UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

JOÃO PEDRO MORETO LOURENÇÃO

SISTEMA DE AQUISIÇÃO E PROCESSAMENTO DE SINAIS DE ELETROMIOGRAFIA DE SUPERFÍCIE EM TEMPO REAL APLICADO EM INTERFACE MES-FES

APUCARANA

2023

JOÃO PEDRO MORETO LOURENÇÃO

SISTEMA DE AQUISIÇÃO E PROCESSAMENTO DE SINAIS DE ELETROMIOGRAFIA DE SUPERFÍCIE EM TEMPO REAL APLICADO EM INTERFACE MES-FES

Real-Time Surface Electromyography Signal Acquisition And Processing System Applied In MES-FES Interface

Trabalho de Conclusão de Curso de Graduação apresentado como requisito para obtenção do título de Bacharel em Engenharia de Computação do Curso de Bacharelado em Engenharia de Computação da Universidade Tecnológica Federal do Paraná.

Orientador: Prof. Dr. Daniel Prado de Campos

APUCARANA

2023

JOÃO PEDRO MORETO LOURENÇÃO

SISTEMA DE AQUISIÇÃO E PROCESSAMENTO DE SINAIS DE ELETROMIOGRAFIA DE SUPERFÍCIE EM TEMPO REAL APLICADO EM INTERFACE MES-FES

Trabalho de Conclusão de Curso de Graduação apresentado como requisito para obtenção do título de Bacharel em Engenharia de Computação do Curso de Bacharelado em Engenharia de Computação da Universidade Tecnológica Federal do Paraná.

Data de aprovação: 22/novembro/2023

Prof. Dr. Fábio Irigon Pereira
Universidade Tecnológica Federal do Paraná

Prof. Dr. Rafael Gomes Mantovani
Universidade Tecnológica Federal do Paraná

Prof. Dr. Daniel Prado de Campos
Universidade Tecnológica Federal do Paraná

APUCARANA
2023

Dedico este trabalho à minha família e minha namorada, pelo apoio incondicional.

AGRADECIMENTOS

Certamente estes parágrafos não irão atender a todas as pessoas que fizeram parte dessa importante fase de minha vida. Portanto, desde já peço desculpas àquelas que não estão presentes entre essas palavras, mas elas podem estar certas que fazem parte do meu pensamento e de minha gratidão.

Agradeço ao meu orientador Prof. Dr. Daniel Prado de Campos, pela sabedoria com que me guiou nesta trajetória.

À minha família e a minha namorada, pois acredito que sem o apoio deles seria muito difícil vencer esse desafio.

À Deus.

Aos meus colegas.

À Coordenação do Curso, pela cooperação.

Gostaria de deixar registrado também, o meu reconhecimento aos professores André Luis Tinassi Damato e Maurício Eiji Nakai pelo apoio e aos professores Fábio Irigon Pereira e Rafael Gomes Mantovani por terem aceitado participar como membros avaliadores.

Enfim, a todos os que por algum motivo contribuíram para a realização desta pesquisa.

"Não importa o quanto você bate, mas sim o quanto aguenta apanhar e continuar. O quanto pode suportar e seguir em frente. É assim que se ganha."
Rocky Balboa, do filme Rocky Balboa (2006)

RESUMO

O trabalho foi idealizado para oferecer uma opção de menor custo e confiável para a aquisição de sinais de eletromiografia de superfície (sEMG). Inicialmente, foi desenvolvido sistema de aquisição, onde, utilizando um módulo *Bluetooth* ESP32 foi implementada a aquisição dos biossionais com o auxílio do sensor de biopotenciais baseado em AD8232. O *backend* foi desenvolvido na linguagem *Python* e foi utilizado o *framework Flask* para a aplicação *web*. O *frontend* contempla duas páginas. A página principal é a responsável por exibir o valor atual de cada parâmetro, permitindo assim que o usuário tenha controle sobre os parâmetros mais relevantes para a ativação da estimulação elétrica funcional (do inglês *functional electrical stimulation*) (FES). A página secundária possibilita a visualização em tempo real dos dados em função do tempo. A ferramenta foi desenvolvida com quase todas as funcionalidades inicialmente previstas. Entretanto, isso não afetou o resultado final da mesma, que vai possibilitar aos profissionais da área e pesquisadores a obter os dados de sEMG com taxa de amostragem constante, visualização dos dados ao vivo e ainda manipular como desejarem os dados adquiridos com o arquivo que pode ser exportado.

Palavras-chave: eletromiografia; processamento eletrônico de dados em tempo real; aplicações web; sistemas embarcados (computadores).

ABSTRACT

The work was designed to offer a lower-cost and reliable option for acquiring surface electromyography (sEMG) signals. Initially, an acquisition system was developed, where, using an ESP32 Bluetooth module, the acquisition of biosionals was implemented with the help of the biopotential sensor based on AD8232. The backend was developed in the Python language and the Flask framework was used for the web application. The frontend includes two pages. The main page is responsible for displaying the current value of each parameter, thus allowing the user to have control over the most relevant parameters for activating functional electrical stimulation (FES). The secondary page allows real-time visualization of data over time. The tool was developed with almost all the functionalities initially planned. However, this did not affect the final result, which will enable professionals in the field and researchers to obtain sEMG data with a constant sampling rate, view the data live and even manipulate the data acquired with the file as desired. be exported.

Keywords: electromyography; real-time electronic data processing; web applications; embedded systems (computers).

LISTA DE FIGURAS

LISTAGEM DE CÓDIGOS FONTE

LISTA DE ABREVIATURAS E SIGLAS

Abreviaturas

AJAX	*JavaScript* assíncrono + XML
AVC	acidente vascular cerebral
CPU	Unidade Central de Processamento
CSV	*comma-separated values*
EMG	eletromiografia
FES	estimulação elétrica funcional (do inglês *functional electrical stimulation*)
FFT	transformada rápida de Fourier
IHC	Interação Humano-Computador
JSON	*JavaScript Object Notation*
LBI	Lei Brasileira de Inclusão da Pessoa com Deficiência
MES	sinais mioelétricos - do inglês *myoelectrical signals*
MES-FES	estimulação muscular elétrica-funcional elétrica
RMS	valor eficar - do inglês *Root-mean-square*
sEMG	eletromiografia de superfície
TA	tecnologia assistiva
TICs	Tecnologias de Informação e Comunicação

SUMÁRIO

1 INTRODUÇÃO

1.1 Considerações iniciais

A sociedade está imersa em um cenário de constante transformação e evolução, impulsionado principalmente pelos avanços tecnológicos e pela globalização. A tecnologia tem como missão facilitar a vida de todas as pessoas, inclusive aquelas com deficiência. Para atender a esse propósito, foi desenvolvido o campo da tecnologia assistiva (TA) (GARCIA, 2018), que engloba uma diversidade de recursos, ferramentas, processos, práticas, serviços, metodologias e estratégias, com o objetivo primordial de proporcionar maior autonomia, independência e qualidade de vida aos seus usuários.

No Brasil, a Lei nº 13.146/2015, também conhecida como Lei Brasileira de Inclusão da Pessoa com Deficiência (LBI), destaca o conceito de tecnologia assistiva ou ajuda técnica como sendo produtos, equipamentos, dispositivos, recursos, metodologias, estratégias e serviços que visam promover a funcionalidade, relacionada à atividade e à participação da pessoa com deficiência ou mobilidade reduzida, almejando sua autonomia, independência, qualidade de vida e inclusão social (Instituto Federal do Rio Grande do Sul, 2021).

