_filepath = split_filepath[0] + '_cleaned.txt'

    with open(new_filepath, 'w') as f:
        f.write(new_lines)


if __name__ == '__main__':

    try:
        filepath = argv[1]
        cleaner(filepath)
    except (IndexError, IOError):
        raise Exception('You must pass a valid filepath as argument')
```

APÊNDICE D – ANÁLISE DOS SINAIS

analysis

April 7, 2021

```
In [1]: import matplotlib.pyplot as plt
        import numpy as np
        from scipy.signal import butter, lfilter, welch
        %matplotlib inline
```

1 Loading signals

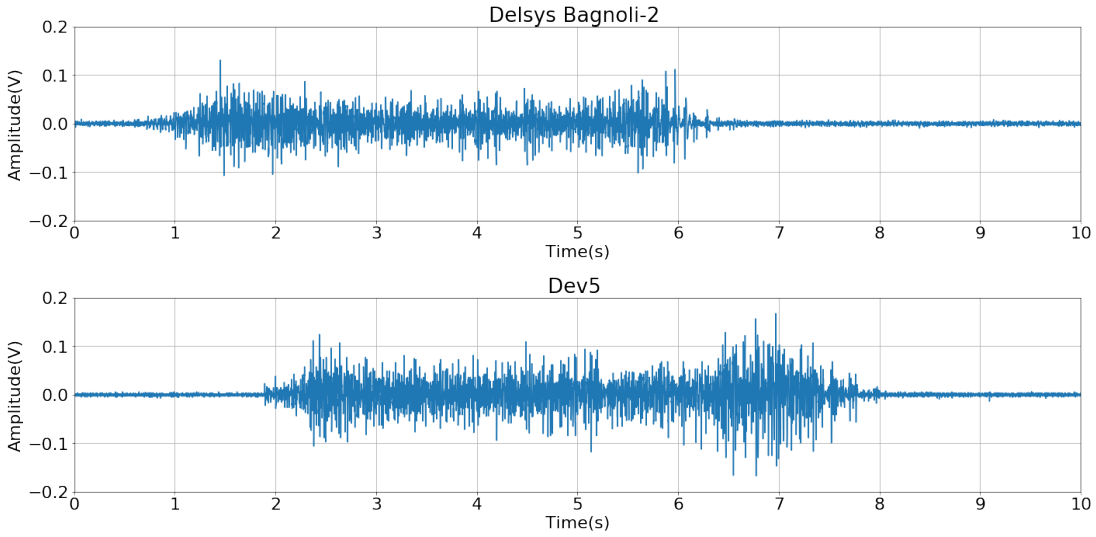
```
In [2]: fs = 2000. #Hz

        delsys_raw =  np.loadtxt("signals/delsys_signal.txt")
        delsys = delsys_raw[:,1] - delsys_raw[:,1].mean()

        dev5_raw = np.loadtxt("signals/dev5_signal.txt")[:,1]
        dev5 = dev5_raw - dev5_raw.mean()

        delsys_noise_raw =  np.loadtxt("signals/delsys_noise.txt")[:, 1]
        delsys_noise = delsys_noise_raw - delsys_noise_raw.mean()

        dev5_noise_raw = np.loadtxt("signals/dev5_noise.txt")[:, 1]
        dev5_noise = dev5_noise_raw - dev5_noise_raw.mean()
```

2 Signals on time domain (full)

```
In [3]: def plot_on_time(delsys, dev5, text='', y_range=(-0.2, 0.2)):
            """Standard way of plotting on time domain."""

            y_range = y_range
            divisions = 11


            plt.figure(figsize=(20, 10))
            plt.rcParams['font.size'] = 22


            t = np.linspace(0, delsys.shape[0]/fs, num=delsys.shape[0])
            x_range = (0., t[-1])
```

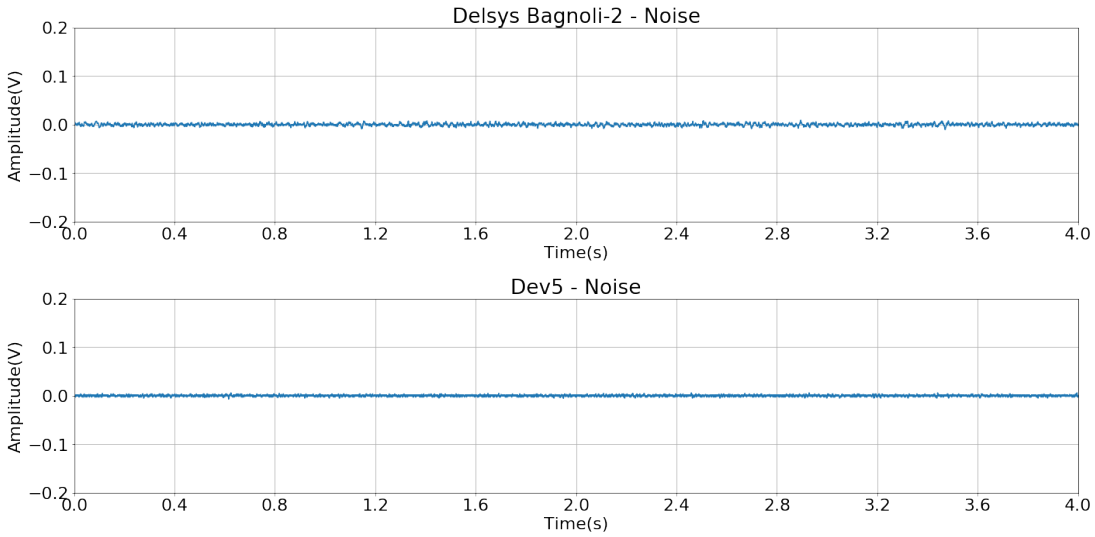
```
    spacing = np.linspace(*x_range, num=divisions)

    plt.subplot(211)
    plt.title("Delsys Bagnoli-2 %s"%text)
    plt.plot(t, delsys)
    plt.grid()
    plt.ylim(*y_range)
    plt.xlim(*x_range)
    plt.xticks(spacing)
    plt.ylabel("Amplitude(V)")
    plt.xlabel("Time(s)")


    t = np.linspace(0, dev5.shape[0]/fs, num=dev5.shape[0])
    x_range = (0., t[-1])
    spacing = np.linspace(*x_range, num=divisions)


    plt.subplot(212)
    plt.title("Dev5 %s"%text)
    plt.plot(t, dev5)
    plt.grid()
    plt.ylim(*y_range)
    plt.xlim(*x_range)
    plt.xticks(spacing)
    plt.ylabel("Amplitude(V)")
    plt.xlabel("Time(s)")

    plt.tight_layout(pad=1.0)
    plt.show()
```

In [4]:
```
plot_on_time(delsys, dev5)
```



2.0.1 Noise

```
In [5]: plot_on_time(delsys_noise, dev5_noise, text='- Noise')
```

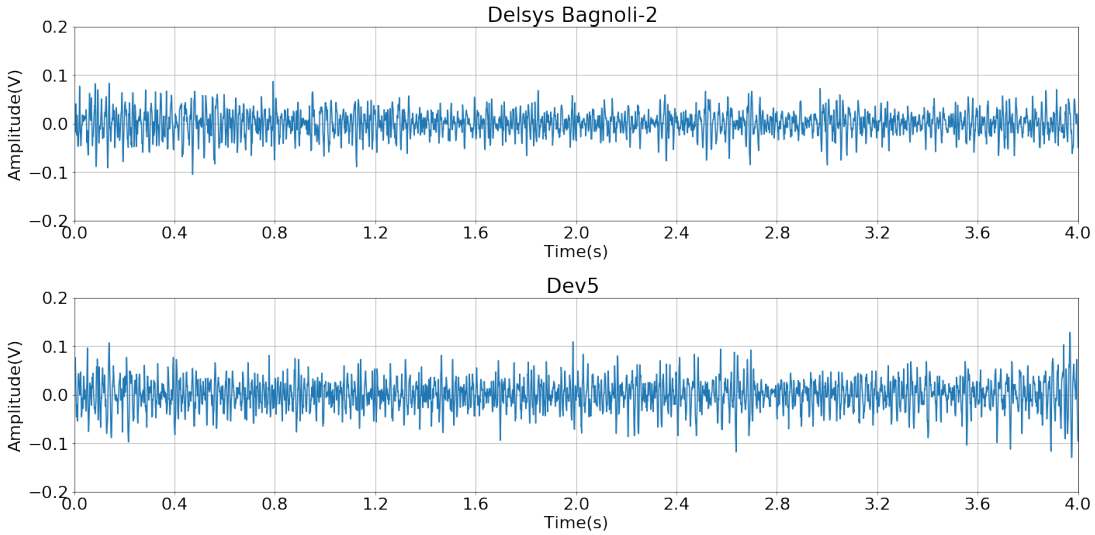


2.1 Cropping signal

Signals are cropped so that only the contraction exists