Dentro do estudo da Interação Humano-Computador (IHC) (GUPTA *et al.*, 2023), pensar em interfaces é considerar a conexão entre máquinas e usuários, sendo responsáveis por traduzir linguagens computacionais para facilitar a interação por meio das Tecnologias de Informação e Comunicação (TICs). As interfaces, como janelas, ícones e menus, baseiam-se principalmente na visão, toque e movimento de cursor, excluindo usuários com deficiência que necessitam de dispositivos auxiliares para interagir. Cerca de 24% da população brasileira possui alguma deficiência (IBGE, 2010), demonstrando a necessidade de pesquisa na área da TA (COSTA ANDRÉIA SIAS RODRIGUES, 2016).

Nesse contexto, a busca pelo conhecimento e pela compreensão dos fenômenos que permeiam a realidade torna-se fundamental. Diante desse panorama, este trabalho de conclusão de curso tem como objetivo analisar e investigar a temática central, proporcionando uma contribuição significativa para o estudo da sEMG. O trabalho a ser desenvolvido foi idealizado para ser uma opção de menor custo e confiável para a aquisição de sinais de sEMG.

1.2 Objetivos

1.2.1 Objetivo geral

Desenvolver um sistema de aquisição de sinais de eletromiografia de superfície (sEMG) em tempo real, com taxa de amostragem constante e pós-processamento em *Python*, com a finalidade de aplicação em uma interface do sistema nervoso periférico.

1.2.2 Objetivos específicos

- Realizar uma revisão de literatura sobre o processamento de eletromiografia (EMG), incluindo técnicas de filtragem, extração de características e classificação.
- Implementação do envio dos dados via *Bluetooth*.
- Desenvolvimento do cálculo do valor eficar - do inglês *Root-mean-square* (RMS) no microcontrolador.
- Processar os dados adquiridos através do Python, mostrá-los em tempo real e permitir a atualização dos parâmetros da FES via *framework Flask*.
- Desenvolver a interface *Web* utilizando HTML, CSS e *JavaScript*.

1.3 Justificativa

A obtenção de dados de sEMG de maneira precisa é um tema relevante e atual que desperta grande interesse no âmbito acadêmico e profissional. A pesquisa tem como principal propósito aprofundar o entendimento sobre o assunto, analisando seus aspectos teóricos e práticos, bem como identificar possíveis lacunas de conhecimento que podem ser preenchidas através desta pesquisa.

A escolha deste tema se deve à sua importância tanto no contexto acadêmico quanto no contexto profissional. Além disso, a crescente demanda por estudos e pesquisas que abordem essa temática reforça a necessidade de uma investigação aprofundada, que possa contribuir para o desenvolvimento do conhecimento na área, bem como para a prática e aplicação dos resultados obtidos.

Espera-se que este trabalho possa contribuir de forma significativa para a área de estudo, proporcionando *insights* relevantes, ampliando o conhecimento e estimulando o aprimoramento de práticas e abordagens relacionadas ao tema. Por fim, é importante ressaltar que este trabalho de conclusão de curso não pretende esgotar todas as possíveis abordagens relacionadas ao tema, mas sim contribuir para o avanço do conhecimento e estimular a reflexão e o debate em torno do assunto. Acredita-se que os resultados obtidos por meio deste estudo possam ser utilizados como subsídios para futuras pesquisas e para aprimorar a compreensão sobre a obtenção de dados de sEMG.

1.4 Estrutura do trabalho

O Capítulo 2 traz uma contextualização da teoria utilizada neste trabalho, permitindo o entendimento dos conceitos necessários.

O Capítulo 3 faz relação com outros trabalhos na mesma área e apresenta os pontos de interseção deles com este trabalho.

O Capítulo 4 traz a descrição de como e quais materiais foram utilizados, salientando como foram utilizados. Ele apresenta também as equações necessárias e traz os fluxogramas de funcionamento geral do módulo ESP32 e da aplicação *Web*. São apresentados também o diagrama de blocos do sistema e o diagrama de *hardware*.

O Capítulo 5 apresenta os resultados obtidos por este trabalho, trazendo figuras e explicações de códigos fonte. Também traz a apresentação do hardware montado, as telas da aplicação e o teste de verificação da taxa de amostragem para confirmar a se a mesma está constante. Por fim, apresenta discussões sobre o sistema desenvolvido, incluindo as limitações encontradas.

O Capítulo 6 apresenta uma retomada do que foi visto neste trabalho, apresentando pontos de melhoria, vantagens, desvantagens e limitações do que foi desenvolvido. Por fim, são apresentadas as conclusões.

2 REFERENCIAL TEÓRICO

No campo da tecnologia assistiva (TA), o termo interface refere-se aos pontos de interação e comunicação entre o usuário e os dispositivos ou sistemas assistivos. Essas interfaces são projetadas para facilitar a comunicação e o controle, permitindo que as pessoas com deficiências ou limitações funcionais utilizem os recursos tecnológicos de maneira acessível e eficiente. As interfaces na tecnologia assistiva podem assumir diferentes formas, dependendo das necessidades específicas dos usuários e do tipo de tecnologia envolvida. Algumas das interfaces mais comuns incluem:

- **Interfaces táteis**: São aquelas que envolvem o toque ou o contato físico para interagir com dispositivos ou sistemas assistivos. Por exemplo: botões, teclas, superfícies sensíveis ao toque ou painéis de controle físicos.
- **Interfaces visuais**: Envolve a apresentação de informações visuais que permitem aos usuários compreender e interagir com o dispositivo. Isso pode incluir ícones, menus, telas de toque, interfaces gráficas e outros elementos visuais.
- **Interfaces auditivas**: São as que fornecem informações e *feedback* em formato sonoro ou de áudio. Isso pode ser útil para pessoas com deficiência visual, por exemplo, que dependem de *feedback* auditivo para interagir com um dispositivo.
- **Interfaces de comunicação**: São as que possibilitam a troca de informações entre o usuário e outras pessoas, como sistemas de comunicação alternativa e aumentativa, que permitem que indivíduos com dificuldades de fala se expressem usando símbolos, imagens ou voz sintetizada.
- **Interfaces gestuais ou de movimento**: Permitem que o usuário controle dispositivos ou sistemas assistivos por meio de gestos, movimentos corporais ou comandos de voz.
- **Interfaces adaptáveis**: São interfaces que podem ser personalizadas e ajustadas conforme as preferências e necessidades do usuário, permitindo maior acessibilidade e usabilidade.

Essas interfaces têm um papel fundamental na tecnologia assistiva, pois são o elo entre o usuário e a tecnologia, tornando possível a utilização efetiva das soluções assistivas e contribuindo para a inclusão e a autonomia das pessoas com deficiências ou limitações funcionais, proporcionando-as, consequentemente, uma melhor qualidade de vida.

Dentro do contexto das interfaces, existem duas grandes áreas: órteses e próteses. Órtese é uma peça ou aparelho de correção ou complementação de membros ou órgãos do corpo. Também definida como qualquer material permanente ou transitório que auxilie as funções de um membro, órgão ou tecido, sendo não ligados ao ato cirúrgico os materiais cuja colocação

ou remoção não requeiram a realização de ato cirúrgico [1]. Prótese é uma peça ou aparelho de substituição dos membros ou órgãos do corpo. Compreende qualquer material permanente ou transitório que substitua total ou parcialmente um membro, órgão ou tecido [2] (Departamento de Atenção Especializada e Temática, 2016). Um exemplo simples para diferenciar órtese de prótese que podemos tomar é a comparação de muletas com as próteses de pernas. Enquanto as muletas têm como objetivo auxiliar a locomoção da pessoa sem substituir as pernas da mesma, as próteses buscam substituí-las, de modo que a pessoa tenha o membro restaurado.

As neuropróteses representam dispositivos desenvolvidos para restaurar funções corporais afetadas, sendo únicas por serem controladas diretamente pelo cérebro do utilizador. Por meio de uma rede de eletrodos, elas monitoram a atividade cerebral, reconhecendo padrões que correspondem a movimentos específicos, como o movimento de um braço. Ao pensar em realizar um movimento, a prótese recebe um sinal correspondente, permitindo que ela se mova de acordo com a intenção do indivíduo (CORREIA, 2017).

Neuropróteses motoras envolvem a ativação elétrica de músculos paralisados para produzir movimentos funcionais sob o controle do usuário. Sensores são usados para determinar a intenção do usuário (controle de comando) e também para controle de *feedback*. As neuropróteses motoras usam a ativação elétrica dos nervos motores para produzir movimento. Ao contrário dos sistemas de estimulação utilizados para exercício ou neuromodulação, as neuropróteses motoras incorporam entradas de controle voluntário do usuário em tempo real. Como mostrado na Figura 1, uma neuroprótese motora típica consiste em uma entrada de comando, um algoritmo de controle, saída de estimulação elétrica e, em alguns casos, componentes de *feedback*. O controle em tempo real, ou entrada de comando, permite que o usuário direcione os movimentos e as forças geradas pela estimulação para realizar movimentos intencionais e realizar atividades (KILGORE, 2013).

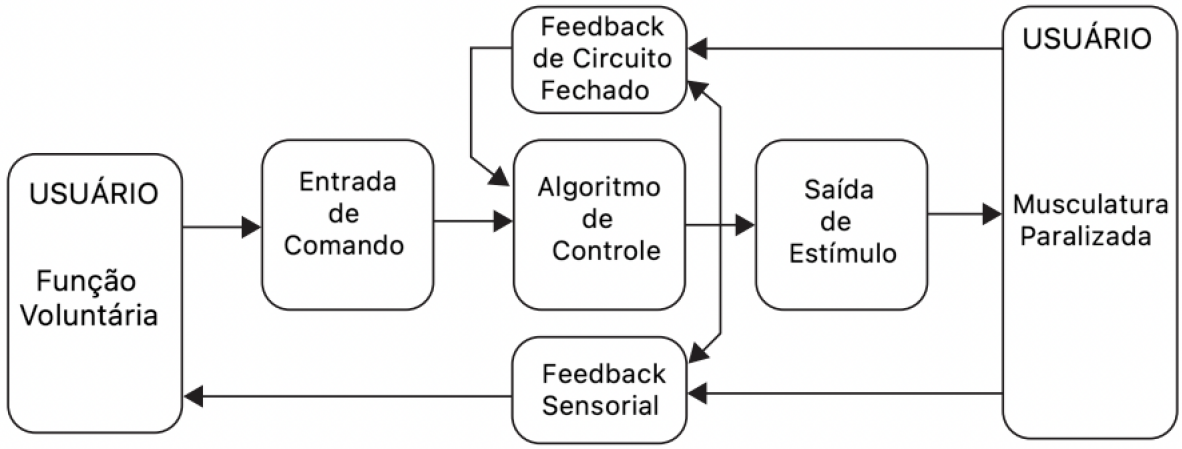
Embora as neuropróteses não restaurem um movimento completamente "normal", foi demonstrado que elas restauram a função a um nível que não pode ser alcançado por nenhum outro meio. Elas têm sido amplamente utilizadas para proporcionar função a indivíduos com lesões na medula espinhal, mas também foram usadas para proporcionar função em casos de acidente vascular cerebral e esclerose múltipla. No entanto, está se tornando cada vez mais comum aplicar estimulação elétrica em casos de acidente vascular cerebral como uma ajuda temporária na reabilitação para auxiliar na aprendizagem motora e recuperação funcional (KILGORE, 2013).

---

[1] (Resolução Normativa da ANS – RN nº 338, de 21 de outubro de 2013, publicada na seção 1, do DOU de 22 de outubro de 2013)

[2] (Resolução Normativa da ANS – RN nº 338, de 21 de outubro de 2013, publicada na seção 1, do DOU de 22 de outubro de 2013)

Figura 1 – Componentes gerais de um sistema neuroprótese motor



Fonte: Adaptado de Kilgore (2013).

2.1 Sinais Mioelétricos (MES)

As raízes da palavras mioelétrico envolvem duas palavras, *myo* do grego *mys*, que significa músculo, e elétrico referente à eletricidade. A partir disso, podemos definir um sinais mioelétricos - do inglês *myoelectrical signals* como a atividade elétrica produzida por um músculo contraído. Quando ocorre a contração muscular, correntes iônicas são geradas profundamente na estrutura muscular, e essas podem ser detectadas por meio de eletrodos. A amplitude e o aspecto desse sinal dependem de várias variáveis, como a profundidade do músculo, o tamanho muscular, a força da contração, o tecido sobrejacente, o tipo de eletrodo usado para detecção, bem como sua localização e orientação (LOVELY, 2004).

Os sinais mioelétricos - do inglês *myoelectrical signals* (MES) podem ser detectados abaixo da superfície da pele usando eletrodos de agulha. Como sua origem está profundamente dentro da estrutura do músculo, aproximar-se da fonte aumentará a magnitude do sinal. Entretanto, esta é uma técnica invasiva e não será utilizada neste trabalho. Ao invés disso, será utilizado o conceito de eletromiografia de superfície (sEMG). A sEMG diz respeito a sinais eletrofisiológicos formados pelos potenciais de ação das unidades motoras, os quais se propagam ao longo dos músculos esqueléticos. Esses sinais podem ser prontamente captados na superfície da pele que recobre os músculos, oferecendo detalhes acerca dos movimentos específicos executados por esses músculos.

2.2 Estimulação Elétrica Funcional (FES)

As técnicas e métodos empregados na estimulação elétrica variam consideravelmente, dependendo da aplicação específica e do problema a ser abordado. No entanto, todos os diferentes tipos envolvem o uso de um dispositivo para gerar pulsos elétricos, conhecido como estimulador elétrico, e uma forma de aplicação no corpo, como o uso de eletrodos colocados na

superfície da pele do usuário. Quando a estimulação elétrica é aplicada para realizar ou complementar tarefas funcionais em pacientes com músculos enfraquecidos ou paralisados, podemos categorizá-la como estimulação elétrica funcional (do inglês *functional electrical stimulation*) (ZHANG *et al.*, 2011).

A FES tem sido amplamente reconhecida recentemente como uma abordagem de tratamento não invasiva. Isso se deve às possíveis vantagens no controle de diversas condições dolorosas (BANG *et al.*, 2023). A FES é uma técnica de neuromodulação, que pode restaurar o movimento ativando os músculos do usuário por meio de pulsos elétricos curtos no sistema nervoso periférico, gerando contrações musculares (LYNCH; POPOVIC, 2008).

2.3 Interface MES-FES

A partir disso, utilizaremos os MES, com a técnica de sEMG, e a FES (FIORIN *et al.*, 2023). A FES será verificada dentro do código do microcontrolador e caso atinja a condição específica, será ativada, gerando a estimulação elétrica no membro do participante.

Diversos esforços estão sendo direcionados para aprimorar a usabilidade das interfaces MES-FES. Os projetos desses sistemas geralmente priorizam a facilidade de uso e a conveniência, frequentemente incorporando elementos como estrutura vestível, baixo peso, conectividade sem fio e ativação voluntária. Tais características têm o potencial de incentivar os usuários a se envolverem com essa tecnologia. Diversos estudos têm proposto interfaces FES que utilizam sinais eletromiográficos, especialmente para pacientes em reabilitação com distúrbios neurológicos (MARQUEZ-CHIN; POPOVIC, 2020). No entanto, existem desafios significativos a serem superados para transformar esses dispositivos em tecnologia de assistência vestível verdadeiramente adequada para uso cotidiano. Além disso, a disponibilização do projeto em código aberto é um componente essencial na disseminação dessa tecnologia, permitindo que outros possam replicar o desenvolvimento e, assim, facilitar uma adoção mais ampla entre os interessados.