```
In [6]: time_to_crop_begin_delsys = 1.5
        time_to_crop_end_delsys = 5.5


        time_to_crop_begin_dev5 = 2.5
        time_to_crop_end_dev5 = 6.5


        delsys_cropped = delsys[
            int(time_to_crop_begin_delsys*fs):int(time_to_crop_end_delsys*fs)
        ]

        dev5_cropped = dev5[
            int(time_to_crop_begin_dev5*fs):int(time_to_crop_end_dev5*fs)
        ]
```

```
In [7]: plot_on_time(delsys_cropped, dev5_cropped)
```

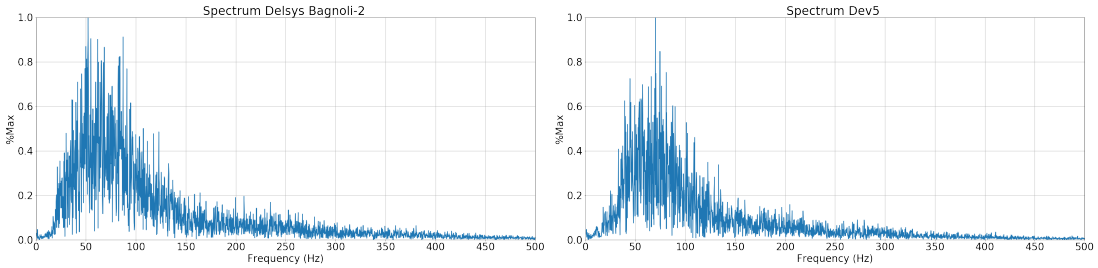


3 Signals on frequency domain

```
In [8]: def plot_on_frequency(delsys, dev5, text=''):
            """Standard way to plot the FFT."""

            t_window = 4. #seconds
            samples = int(fs * t_window)
            frequencies = np.fft.fftfreq(samples)
            F = np.abs(frequencies * fs)

            window = np.fft.fftshift(np.hanning(samples))

            delsys_spectrum =  np.abs(np.fft.fft(delsys)) * window
            dev5_spectrum = np.abs(np. fft.fft(dev5)) * window

            normalized_delsys_spectrum = delsys_spectrum/delsys_spectrum.max()
            normalized_dev5_spectrum = dev5_spectrum/dev5_spectrum.max()


            x_range = (0., 500.) # Hz
            y_range = (0.,1.) # %
            divisions = 11
            spacing = np.linspace(*x_range, num=divisions)

            plt.figure(figsize=(40, 10))
            plt.rcParams['font.size'] = 26
```

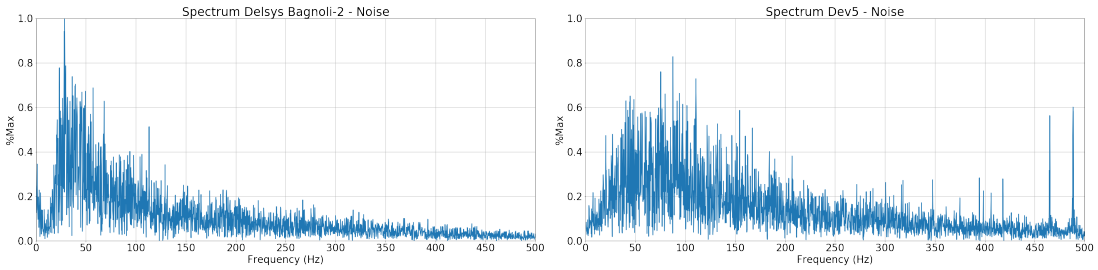
```
    plt.subplot(121)
    plt.title("Spectrum Delsys Bagnoli-2 %s"%text)
    plt.plot(F, normalized_delsys_spectrum)
    plt.grid()
    plt.xlim(*x_range)
    plt.xticks(spacing)
    plt.ylim(*y_range)
    plt.ylabel("%Max")
    plt.xlabel("Frequency (Hz)")


    plt.subplot(122)
    plt.title("Spectrum Dev5 %s"%text)
    plt.plot(F, normalized_dev5_spectrum)
    plt.grid()
    plt.xlim(*x_range)
    plt.xticks(spacing)
    plt.ylim(*y_range)
    plt.ylabel("%Max")
    plt.xlabel("Frequency (Hz)")

    plt.tight_layout(pad=1.0)
    plt.show()
```

```
In [9]: plot_on_frequency(delsys_cropped, dev5_cropped)
```



3.0.1 Noise

```
In [10]: plot_on_frequency(delsys_noise, dev5_noise, '- Noise')
```



4 SNR and RMS

$$SNR = 20log_{10}\left(\frac{S_{RMS}}{N_{RMS}}\right)$$

Where, S_{RMS} is the signal RMS value, and N_{RMS} is the noise RMS value.

RMS (root mean square):

$$x_{RMS} = \sqrt{\frac{1}{N}\sum_{i=1}^{N} x_i^2}$$

```
In [11]: def rms(x_array):
             """Calculates the RMS value for the whole series."""

             n = float(x_array.shape[0])
             return  np.sqrt((x_array**2).sum()/n)

         snr = lambda  s, n: 20. * np.log10(s/n)

In [12]: delsys_rms = rms(delsys_cropped)
         delsys_noise_rms = rms(delsys_noise)

         dev5_rms = rms(dev5_cropped)
         dev5_noise_rms = rms(dev5_noise)



         print('RMS noise Delsys Bagnoli-2: %.2f mV(rms)'%(
             delsys_noise_rms*1000))
         print('RMS noise dev5: %.2f mV(rms)'%(dev5_noise_rms*1000))

         print('\n')

         print('RMS signal Delsys Bagnoli-2: %.2f mV(rms)'%(delsys_rms*1000))
         print('RMS signal dev5: %.2f mV(rms)'%(dev5_rms*1000))


         print('\n')

         print("SNR Delsys Bagnoli-2: %.2f dB" % snr(
             delsys_rms, delsys_noise_rms ))
         print("SNR Dev5 SNR: %.2f dB" % snr(dev5_rms, dev5_noise_rms))

RMS noise Delsys Bagnoli-2: 2.48 mV(rms)
RMS noise dev5: 1.55 mV(rms)
```

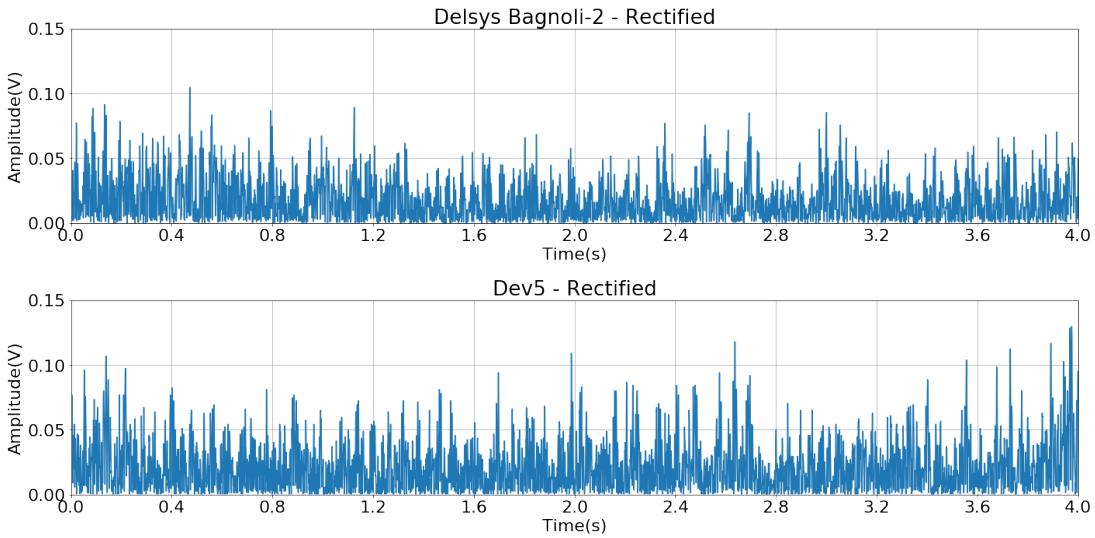
```
RMS signal Delsys Bagnoli-2: 24.84 mV(rms)
RMS signal dev5: 30.24 mV(rms)