Os dispositivos vestíveis FES têm um potencial promissor para auxiliar nas atividades diárias e na terapia manual, mas necessitam de aprimoramentos tecnológicos, como melhor portabilidade e facilidade de uso, antes de serem amplamente adotados em contextos clínicos ou domésticos. Desafios incluem o tempo despendido na calibração e na colocação/remoção desses dispositivos. Mesmo quando originalmente projetados para fins de reabilitação, há uma crescente preocupação sobre a compatibilidade desses dispositivos com *smartphones* e a sua usabilidade. Em particular, há apreensões quanto à capacidade desses dispositivos se comunicarem com *smartphones* e à sua adequação como tecnologia vestível (WANG *et al.*, 2017).

3 TRABALHOS RELACIONADOS

Este trabalho se mostrou relevante para complementar estudos como Efetividade da interface MESS-FESS na dorsiflexão de tornozelo após AVC por cavernoma: estudo de caso (PEREIRA *et al.*, 2022) e Reconhecimento dinâmico contínuo de gestos usando sinais EMG de superfície baseados em internet de coisas médicas habilitada para blockchain (LI *et al.*, 2023).

O primeiro trabalho traz um estudo de caso sobre a efetividade da interface MES-FES após acidente vascular cerebral (AVC) cavernoma, onde as necessidades do trabalho, com relação ao tipo de filtro utilizado, faixa de EMG utilizada e a utilização da FES seriam fornecidas de maneira precisa, além de permitir a configuração dos parâmetros facilitada, visualização ao vivo do sinal sEMG, possibilidade de parar o sistema a fim de evitar problemas com a participante e possibilitar a pós análise dos dados com o arquivo *comma-separated values* (CSV) que poderia ser gerado.

Já o segundo trabalho, concentra-se no reconhecimento de gestos dinâmicos através de sinais sEMG. Então, o presente trabalho poderia auxiliar tanto na obtenção dos sinais brutos, a partir do arquivo CSV que é gerado, quanto no acompanhamento do comportamento do sinal sEMG durante a realização dos testes.

4 MATERIAIS E MÉTODOS

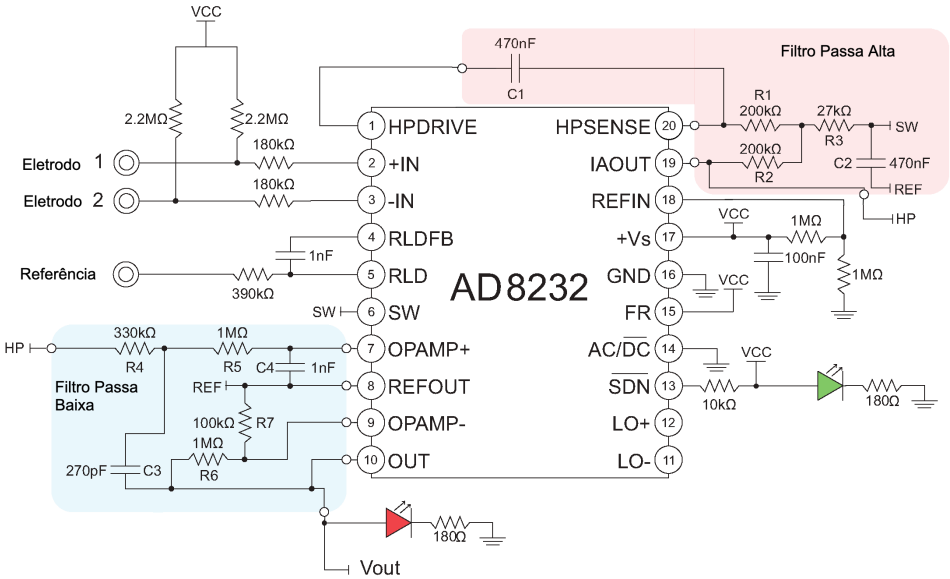
Neste capítulo primeiro são apresentados os materiais e tecnologias utilizados para o desenvolvimento do trabalho, bem como suas versões. Os detalhes de como esses materiais e tecnologias foram utilizados são mostrados na sequência, detalhando as integrações entre *hardware* e *software* presentes no sistema desenvolvido.

4.1 Materiais

Neste trabalho, foi utilizado um Módulo WiFi ESP32 *Bluetooth*, com Unidade Central de Processamento (CPU) *dual-core* de 32 bits e *clock* máximo de 240 MHz, que é o responsável por adquirir e enviar os sinais. Além dele, é utilizado um circuito de aquisição de biopotenciais baseado em AD8232 (JÚNIOR *et al.*, 2023), que, a partir de eletrodos diferenciais conectados à pele, captura os sinais e os envia ao microcontrolador. O circuito mostrado na Figura 2 é o esquemático do sensor de aquisição de biopotenciais utilizado e a Figura 3 mostra a placa de circuito impresso, a partir do esquemático, do sensor de aquisição de biopotenciais.

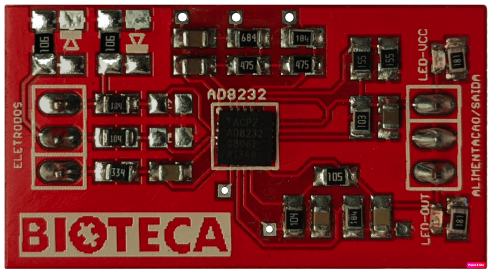
O *backend*, que é a parte não visível do sistema de software, responsável pelo processamento de dados, foi desenvolvido utilizando a linguagem *Python*, na versão 3.11.5 e, utilizando o *framework Flask*, na versão 3.0.0 (GRINBERG, 2018) o usuário poderá utilizar as funcionalidades de uma aplicação *Web* desenvolvida com HTML, CSS e *JavaScript*.

Figura 2 – Circuito de aquisição de biopotenciais baseado em AD3232



Fonte: Júnior *et al.* (2023).

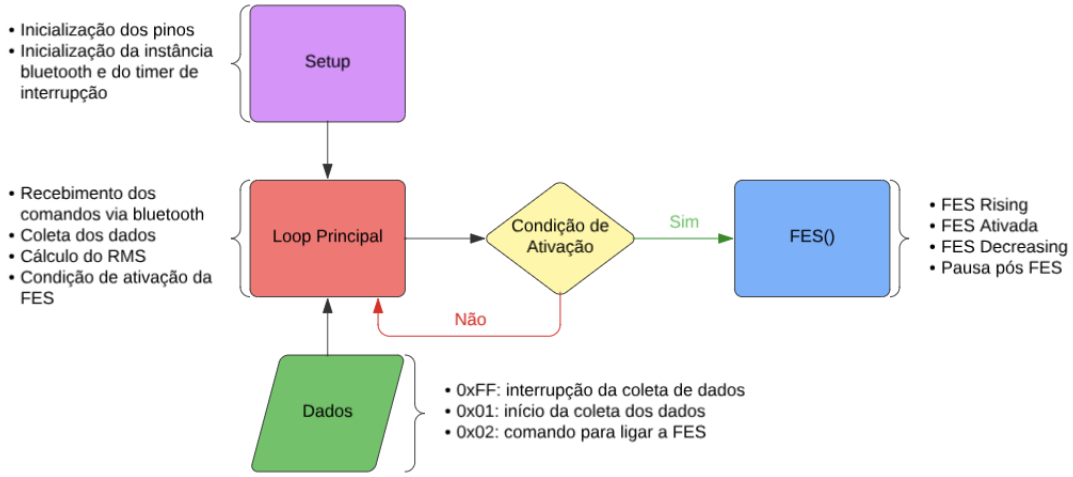
Figura 3 – Placa de circuito impresso baseada em AD8232



Fonte: Autoria própria (2023).