SNR Delsys Bagnoli-2: 20.01 dB
SNR Dev5 SNR: 25.81 dB
```

4.1 Average Rectified Value (ARV) and linear envelope

```
In [13]: delsys_rectified = np.abs(delsys_cropped)
         dev5_rectified = np.abs(dev5_cropped)


         print("Delsys' ARV: %.2f mV"%(delsys_rectified.mean() *1000))
         print("Dev5's ARV: %.2f mV"%(dev5_rectified.mean() *1000))
```

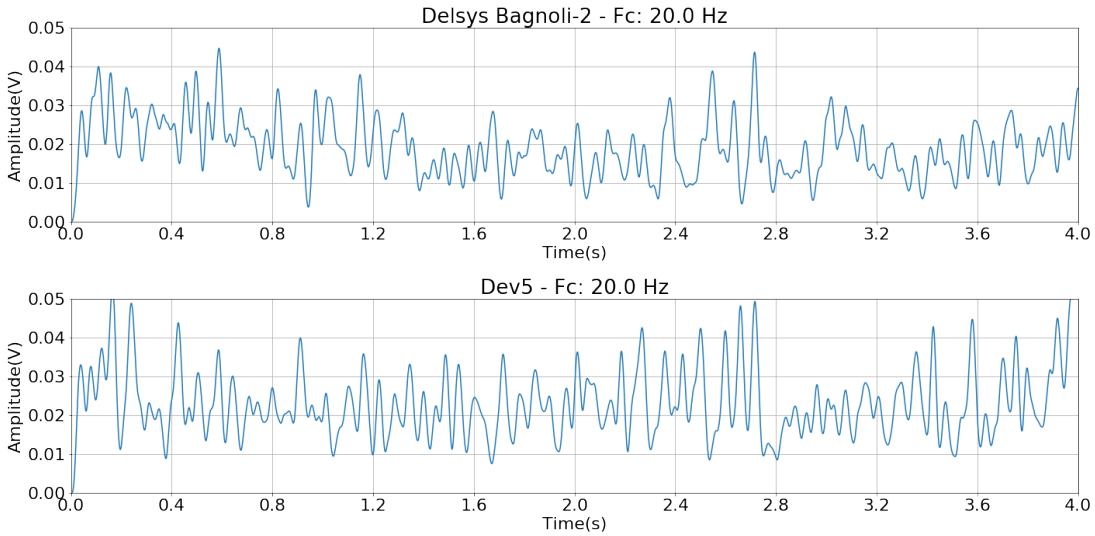
```
Delsys' ARV: 19.29 mV
Dev5's ARV: 23.46 mV
```

```
In [14]: plot_on_time(delsys_rectified, dev5_rectified,
                      text='- Rectified', y_range=(0, 0.15))
```



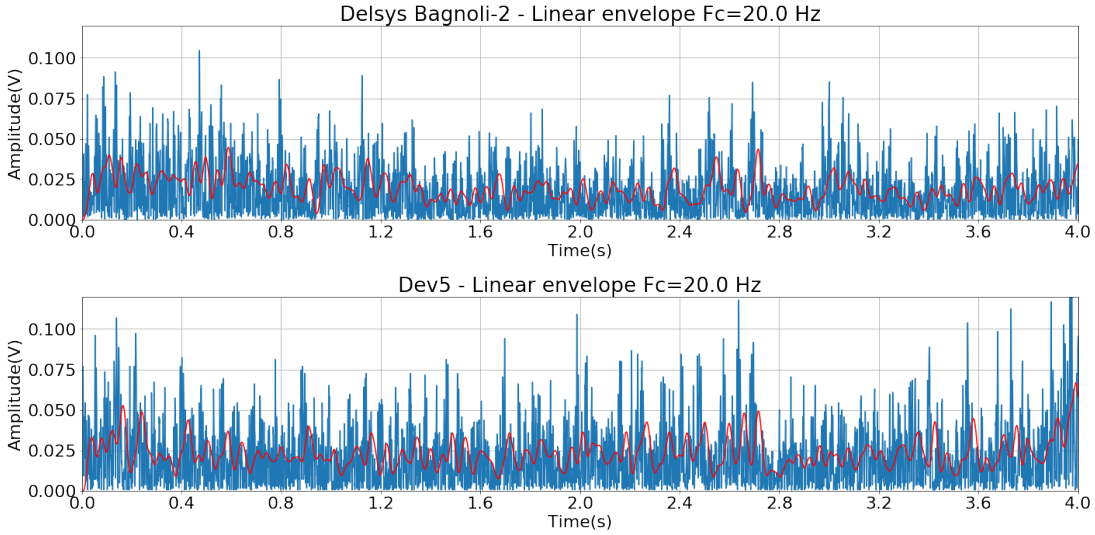

```
In [15]: def lowpass(signal, fc=20., order=4):
             b, a = butter(order, fc, btype='lowpass',fs=fs, analog=False)
             return lfilter(b,a, signal)
```

In [16]:
```
fc = 20.
delsys_lp = lowpass(delsys_rectified, fc)
dev5_lp = lowpass(dev5_rectified, fc)

plot_on_time(delsys_lp, dev5_lp, text='- Fc: %s Hz'%fc,  y_range=(0, 0.05))
```



In [17]:
```
# Plotting the envelope and the rectified signal

y_range = (0,0.12)
divisions = 11
text='- Linear envelope Fc=%s Hz'%fc

plt.figure(figsize=(20, 10))
plt.rcParams['font.size'] = 22


t = np.linspace(0, delsys_rectified.shape[0]/fs, num=delsys_rectified.shape[0])
x_range = (0., t[-1])
spacing = np.linspace(*x_range, num=divisions)

plt.subplot(211)
plt.title("Delsys Bagnoli-2 %s"%text)
plt.plot(t, delsys_rectified)
plt.plot(t, delsys_lp, c='r')
plt.grid()
plt.ylim(*y_range)
plt.xlim(*x_range)
plt.xticks(spacing)
plt.ylabel("Amplitude(V)")
```

```
plt.xlabel("Time(s)")


t = np.linspace(0, dev5_rectified.shape[0]/fs,
                num=dev5_rectified.shape[0])
x_range = (0., t[-1])
spacing = np.linspace(*x_range, num=divisions)


plt.subplot(212)
plt.title("Dev5 %s"%text)
plt.plot(t, dev5_rectified)
plt.plot(t, dev5_lp, c='r')
plt.grid()
plt.ylim(*y_range)
plt.xlim(*x_range)
plt.xticks(spacing)
plt.ylabel("Amplitude(V)")
plt.xlabel("Time(s)")

plt.tight_layout(pad=1.0)
plt.show()
```



Area below the curve via Riemann sum

$$\sum_{i=1}^{n} f(x_i)\Delta x$$

```
In [18]: delta_t = 1./fs
```

```
print("Delsys' area %.2f mV.s"%((delta_t*delsys_lp*1000).sum()))
print("Dev5's area %.2f mV.s"%((delta_t*dev5_lp*1000).sum()))
```