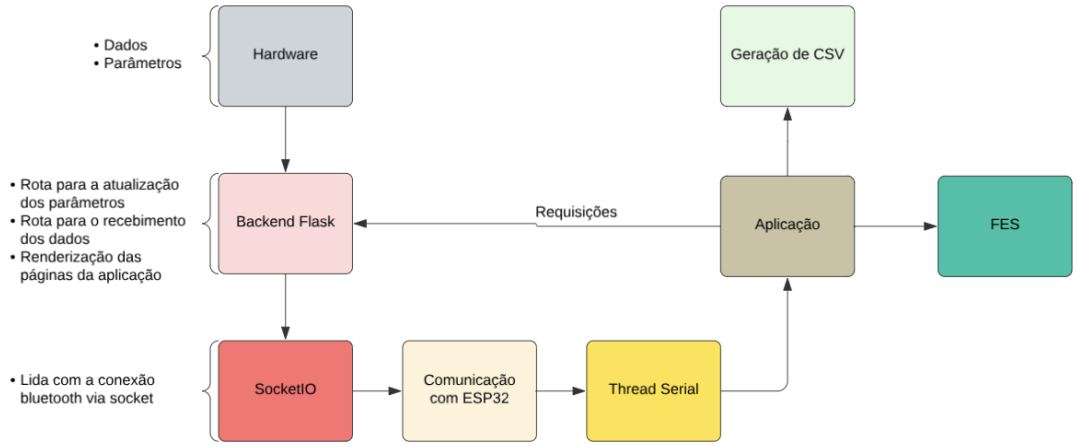
4.2 Métodos

Figura 4 – Fluxograma de funcionamento geral do módulo ESP32



Fonte: Autoria própria (2023).

Figura 5 – Fluxograma de funcionamento geral da aplicação *Web*



Fonte: Autoria própria (2023).

O primeiro passo do trabalho foi implementar o *firmware* do *hardware* para que ele adquira os sEMG com taxa de amostragem constante de 1 kHz, seguindo o funcionamento da Figura 4. Para tal, foram utilizadas interrupções do *timer*. A interrupção do *timer* dispara quando o contador terminar sua contagem, com isso teremos uma interrupção que é executada com um intervalo de tempo definido, que nesse caso será de 1 ms (milissegundo). Após isso, utilizando a linguagem *Python* e as bibliotecas externas *pyserial*, na versão 3.5, (FOUNDATION, 2020) e *scipy*, na versão 1.11.3 (VIRTANEN *et al.*, 2020), as bibliotecas internas *threading, io, csv* e *os*, além das bibliotecas do *Flask render_template, jsonify, request, make_response* e *flask_socketio*, na versão 5.3.6, foram implementadas as funcionalidades necessárias.

A primeira funcionalidade é a leitura dos dados via *Bluetooth*, que são enviados após a coleta com o circuito de aquisição de biossinais. Entre a leitura de um dado e outro, o *script* fará a conversão dos dados, que chegam como 2 bytes, e são convertidos para um valor inteiro. Na sequência, é aplicado um filtro do tipo rejeita-faixa (*notch*) em 60 Hz, a fim de eliminar a frequência da rede de alimentação. Na sequência é aplicado um filtro *Butterworth* de banda passante [1] de terceira ordem com frequência de corte inferior igual a 10 Hz e frequência de corte superior igual a 450 Hz, pois dentro dessa faixa estão concentrados os sinais necessários.

Com isso, é possível fazer a conversão dos dados para ficarem na faixa de 0 a 3,3 V, através da equação 1:

$$dado = \frac{dado_{filtrado} \cdot 3.3}{4095}. \tag{1}$$

Para mostrar os dados no gráfico que é atualizado ao vivo, são utilizadas as listas mencionadas acima. Também é calculada a raiz quadrada da média, RMS, dado pela equação 2:

$$x_{rms} = \sqrt{\frac{1}{N}\sum_{i=1}^{N} x_i^2}, \tag{2}$$

onde, a cada 100 ms, ou seja, a cada 100 amostras, para que a condição de ativação da FES seja calculada, através da equação 3:

$$MES_i > MES_{th} = (1 + th) \times MES_{i-1}, \tag{3}$$

sendo MES_i e MES_{i-1} a raiz quadrada da média a cada 100 amostras, MES_{th} o valor da raiz quadrada da média que deve ser superado e th o valor do *threshold* escolhido. Assim, toda vez que $MES_i > MES_{th}$, ocorre a estimulação elétrica.

Os parâmetros que o sistema possui são:

- *Threshold*: é um parâmetro que determina quanto o valor da MES_i deve ser maior que o valor da MES_{i-1}, em porcentagem, para que ocorra a estimulação elétrica;

[1] filtro eletrônico que retorna uma resposta em frequência o mais plana o quanto for matematicamente possível na banda passante

- Intensidade: é a largura de pulso que define intensidade da estimulação elétrica;
- Tempo de ativação: é o tempo que a estimulação elétrica permanece ligada;
- Tempo pós FES: é o tempo necessário após receber uma estimulação elétrica, a fim de evitar a fadiga muscular.

Utilizando o *framework Flask*, é lançado o *frontend* - que é a parte visível de um sistema de software, representando a interface com a qual os usuários interagem diretamente - da aplicação, seguindo o fluxograma da Figura 5, onde o usuário tem controle sobre o *hardware*, podendo parar a coleta de dados, iniciar a coleta de dados, visualizar o gráfico em tempo real e atualizar os parâmetros *threshold*, tempo de ativação, intensidade e tempo pós FES. Um *script* em *JavaScript* é utilizado para verificar a conexão do *socket Bluetooth*, para atualizar os parâmetros em tempo real com *JavaScript* assíncrono + XML (AJAX), para enviar os novos parâmetros ao servidor *Flask*, para definir um botão nomeado 'Parar', que envia um valor hexadecimal ao microcontrolador para interromper a coleta, um outro botão nomeado 'Iniciar', que envia um outro valor hexadecimal para retomar a coleta e, por fim, um botão 'Ativar FES', para ativar a FES a qualquer momento.

Para a ativação da FES, sem pressionar o botão, deverá ser atingido o critério determinado pela equação 3. O cálculo dos valores RMS necessários para esta condição é feito no microcontrolador, a partir de duas listas. A cada duzentas amostras, que são armazenadas numa lista denominada amostras, ocorrerá a divisão da mesma em duas outras listas de metade do tamanho da lista original, para que possa ser calculado o RMS de cada uma, originando assim os valores de MES_i e MES_{i-1}.

As solicitações que a página faz seguem os modelos da *API RESTful*, com solicitações *GET* e *POST*. O método que utilizar a solicitação do tipo *POST* é o *update_parameters()*, o qual atualiza os quatro parâmetros no servidor e os envia ao ESP32 via *socket Bluetooth*, com a biblioteca *SocketIO* do *Flask*. Os métodos que utilizam solicitações do tipo *GET* são *get_data()* e *FES()*, que retornaram à página *Web* os dados recebidos e o estado da FES, respectivamente. Além disso, utilizando a função *render_template()* do *Flask*, são renderizados os *templates* HTML para o *frontend*.