```
Delsys' area 76.63 mV.s
Dev5's area 92.95 mV.s
```

5 Mean (MNF) and median (MDF) frequencies

5.0.1 Mean frequency (MNF):

$$MNF = \frac{\sum_{j=1}^{M} f_j P_j}{\sum_{j=1}^{M} P_j}$$

5.0.2 Median frequency (MDF) :

$$\sum_{j=1}^{MDF} P_j = \sum_{j=MDF}^{M} P_j = \frac{1}{2} \sum_{j=MDF}^{M} P_j$$

In [19]:
```
f_delsys, delsys_power_spectrum = welch(delsys_cropped, fs)
f_dev5, dev5_power_spectrum = welch(dev5_cropped, fs)


# Mean frequencies
delsys_mean_frequency = (delsys_power_spectrum*f_delsys
                         ).sum()/delsys_power_spectrum.sum()
dev5_mean_frequency = (dev5_power_spectrum*f_dev5
                       ).sum()/dev5_power_spectrum.sum()
```

In [20]:
```
print("Delsys' MNF  %.2f Hz"%delsys_mean_frequency)
print("Dev5's MNF %.2f Hz"%dev5_mean_frequency)
```

```
Delsys' MNF  82.06 Hz
Dev5's MNF 84.04 Hz
```

In [21]:
```
def estimate_mdf(power_spectrum, frequencies):
    """Estimates the MDF."""

    half_power = power_spectrum.sum()/2.

    summed_power = 0
    for i, value in enumerate(power_spectrum):

        summed_power += value
        if summed_power >= half_power:
            break

    return frequencies[i]
```

In [22]:
```
# Median frequencies
delsys_median_frequency = estimate_mdf(delsys_power_spectrum, f_delsys)
dev5_median_frequency = estimate_mdf(dev5_power_spectrum, f_dev5)
```

In [23]:
```
print("Delsys' MDF %.2f Hz"%delsys_median_frequency)
print("Dev5's MDF %.2f Hz"%dev5_median_frequency)
```

```
Delsys' MDF 70.31 Hz
Dev5's MDF 70.31 Hz
```

APÊNDICE E – PUBLICAÇÕES

E.1 *PREDICTING KNEE ANGLES FROM VIDEO: AN INITIAL EXPERIMENT WITH MACHINE LEARNING*

Predicting knee angles from video: an initial experiment with Machine Learning

Guimarães, I.J.A.[1[0000-0002-1303-3317]]; Lopes, R.M.[2]; Junior, J.F.L.S.[1[0000-0002-0874-6056]]; Sousa, B.S.[1]; Marães, V.R.F.S.[1,2] and Brasil, L.M.[1[0000-0002-7437-107X]]

[1] Universidade de Brasília (UnB) at Gama, Post-Graduation Program in Biomedical Engineering, Brasília - DF, Brazil

[2] Universidade de Brasília (UnB) at Ceilândia, Brasília - DF, Brazil
ithallojunior@outlook.com

Abstract. Machine Learning (ML) has drawn a lot of attention these days due to its capability to automate processes that where very complicated and/or only performed by humans before. It is done by not having the need to write hard coded rules to solve problems, letting the machine find the rules by itself, and for being able to universally approximate functions with certain algorithms. On the other side, motion capture systems are quite expensive, making it more difficult to health professionals and physicians to have a more precise way to analyze the gait of patients and the diseases related to it. Having that said, this paper aimed to show that is possible to predict (generate) the angles of the right knee of patients directly from a common video of their walks, reducing costs and opening the opportunity for a more complete and affordable system to realize motion capture. Thus, this study is an initial experiment and the first step towards a possible future more complete product. As an initial study, the data of three patients were acquired and the resulting model, a Multilayer Perceptron Artificial Neural Network (MLP), acquainted a score of 92.2% when predicting the angles of the right knee for unknown data.

Keywords: Machine Learning (ML), Multilayer Perceptron (MLP), Knee Angles.

1 Introduction

The biomechanics of the musculoskeletal system of the human body has been studied over many years as an attempt to better understand the behavior and interactions between its components. The joints knees, hip and intervertebral discs are the components of the human body traditionally studied by the variability and complexity of the problems that affect them.

According to [1], numerous studies at the computational level have been developed to try to translate the biomechanical behavior of the knee joint, and to perceive the problems that affect this joint. Works, like the ones of [2] and [3], developed analytical models of the knee in the sagittal plane to study the effect of ligaments on joint

control and forces, as well as there is a three-dimensional knee model using finite elements that was developed by [4]. In this context, the use of a computer aid and automation seems reasonable, having that been said, the idea of ML and its techniques comes easily to mind, because of the variety of their potential uses in Biomedical Engineering and related fields.

1.1 Machine Learning and Artificial Neural Networks

ML is defined by [5] as a set of techniques which can detect patterns in data, then, using these uncovered, predict the future data, or even to perform different kinds of decision making under uncertainty.

ML can be divided into three main types: supervised learning, unsupervised learning and reinforcement learning. According to [5] and [6], it can be defined as follows: supervised learning uses a determined set of known (labeled) input-output pairs (training set) in order to train the algorithm; unsupervised learning, in an opposite way, does not know beforehand the outputs for given inputs (unlabeled data or unknown structure of data) and it is used to find patterns and meaningful information in data without the usage of an obvious error metric; by last, reinforcement learning is said use reward or punishment signal (feedbacks for measures of how well the system behaved) in order to improve the performance of the system based on its interactions with the "environment".

Bearing in mind the context of ML, this paper wants to use a feed-forward MLP to get continuous values (regression). This algorithm fits under the supervised type of learning.

MLP is said to be an artificial neural network (ANN) that can solve nonlinearly separable problems, which includes a nonlinear separable activation function, contains one or more "hidden" layers (in between the input and output), and a high degree of "connectivity" (its extent is determined by its synaptic weights) [7].

The use of ANNs and their capabilities is demonstrated by the works of [8] and [9], which have shown the use of two different kinds of ANNs (a Cerebellar Model Articulation Control and an MLP, respectively) to predict the angular velocities of human knee from signals of the contralateral knee [8] and from some signals of both knees [9].

This paper follows the aforementioned works, but takes a different approach. It aims to develop an automated system to predict the angles of the knee using only the video frames of the walks and, therefore, to help physicians and health-related professionals that have to use motion capture systems on their works.

1.2 Human Gait

Human gait is characterized by a sequence of multiple fast and complex events, so clinical observation, identification of changes in phenomena, and observation of their degree of withdrawal from normal become difficult [10]. It is a repetitive sequence of a movement of subordination to movement. In the gait a member acts as a moving

support, in contact as solo as the opposing member advances it in the air, the set of tasks that can be repeated cyclically and the members of their roles at every step [11]. The Fig. 1 shows the human gait in a complete cycle.

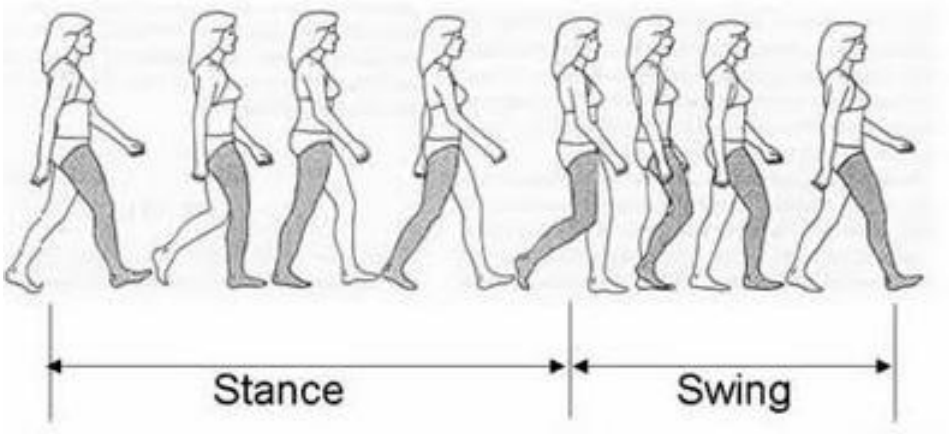


Fig. 1. Human gait. Adapted from [11].

Many patients may benefit from instrumented gait analysis, including: cerebral palsy, cranioencephalic trauma, neuromuscular diseases, traumatic spinal cord injuries, congenital and lower limb amputations. The instrumented gait analysis is used to aid in oncologic surgery, physical therapy programs, use of peripheral neuromuscular blocks, indication and adequacy of orthotics and prostheses. Although this analysis is carried out in several laboratories in several countries, it is still not very widespread in Brazil. And the main factor limiting its diffusion is the high cost of commercial systems available [11].

One of the most studied joints during gait analysis is that of the knee. It has a very complex structure and is also one of the most susceptible to involvement by various lesions. Thus, the quantification of movement and the detection of changes not perceived by the naked eye add many advantages to research and understanding of gait [10].

2 Materials and Methods

As this paper is based on previous works of [8] and [9], its materials and methods follow a similar fashion, having some changes due to the different nature of this work. It is presented as follows.

2.1 Data acquisition

The data used were obtained from three patients on the Human Performance Laboratory at Faculty UnB-Ceilândia, University of Brasilia (UnB). The patients were young males, who repeated a walk of five seconds for few times in front the system,

all the walks were required to start at the same point and angle to the cameras, in order to homogenize the data and prevent random fluctuations.

The system used was the Qualisys Oqus MRI and the Qualisys QTM 3.2 software package, this system was composed of twelve cameras to acquire the position of the body markers. Nine passive body markers were distributed along the inferior limbs of the subject and another camera was synced to the system to record the video of the walks.

The cameras used to acquire the body markers' positions had a frame rate of 200 frames per second (fps) and the one used to record video had one of 30 fps and a resolution of 640 by 480 pixels.

This data acquisition process was approved by the Health Sciences Faculty Ethics Committee from UnB, protocol number 38386714.8.0000.0030 on March 11, 2016.