Por fim, com todas essas funcionalidades já implementadas no servidor *Flask*, o último passo foi adicionar um método e uma rota necessários para que a pessoa possa realizar o download de um arquivo *csv (comma-separated values)* que contenha os valores do dado bruto e o tempo em que foram adquiridos.

4.3 Escopo do sistema

O sistema MES-FES deve adquirir dados de eletromiografia de superfície, através do circuito de aquisição de biopotenciais baseado em AD8232 e de um módulo ESP32 *Bluetooth*,

enviá-los até um *backend* que irá processar esses dados, ou seja, irá convertê-los de binário para inteiro, colocá-los na faixa de tensão do microcontrolador, que é de 0 a 3,3 V, aplicar dois filtros, sendo o primeiro um filtro rejeita faixa em 60 Hz e o segundo um filtro passa-banda Butterworth de terceira ordem, com frequência de corte inferior igual a 10 Hz e superior igual a 450 Hz.

A Figura 4 mostra o funcionamento geral do módulo *Bluetooth* ESP32. Ela apresenta como os dados serão adquiridos e processados, além de mostrar como a comunicação *Bluetooth* se dará. Além disso, evidencia a condição de ativação da FES, onde, caso ela seja atingida, o microcontrolador gera a estimulação elétrica e caso não seja atingida, a coleta continua sendo realizada no *loop* principal. Ainda no *loop* principal, ocorre o cálculo do RMS, para verificar a condição da FES, no que está descrito como processamento dos dados.

A Figura 5 mostra o funcionamento geral da aplicação *Web*. O *frontend* contém duas páginas *Web*, sendo a primeira um *dashboard* que contém o valor dos parâmetros necessários para o funcionamento do sistema MES-FES, que contém um formulário para a alteração desses parâmetros e botões para parar a coleta, iniciar a coleta, ativar a FES e ir para o gráfico. A outra página que a aplicação contém é uma página que mostra o gráfico dos dados pelo tempo, numa janela fixa.

A FES é ativada a partir de uma condição específica do valor do RMS de duas janelas. É feita uma comparação entre o valor do RMS de uma janela de 100 amostras e o valor do RMS de 100 amostras anteriores. Com isso, ao atingir o critério da equação 3, a FES é ativada.

4.4 Modelagem do sistema

A modelagem do sistema foi realizada baseada no diagrama presente na **??**, onde a sequência dada deve ser seguida. Então, o passo inicial é conectar os eletrodos diferenciais no membro desejado do participante. Em seguida, o sensor de biopotenciais baseado em AD8232 irá capturar os sinais sEMG e enviá-los ao módulo ESP32. O módulo então irá enviar esses dados ao servidor *Flask* (*backend*) que os mostrará num gráfico em função do tempo na página do gráfico (página secundária do *frontend*). A página principal da aplicação *Web* mostra o valor dos parâmetros alteráveis (*threshold*, tempo de ativação, intensidade e tempo pós FES), possibilita a alteração dos mesmos e ainda possui botões para visualizar o gráfico, parar a coleta, iniciar a coleta e ativar manualmente a FES.

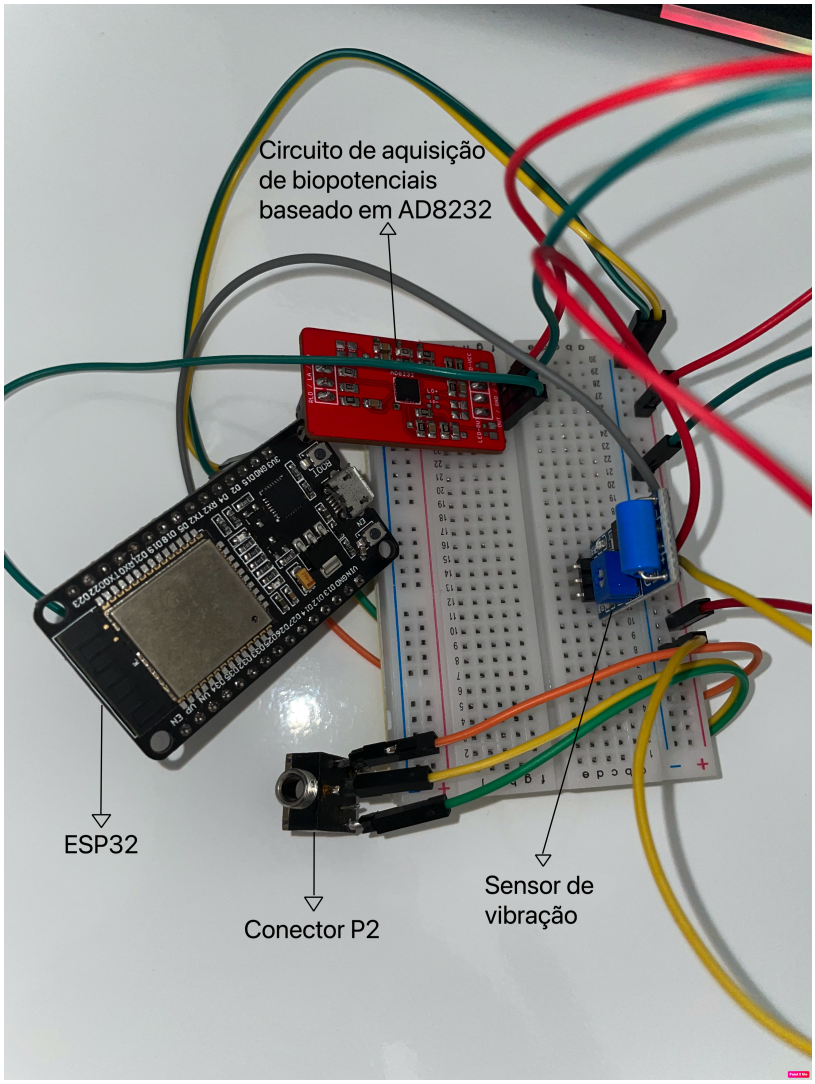
A ativação da FES, dada pela seta que liga a *dashboard* ao ESP32 e, após, o ESP32 ao sistema de ativação da FES ocorre após atingir a condição de ativação da FES, que é dada pela equação 3.

5 RESULTADOS

5.1 Apresentação do sistema

A montagem do hardware é mostrada na Figura 6. Ela apresenta os componentes utilizados para o funcionamento do sistema, como o módulo *Bluetooth* ESP32, o circuito de aquisição de biopotenciais baseado em AD8232, o sensor de vibração e o conector P2, que é responsável por receber os dados dos eletrodos.

Figura 6 – *Hardware* montado utilizado



Fonte: Autoria própria (2023).

As funcionalidades que o sistema possui são divididas em duas telas na aplicação *Web*. A página principal possui o formulário para alteração dos parâmetros, possui os valores atualizados dos parâmetros e os botões de ação.

Na Figura 7 é apresentada a tela para o usuário configurar os parâmetros necessários. Por meio dos botões 'Iniciar', 'Parar Coleta', 'Ativar FES' e 'Ir para o Gráfico', o usuário pode

reiniciar a coleta, parar a coleta, ativar a FES manualmente e ir para a página do gráfico ao vivo, respectivamente. Nessa *dashboard*, existem eventos de clique, para que o usuário saiba corretamente o que ocorreu ao clicar num botão, que são mostrados na Figura 8, na Figura 9 e na Figura 10.

Figura 7 – *Dashboard* interface MES-FES

Fonte: Autoria própria (2023).

Figura 8 – Balão de confirmação de alteração dos parâmetros

Fonte: Autoria própria (2023).

Figura 9 – Balão de confirmação de parada de coleta

Fonte: Autoria própria (2023).

Figura 10 – Balão de confirmação de início de coleta

Fonte: Autoria própria (2023).

A página do gráfico, a qual tem como finalidade mostrar ao usuário o sinal sEMG que está sendo coletado, tem aparência mostrada na Figura 11. Esta página é importante para verificar se o sinal está dentro do esperado, capturando as contrações, para verificar se os eletrodos estão posicionados corretamente e observar rapidamente as características do sinal sem a necessidade do uso de outros softwares. Neste caso, o gráfico mostra uma contração voluntária captada pelo sistema de aquisição. O gráfico é gerado a partir de um código em *JavaScript* que utiliza a biblioteca *PlotLy*.

Figura 11 – Página do gráfico

Fonte: Autoria própria (2023).

5.2 Implementação do sistema

O sistema desenvolvido coleta biossinais, mais especificamente os eletromiografia de superfície (sEMG), através do circuito de aquisição de biopotenciais baseado em AD8232 e de um módulo ESP32 *Bluetooth* e os envia ao *backend* da aplicação. O *firmware* do ESP32 conta com a leitura dos dados recebidos pelo sensor e o envio deles através do *Bluetooth*, a cada 1 ms. A todo instante é verificado o recebimento do comando - que determina a parada da coleta, o início da coleta e a ativação da estimulação elétrica funcional (do inglês *functional electrical stimulation*) manualmente - e a atualização dos parâmetros. Além disso, são armazenadas sempre duzentas amostras para o cálculo de MES_i e MES_{i-1} para ativar ou não a FES. Por fim, ele possui a rotina de ativação da FES que, após atingir o critério determinado na equação 3, aumenta gradualmente a estimulação elétrica até o valor definido pela intensidade, mantém a intensidade pelo período determinado pelo tempo de ativação e depois reduz gradualmente até não ter mais estimulação. Isso é feito para evitar contrações abruptas e, consequentemente, lesionar o participante.

Com relação aos botões presentes na aplicação, o Código 1 traz como eles são tratados no microcontrolador. Caso a serial *Bluetooth* possua pelo menos 1 byte disponível para leitura no *buffer* de entrada, este trecho será o responsável por lidar com este byte. Caso ele seja 0xFF, a coleta será interrompida, sendo retomada apenas ao receber o comando 0x01. Caso o comando recebido seja 0x02, a FES é ativada manualmente.

Código 1 – Comandos que devem ser recebidos ao clicar em um dos botões

```
if (SerialBT.available() >= 1) {
    uint8_t command = SerialBT.read();
    if (command == 0xFF) {
      coleta = false;
      digitalWrite(LED_BUILTIN, LOW);
      while (command != 0x01) {
        Serial.println("Coleta Parada");
        if (SerialBT.available() >= 1) {
          command = SerialBT.read();
          if (command == 0x01) {
            coleta = true;
            Serial.println("Coleta retomada");
            digitalWrite(LED_BUILTIN, HIGH);
          }
        }
      }
    } else if (command == 0x02) {
      Serial.println("Comando de ligar FES recebido!");
      FES();
    }
}
```

Fonte: Autoria própria (2023).

O Código 2 mostra como é feita a coleta e envio dos dados do microcontrolador para a aplicação. A variável coleta, do tipo *bool*, é a variável de controle da interrupção do *timer*, onde a cada 1 ms, ela assume valor lógico verdadeiro, permitindo a coleta e envio dos dados. Ainda nesse trecho do código, os valores da leitura são armazenados na lista amostras para o cálculo do RMS e é verificado se há uma vibração excessiva no membro do participante. Essa verificação se dá para evitar que a FES seja ativada quando houver muita vibração. No envio dos dados via *Bluetooth* é enviado um byte inicial, que atua como identificador para o receptor, marcando o início da transmissão dos dados.

A atualização dos parâmetros é feita com base no Código 3, onde caso haja 4 bytes disponíveis para leitura no *buffer* de entrada, significa que são os valores parâmetros. Eles são alterados na ordem em que são recebidos, ou seja, o primeiro valor enviado da aplicação *Web* corresponde ao *threshold*, o segundo ao tempo de ativação, o terceiro à intensidade e o quarto ao tempo pós FES.

Código 2 – Coleta e envio dos dados do microcontrolador via *Bluetooth*

```
if (coleta == true) {
    int index = 0;
    timerWrite(timer_coleta, 0);
    emgValue = analogRead(emgPin);
    SerialBT.write(0xCC);  //byte inicial
    SerialBT.write((emgValue >> 8));
    SerialBT.write((emgValue)&0xFF);
    amostras[index] = emgValue;
    coleta = false;
    index++;
    tilt = digitalRead(12);
    if (index == 200) {
      index = 0;
    }
}
```

Fonte: Autoria própria (2023).

Código 3 – Atualização dos parâmetros

```
if (SerialBT.available() >= 4) {
    th = SerialBT.read();
    ta = SerialBT.read();
    in = SerialBT.read();
    T_pos = SerialBT.read();
}
```

Fonte: Autoria própria (2023).

A função de ativação da FES é chamada quando a condição presente no Código 4 é atingida. Os parâmetros *th_amp_min* e *th_amp_max* são *thresholds* de amplitude máxima e mínima, definidos anteriormente no código.

Código 4 – Condição de ativação da FES

```
if (rms2 > (1 + th) * rms1 && rms2 > th_amp_min && rms2 < th_apm_max
   && tilt == 0) {
    Serial.println("Rotina FES");
    FES();
}
```

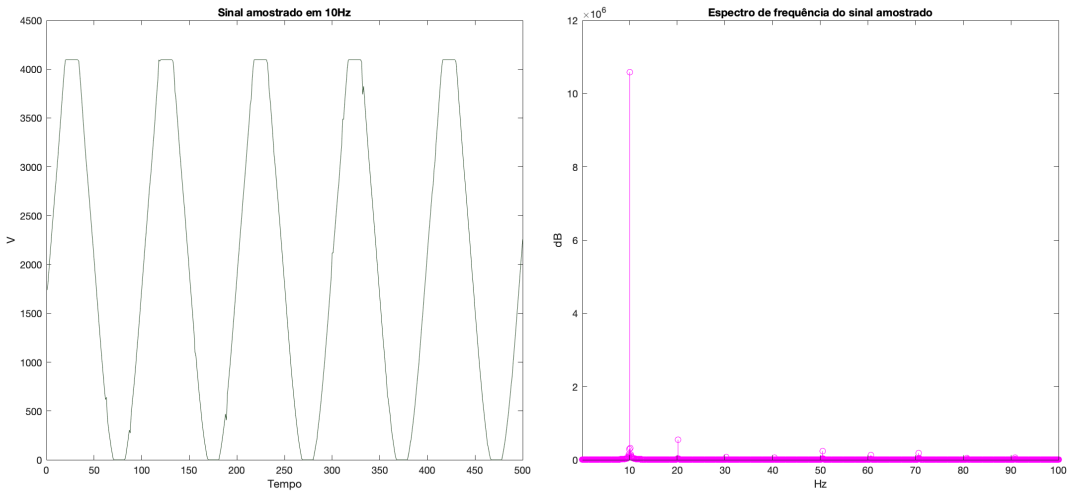
Fonte: Autoria própria (2023).