2.2 Preprocessing

Not all the data acquired were used for the predictions. Firstly, only the data from the points marked as knee, trochanter and tibia were selected and they were fitted into one gait cycle. The walks that contained invalid data points inside the gait cycle were discarded. The data visualization and marker number selection were made utilizing the system developed by [12], in which a gait analysis and simulation system was created.

After defining the beginning and the end of the gait cycles, the videos were trimmed to fit that duration, had its colors removed and cut, in order to remove unnecessary data (such as the hands of the patients) and focus on the moving legs. It resulted in frames with a resolution of 200 by 250 pixels. The conversion of the video into vectors was made utilizing a Python library for video processing called Scikit-video, which provides easily accessible tools for reading and writing videos with Python. Those values were chosen because they were the ones that more accurately fitted the beginning and the ending of a gait cycle. The Fig. 2 shows the initial and final frames after preprocessing.

After this process, the data selected were passed on to the kinematics calculations.

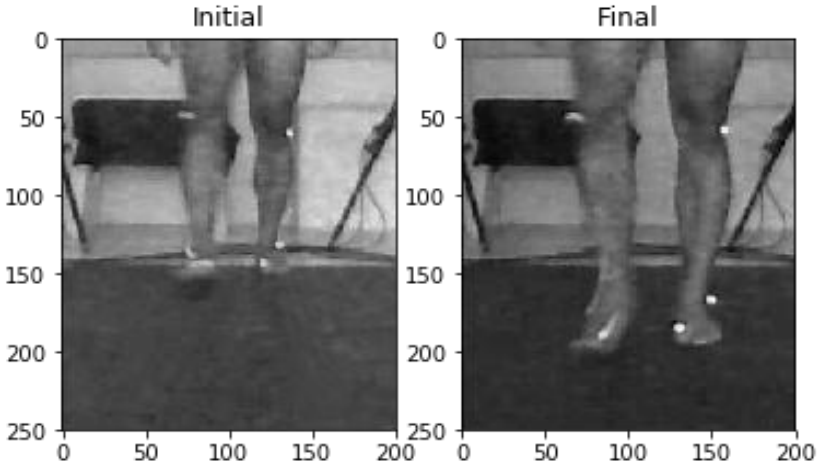


Fig. 2. Initial and final frames of a gait cycle after video preprocessing.

2.3 Kinematic Calculations

Python (2.7.12) was used for these computation processes. In these processes, Python libraries such as SciPy and Numpy inside an Anaconda environment were used in order to read and process the data. Then, angles of the knees were calculated for both legs. Equation (1) [13] shows how the angles were obtained using trochanter, knee and tibia.

$$\theta = \cos^{-1} \frac{u \cdot v}{u \quad v} \qquad (1)$$

The knee was used as origin of the system, and then relative positions of the other points were calculated using it. After this, the angles could be calculated using (1). In order to make this work more feasible, only the right knee angles were selected to be used in this initial experiment.

2.4 Algorithm used

The MLP algorithm was chosen because of its capacities of being a universal approximator, provided that enough hidden nodes are available, as established by [14].

The purpose of this paper is not to develop a state of the art ML technique, but to show the possibilities of its use, because of it, the Scikit-learn (version 0.19) Python package [15] was used instead of developing it all over. That package provides the state of the art algorithms for ML, metrics for the quality and preprocessing for the data [15].

2.5 Implementation details

This paper aimed to show that the video record of a walk could be used to get the angles of the knee during that, for this reason, the video frames were used as inputs and the knee angle as output. Only the right leg was used for this implementation.

To use the videos as inputs, its 200 by 250 pixels of about 41 frames per gait cycle were reshaped into strings of 50,000 data points per frame. As there was a difference between the number of output points and the input frames, because of the different frame rates, the knee angles outputs were resampled evenly accordingly.

The data from the gait cycles were split randomly into three groups: one for training, a second and validating the model and improving the results that would be obtained, and a third for testing the model after its implementation.

The datasets were separated using a function from the Sci-kit-learn package [15] and kept the proportion of 70% for training and 15% for validating, and, 15% for testing, randomly separated. After splitting the data, the groups were scaled up using another Scikit-learn library [15] in order to speed up the convergence process.

The MLP used had two hidden layers of 10 and 15 nodes respectively, used the hyperbolic tangent function as nonlinear activation function, and was run for 1000 iterations each time it was validated. Those parameters where selected empirically for being the ones that provided the best results when tinkering with the problem.

3 Results and Discussion

After testing many different configurations for the MLP using the validation set to improve the result and settling down the values established in the previous section, a score of 92.2% (or 0.922 as it can go up to 1) was obtained from the test dataset, demonstrating that it is perfectly possible to predict joint angles from a video input.

That score was calculated using the coefficient of determination (R^2 score) which shows to what degree the predictions of the regression model surpass the predictions based on the mean value of the desired output, this way, it provides a measure of how well future samples are likely to be predicted by the model. In that coefficient, the closer it gets to 1 the better it is, it can be negative as well (as the results of the model could be arbitrarily worse) [16, 17]. Equation (2) shows how the score could be calculated, where $\hat{y}_i$ represents the value predicted by the i-th input sample, y_i is the actual result of that i-th sample, and $\bar{y}$ shows the mean value of y for a number n of samples.

$$R^2\ y, \hat{y}\ = \frac{\sum_{i=0}^{n-1}(y_i - \hat{y}_i)^2}{\sum_{i=0}^{n-1}(y_i - \bar{y}_i)^2} \tag{2}$$

The strength of the results were also tested out using the explained variance score, this latter score measures to what proportion a mathematical model responds to changes (dispersion) of a certain dataset [16, 17]. In this score, the closer to 1 the better it is as well. The model had a score of 93.2% (or 0.932).

Fig. 3 shows the relationship between the predicted results and the test dataset. It should be noted that it does not have the usual shape of the knee angles because of the way the data were divided, but it is useful to see the relationships of the results.

According to [18], one of the most studied joints is that of the knee because of its importance in human gait. Positive effects were evidenced by [19], the results found conclude that it is possible to perform analysis of the human gait by the video system, with the possibility of analyzing joints.

Accuracy may have been affected by the small sample size, associated with the heterogeneity of the studied group. These two factors prevented the stratified analysis of some inclusion factors, such as sex, age and pathologies, which could be considered a limitation of this research, but these results, however, are the starting point for future studies involving a larger sample.

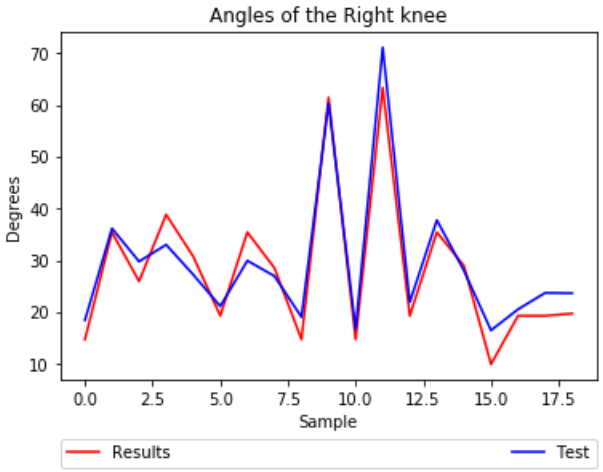


Fig. 3. Similarity of predicted data and actual test data.

4 Conclusion and final thoughts

This paper demonstrated the use of an ML technique (an MLP) to predict the angles of a human knee from the video frames. The results obtained were very satisfactory as an initial experiment, showing that the internal relations of their input and output could be used in the future as a tool to assist professionals who cannot use a high cost system of motion capture.

Future works could include different ML techniques, such as Support Vector Machines (SVM) and deep learning, or even a comparison between the three of them. As it is thought to be an aid to health professionals and physicians, a better user interface must be developed.

In addition, with a more medical-oriented approach, this study would also aid in more accurate diagnoses with a promising prognosis to prevent injury and improve performance.

5 Conflict of interests

The authors declare that there is no conflict of interests with this paper.

6 Acknowledgements

The first author would like to thank the Informatics Laboratory in Health (ILH) from UnB at Gama and the Human Movement Laboratory of UnB at Ceilândia for the resources given and also the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) for the scholarship given to the first author.

References

1. Peña E., Calvo B., Doblaré, M.: Biomecánica de la articulación de la rodilla tras lesiones ligamentosas. Revista Internacional de Métodos Numéricos para Cálculo y Diseño en Ingeniería 22(1), 63-78 (2006).
2. Chan, S.C., Seedhom, B.B.: The effect of the geometry of the tibia on prediction of the cruciate ligament forces: A theoretical analysis. In: Proceedings of the Institution of Mechanical Engineers, Part H: Journal of Engineering in Medicine 209(1), 17–30 (1995).
3. Beynnon, B. et al: A sagittal plane model of the knee and cruciate ligaments with application of a sensitivity analysis, Journal of Biomechanical Engineering 118(2), 227–239 (1996).
4. Bendjaballah, M. Z., Shirazi-adl, A., Zukor, D. J.: Biomechanical response of the passive human knee joint under anterior-posterior Forces. Clinical Biomechanics 13(8), 625–633 (1998).
5. Murphy, K.P.: Machine Learning: A Probabilistic Perspective. The MIT Press, Cambridge, Massachusetts (2012).
6. Raschka, S.: Python Machine Learning. Packt Publishing, Birmingham (2015).
7. Haykin, S.: Neural Networks and Learning Machines. 3rd edition. Pearson Prentice Hall, New York (2008).
8. Lima, R.A. et al: Human Knee Simulation Using CMAC ANN. In: D.A. Jaffray (ed.), World Congress on Medical Physics and Biomedical Engineering (2015).
9. Guimarães, I.J.A. et al: Human knee simulation using multilayer perceptron artificial neural network. In: XXV Congresso Brasileiro de Engenharia Biomédica (2016).
10. Leite, N.M., Faloppa, F.: Propedêutica Ortopédica e Traumatologia. Artmed, Porto Alegre (2013).
11. Perry, J., Burnfield, J.M.: Gait Analysis: Normal and Pathological Function. Slack, Thorofare, N. J. (2010).
12. Aguiar L.R.: Implementando um Software como Serviço para Análise e Simulação de Marcha Humana. MSc dissertation, Universidade de Brasília (UnB), Gama, DF (2015).
13. Larson R., Edwards B.H.: Multivariable Calculus. 9th ed. Brooks/cole, Belmont (2010).
14. Hornik, k.: Approximation Capabilities of Multilayer Feedforward Networks. In: Neural Networks, vol. 4, pp. 251-257. Pergamon Press PLC (1991).
15. Pedregosa et al.: Scikit-learn: Machine Learning in Python. In: JMLR 12, pp. 2825-2830 (2011).
16. Glantz, S.A.; Slinker, B.K.: Primer of Applied Regression and Analysis of Variance. McGraw-Hill (1990).
17. Nagelkerke, N.J.D.: A Note on a General Definition of the Coefficient of Determination. In: Biometrika vol. 78 (3): 691–692 (1991).
18. Souza, F.C.S.: Avaliação de Joelho Mecânico para Amputados. MSc Dissertation, Universidade de Brasília, Brasília, DF (2014).
19. Araújo, A.G.N., Andrade, L.M., Barros, R.M.L.: Sistema para análise cinemática da marcha humana baseado em videogrametria. In: Fisioterapia e Pesquisa vol. 2 pp. 3-10 (2005).

E.2 *HUMAN KNEE SIMULATION USING MULTILAYER PERCEPTRON ARTIFICIAL NEURAL NETWORK*

CAPÍTULO 21

HUMAN KNEE SIMULATION USING MULTILAYER PERCEPTRON ARTIFICIAL NEURAL NETWORK

Ithallo Junior Alves Guimarães
Universidade de Brasília (UnB), Faculdade Gama (FGA)
Brasília – Distrito Federal

Roberto Aguiar Lima
Universidade de Brasília (UnB), Faculdade Gama (FGA)
Brasília – Distrito Federal

Vera Regina Fernandes da Silva Marães
Universidade de Brasília (UnB), Faculdade Gama (FGA)
Universidade de Brasília (UnB), Faculdade Ceilândia (FCE)
Brasília – Distrito Federal

Lourdes Mattos Brasil
Universidade de Brasília (UnB), Faculdade Gama (FGA).
Brasília – Distrito Federal

ABSTRACT: This work intends to show the usage of a Multilayer Perceptron Artificial Neural Network (MLP ANN) to simulate the human knee, more precisely, its angular velocities. The MLP was chosen due to its ability to converge in non-linear problems and, so, form more complex separating boundaries for the problems. This is an open source project and its source code is available at https://github.com/rob-nn/mlp_knee.

KEYWORDS: MLP, ANN, Back-propagation.

1 I INTRODUCTION

Artificial intelligence techniques are very common in modern days, and are even invisible most of time. They are used from websites, prediction making, cellphones up to medical purposes.

This paper is based on a previous paper (LIMA et al., 2015), which discusses about the use of a Cerebellar Model Articulation Control artificial neural network (CMAC ANN) in order to predict and simulate the behavior of the human knee. Here, the use of Multilayer Perceptron Artificial Neural Network (MLP ANN) is shown in order to simulate its angular velocities. As it suggests, the MLP comes from the perceptron, which, according to Begg; Lai; Palaniswami (2008), mimics the basics of mammalian visual system and is an example of the simplest feedforward network, consisting of a single neuron. MLPs consist of at least three different layers of perceptrons, input layer, hidden layer and output layer.

The MLP is used due to the fact that the single perceptron cannot converge for non-linearly separable problems (BEGG; LAI; PALANISWAMI, 2008). It has some features that, according to Haykin (2008) should be highlighted, such as: the existence of one or more hidden layers from both input and output

nodes; the fact that all the neurons contain a non-linear differentiable function and the high degree of connectivity present in this network (“fully connected”, each node in all the layers is connected to all nodes in the next layer (BEGG; LAI; PALANISWAMI, 2008)). The MLP shown in this paper uses the back-propagation algorithm to set the weights of its nodes accordingly to the data fed into the network. This algorithm has its name because the computation of the error goes from the output layer back to the first hidden layer (NG, 2016), the input does not enter in the calculation as it is just the features observed on the training set (NG, 2016).

Thus, it can be seen as a very important machine learning technique which may be used in applications of biomedical engineering, such as developing better and more precise active prostheses that would improve life quality for the ones that depend on them.

2 | MATERIALS AND METHODS

The methods and materials for this paper were based in Lima et al. (2015) and it comes as modified version of it. The Figure 1 was modified from Lima et al. (2015) and shows the steps for the simulation suggested.

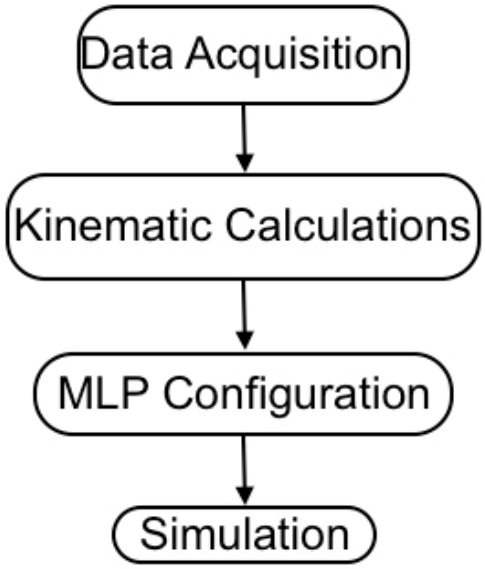


Figure 1: Steps for the simulation process.

Data Acquisition – The data used to train the MLP was obtained from a subject in the Human Performance Laboratory at Faculty UnB-Ceilândia. According to Lima et al. (2015), the data was acquired using twelve cameras (Qualisys Oqus MRI), passive body markers and the Qualisys QTM 3.2 software package. Motion capture techniques were used here to obtain the data. A healthy female subject was selected and she repeated a walk of about five second for five times. The markers used were distributed along 40 positions on the inferior limbs of the subject. The data generated comes

from one of the walks and had its beginning and ending cut in order to constitute a comfortable gait cycle (LIMA et al., 2015).

The data acquisition process was approved by the Health Faculty Ethics Committee from UnB, protocol N11911/12 (LIMA et al., 2015).

Kinematic Calculations – Octave was used on the computation processes. The data from the Qualisys QTM was exported to the MATLAB format, as it is possible (QUALISYS, 2016). MATLAB files can be read by Octave, as they are mostly compatible (GNU OCTAVE, 2016).

Equation (1) shows how to obtain the angles (LARSON and EDWARDS, 2010).

$$\theta = \cos^{-1}\left(\frac{u \,.\, v}{\|u\|\|v\|}\right) \qquad (1)$$

The angles came from malleolus, knee and trochanter from both legs. Then, the knee was set as origin of the system and the relative positions of the other elements were calculated. After this, the angles could be calculated using (1). Calculations were done separately for each limb.

The velocities and angular velocities were calculated, respectively, by the difference of an angle and its predecessor and the position in the space and its predecessor divided by a time *t*. The time *t* is fixed and given by the time between samples. For this case, as the sampling rate was 60 frames per second, so, *t* is about 17 milliseconds.

Angular accelerations were calculated using a similar fashion.

MPL Configuration – Nine input signals were used in order to predict one output signal. The used signals were the angular velocities, angles and angular accelerations from the left knee and the velocities from both knees (composed of three velocities for the planes x, y and z each).

The output signal (predicted one) was the angular velocities from the right knee. The implemented MLP use biases and is composed of two hidden layers of forty elements each, one input layer and one output layer, summing up four layers, these parameters were set based on a trial and error basis. This is a feedforward trained using the back-propagation algorithm. It used the hyperbolical tangent function, shown in (2), as activation function (BRASIL; AZEVEDO; LIMÃO, 2000).

$$\tanh x = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \qquad (2)$$

The learning parameters (learning rate and momentum constant (HAYKIN, S., 2008)) were set using an external code to generate random number in a range between 0 and 1. Then the numbers were passed to the network and tried, the best fitted ones (the ones with the least error) were selected. This process was repeated

for several times and resulted in the following values for learning rate and momentum (respectively) 0.043142 and 0.0033804.

The error function for this problem is given by the cost function. The cost function is said by Ng (2016) to be a generalization of the one used for linear regression.

Simulation Details – In order to implement this simulation, the Octave software was used. Octave is a high-level interpreted language primed for numerical computations and it comes as free software (GNU OCTAVE, 2016).

A MacBook Pro (mid 2014) running OS X El Capitan (version 10.11.5), processor 2.8 GHz Intel Core i5, RAM 8 GB was used to run the code, using Octave 4.0.0.

The data gotten from the subject is depicted below on Figure 2. This figure was generated from the data obtained from the subject by the software developed by Lima (2015). Its parameters and separations were based on the work of Perry and Burnfield (2010).

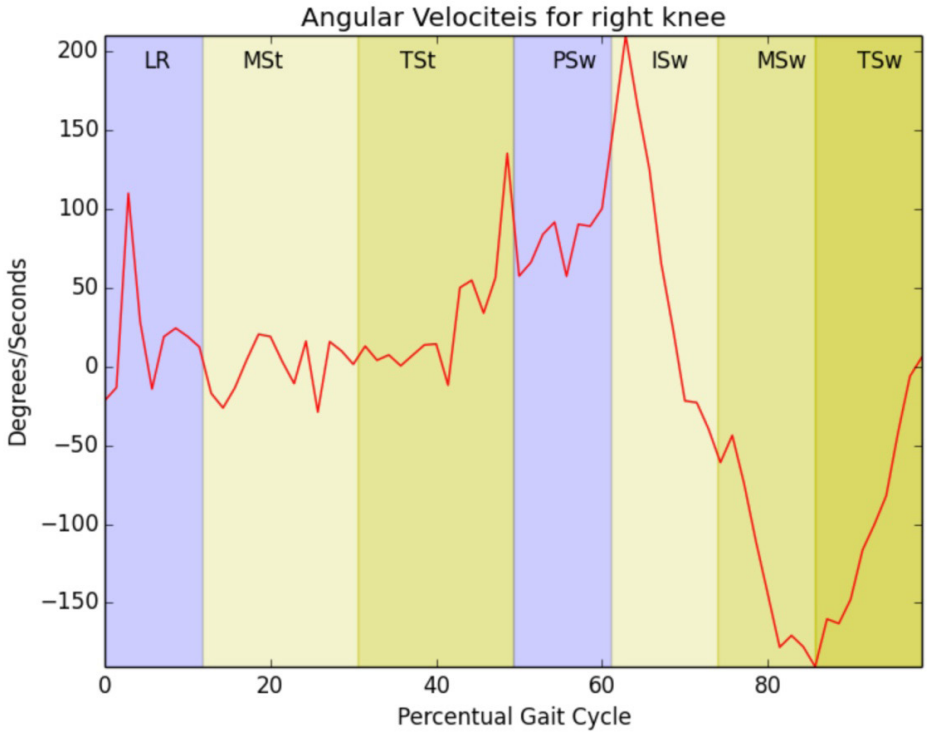


Figure 2: Angular velocities for the right knee of the subject.

Not all data gotten were used to train the network. After filtering the data that would be used as input (nine signals that were mentioned before), a process samples randomly about sixty percent of the original data and passes it to the network for training. After the training, the remaining forty percent is passed to the network to plot the results and it is plotted alongside with that. Therefore, each time the network is trained it uses different data. Results shown here were made using this procedure.

The initial weights were set randomly in a range from -0.2 to 0.2.

The code was set to run for 200 iterations before its results were plotted. Two

plots were generated; one that shows the error decreasing across the iterations and another for results, showing the predicted data (output of the neural network) and the randomly sampled data.

3 I RESULTS

The code developed was used to predict the angular velocities of a knee, based on its velocities and the data from the contralateral knee. The Figure 3 shows how the error decreases across the generations for the MLP. Figure 4 shows the desired output compared to the obtained one. The results were obtained for one patient and came from a small amount of initial data, considering that the original data were separated into two parts, one for training and another to get the results shown, these results can be considered relevant.

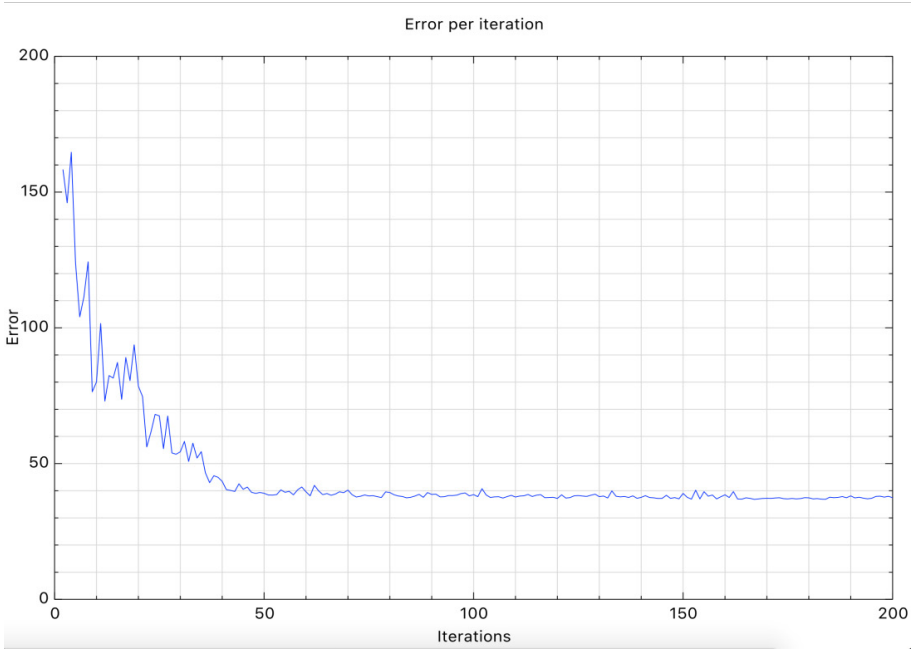


Figure 3: Error decreasing as the number of iterations increase.

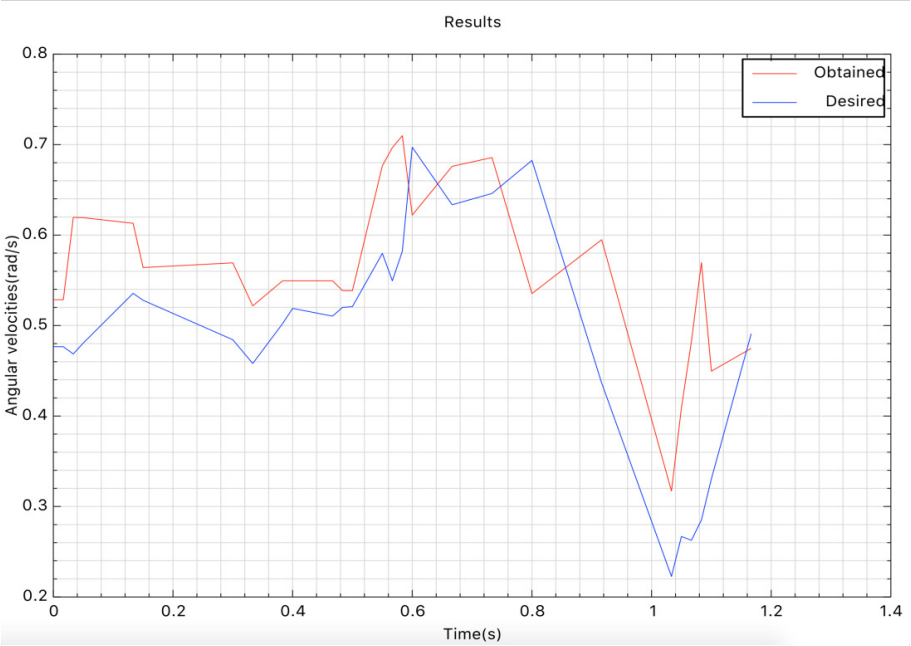


Figure 4: Results from the network.

4 I DISCUSSION

Figure 2, which shows the plot for angular velocities from the right knee of the subject using all the data can be compared to the plot on Figure 4. They show to be, somehow, different, it happens due to the fact that the training set of the network was a sample of the data gotten from the patient, as afore mentioned.

The results could be improved using bigger amounts of data (for instance, a greater sampling rate, a greater number of subjects, etc.) and it should be done for future works. Convergence of the ANN can be seen as it uses different data for training and obtaining results.

This work opens up possibilities for real world applications such as powered prosthesis and software for gait analysis. The technique could be used in order to get similar results as the ones gotten by MIT on its world's first powered ankle-foot prosthesis (MIT MEDIA LABORATORY, 2016).

There is a current project on developing a powered transfemoral prosthesis which the authors take part and it has MLP ANN as one of possible techniques to be used.

In practical perspective of building prosthesis, the usage of the data afore mentioned implies in putting sensors on the knees, which might have a high level of annoyance for the patient. Data from others inputs, such as from electromyography or from pressure sensors for instance, could have been used in order to achieve the obtained results, but they were not available at the time this paper was written.

5 I CONCLUSION

This paper showed the usage of MLP ANN in order to predict the angular velocities of a human knee.The results obtained from the ANN are alike the real ones, although, many differences can been seen.

The software developed while this paper was written is open source and given under the terms of the MIT License (MIT) (OPEN SOURCE INITIATIVE, 2016) to anyone who wishes to use it. The source code is available at https://github.com/rob-nn/mlp_knee.

This work has to continue in order to improve its results, getting closer to the real signal, using less input signals, using another input signals or even developing a real world application such as software for gait analysis or even powered prosthesis with this model of ANN.

6 I ACKNOWLEDGEMENTS

The authors would like to thank LIS from FGA-UnB, the Human Movement Laboratory of FCE-UnB for the resources given and also the Fundação de Apoio à Pesquisa do Distrito Federal (FAPDF), the Coordenadação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES), and the Universidade de Brasília (UnB) for the financial support.

REFERENCES

LIMA, R.A. et al. **Human Knee Simulation Using CMAM ANN**. In: Jaffray D. (eds) World Congress on Medical Physics and Biomedical Engineering, June 7-12, 2015, Toronto, Canada. *IFMBE Proceedings...*, vol 51. Springer, 2015.p.1159-1162.

BEGG, R.; LAI, T.H.D.; PALANISWAMI, M. **Computational Intelligence in Biomedical Engineering**. New York: CRC Press, 2008.

HAYKIN, S. **Neural Networks and Learning Machines**. 3rd ed. New York: Pearson Prentice Hall, 2008.

NG, A. **Machine learning by Stanford University**. Available on: <https://www.coursera.org/learn/machine-learning/home/week/5>. Access on April 2016.

QUALISYS. **Qualisys Track Manager: User friendly mocap software**. Available on: <http://www.qualisys.com/software/qualisys-track-manager/>. Access on April 2016.

GNU OCTAVE. **About GNU Octave**. Available on: <https://www.gnu.org/software/octave/about.html>. Access on April 2016.

LARSON, R.; EDWARDS, B.H. **Multivariable Calculus**. 9th ed. Belmont: Brooks/cole, 2010.

BRASIL, L.M.; AZEVEDO, F.M. de ; LIMÃO, R.C.O. **Redes Neurais com Aplicações em Controle e em Sistemas Especialistas**. 1. ed. Florianópolis - SC: Bookstore Livraria Ltda., 2000.

LIMA, R.A. ***Implementando Um Software Como Serviço Para Análise E Simulação De Marcha Humana.*** 2015. 101p. MSc dissertation - University of Brasilia (UnB), Gama, 2015.

PERRY J.; BURNFIELD J.M. **Gait Analysis: Normal and Pathological Function**. 2nd ed. Thorofare: SLACK Incorporated, 2010.

OPEN SOURCE INITIATIVE. **The MIT License (MIT)**. Available on: <https://opensource.org/licenses/MIT>. Access on April 2016.

MIT MEDIA LABORATORY. **World's First Powered Ankle-Foot Prosthesis Developed by the MIT Media Lab**. Available on: <http://www.media.mit.edu/press/ankle/anklefoot-bg.pdf>. Access on July 2016.