Para verificar se a taxa de amostragem estava constante em 1 kHz, foi realizado um teste em laboratório. Nele, foi coletado uma onda senoidal com frequência igual a 10 Hz, gerada através de um osciloscópio. Com isso, foi calculada a transformada rápida de Fourier (FFT) dada pela equação

$$F(u) = \frac{1}{N} \sum_{x=0}^{N-1} f(x) W_N^{ux}, \tag{4}$$

onde $W_N^{ux} = e^{-j2\pi u \cdot x/N}$, que converte o sinal do domínio original para uma representação no domínio da frequência. Com isso, foi possível observar que a maior componente está presente em 10 Hz, ou seja, o sistema possui taxa de amostragem constante, como mostra a Figura 12. Entretanto, surgiram algumas componentes em outras frequências, como em 20 Hz, 50 Hz e 60 Hz, por exemplo, são causadas pela distorção do sinal senoidal adquirido. Essa limitação observada ocorre quando os valores estão próximos do limite do conversor analógico digital, cuja faixa vai de zero à 3.3 V (analógico) ou de zero a 4095 (digital). No gráfico à esquerda da Figura 12 evidencia essa ocorrência.

Figura 12 – Teste de verificação da taxa de amostragem



Fonte: Autoria própria (2023).

No *backend*, o método apresentado na Código 5 é responsável por obter os dados vindos do *socket Bluetooth* - utilizando a biblioteca *pyserial* -, ajustá-los para a faixa de 0 V a 3,3 V (faixa do conversor analógico digital do módulo ESP32), convertê-los de 16 bits para inteiro e calcular o valor do tempo. Depois, a partir de um filtro rejeita faixa, serão excluídos os dados na frequência de 60 Hz, para eliminar qualquer interferência vinda da alimentação do *hardware*. Ainda nesse método, será utilizado um filtro de banda passante Butterworth de terceira ordem, com frequência de corte inferior 10 Hz e frequência de corte superior 450 Hz, pois é nessa faixa que estão os dados mais significativos para a EMG. Ambos os filtros pertencem à biblioteca *Signal* do *Scipy*.

Os dados obtidos por este método supracitado são enviados à aplicação por meio do método do Código 6, que recebe uma requisição do tipo GET e retorna um *JavaScript Object Notation* (JSON) contendo os dados e o tempo. Essa requisição é feita pela página do gráfico a cada 1 milissegundo, para ficar o mais próximo possível da aquisição.

Código 5 – Método read_serial_data()

```
def read_serial_data():
    global zi_notch, zi, fes
    start = 0
    while True:
        try:
            while int.from_bytes(ser.read(), "big") != 204:
                pass
            b1 = int.from_bytes(ser.read(), "big")
            b2 = int.from_bytes(ser.read(), "big")
            dado = b1 * 256 + b2
            dados.append(dado)
            if dado == 70:
                print('FES Ativada')
                fes = 1
            dado_conv = (dado*3.3)/4095
            dado_filtrado_notch, zi_notch = signal.lfilter(b_notch,
   a_notch, [dado_conv], zi=zi_notch)
            dado_filtrado, zi = signal.lfilter(b, a,
   dado_filtrado_notch, zi=zi)
            start = start + (1/1000)
            data_buffer.append(abs(dado_filtrado[0]))
            time_buffer.append(start)
        except ValueError:
            pass
```

Fonte: Autoria própria (2023).

Código 6 – Método get_data()

```
@app.route('/data')
def get_data():
    return jsonify({'time': time_buffer, 'data': data_buffer})
```

Fonte: Autoria própria (2023).

Para que o aplicativo seja responsivo, foi adicionada uma nova *thread* para o método do Código 5, pois como ele lê dados da porta serial em um *loop* infinito, bloquearia o servidor da *Web* e tornaria o aplicativo não responsivo a outras solicitações de clientes caso estivesse na *thread* principal. Os pontos abaixo foram importantes para a criação dessa nova *thread*:

- Bloqueio de Entrada/Saída: A leitura de dados da porta serial é uma operação de entrada/saída que pode ser bloqueante. Isso significa que, se não houver dados disponíveis na porta serial, a função *ser.read()* pode ficar esperando indefinidamente até que os dados cheguem. Durante esse tempo, o aplicativo *Flask* não poderia atender a outras solicitações dos clientes.

- Responsividade: Um aplicativo da *Web*, como o *Flask*, deve ser responsivo a todas as solicitações dos clientes. Se a leitura da porta serial fosse feita na *thread* principal,

qualquer solicitação HTTP ao aplicativo ficaria pendente até que a leitura da porta serial fosse concluída.

- *Threads*: O uso de uma *thread* separada para a leitura serial resolve esse problema. A função *read_serial_data()* é executada em segundo plano em uma *thread* separada, permitindo que o servidor *Flask* continue a atender às solicitações dos clientes, independentemente de quanto tempo a leitura da porta serial possa levar.

- *Daemon Thread*: É uma funcionalidade utilizada para fazer com que a *thread* seja encerrada automaticamente quando o programa principal encerrar. Isso é útil para evitar que o programa continue em execução indefinidamente, mesmo que o servidor da web *Flask* seja encerrado.

A atualização dos parâmetros se dá através do método do Código 7, que é uma rota do aplicativo *Web Flask* que recebe solicitações *POST*. Ela atualiza os parâmetros *threshold*, tempo de ativação, intensidade e tempo pós FES com os valores fornecidos na solicitação *POST* e, em seguida, envia esses novos parâmetros como bytes para o módulo ESP32. A função retorna uma resposta JSON indicando o sucesso da atualização.

Código 7 – Método update_parameters()

```
@app.route('/update_params', methods=['POST'])
def update_parameters():
    global th, ta, i, tempoPos
    data = request.form
    th = float(data['threshold'])
    ta = float(data['tempo_ativacao'])
    i = float(data['intensidade'])
    tempoPos = float(data['tempoPos'])
    ser.write(bytes([int(th), int(ta), int(i), int(tempoPos)]))
    return jsonify({'success': True})
```

Fonte: Autoria própria (2023).

Foram definidos métodos para tratar os comandos de parar e iniciar coleta, utilizando as anotações *socketio.on('update_params')* e *socketio.on('iniciar_coleta')*, respectivamente. Essas anotações lidam com a comunicação do *socket*, utilizado para poder ter uma aplicação em tempo real.

Quando o botão 'Parar Coleta' é pressionado, o servidor recebe um argumento data, que é um comando enviado pelo cliente, contendo o valor hexadecimal 0xFF e, em seguida, envia esse valor ao microcontrolador. Ao receber esse valor, o microcontrolador encerra a coleta de dados.

Quando o botão 'Iniciar' é pressionado, a mesma lógica é utilizada, porém nesse caso o comando contém o valor hexadecimal 0x01 e o evento acionado é *iniciar_coleta*.

Código 8 – Método stop_collection()

```
@socketio.on('update_params')
def stop_collection(data):
    command = data['command']
    ser.write(bytes([command]))
    print('Comando para parar coleta enviado ao ESP32')
```

Fonte: Autoria própria (2023).

Código 9 – Método start_collection()

```
@socketio.on('iniciar_coleta')
def start_collection(data):
    global coleta_ativa
    command = data['command']
    ser.write(bytes([command]))
    print('Comando para iniciar coleta enviado ao ESP32')
    coleta_ativa = True
```

Fonte: Autoria própria (2023).

No servidor há também a verificação do estado da FES, que é dado como ativado quando o microcontrolador, ao entrar na rotina de ativação da FES, envia o valor 70 dividido em dois bytes ao servidor. Assim que o servidor recebe este valor, a *flag* 'fes' obtém valor 1 e o método do Código 10 retorna um JSON confirmando a ativação. Além disso, na página *Web*, é feita uma comparação dos estados anterior e atual para que, ao haver mudança ela possa informar ao usuário que a FES foi ativada. O método do servidor e a função em *JavaScript* são mostrados abaixo, respectivamente no métodos do Código 10 e do Código 11.

Código 10 – Método FES()

```
@app.route('/FES')
    def FES():
        if fes == 1:
            return jsonify({'FES': 'ativada'})
        elif fes == 0:
            return jsonify({'FES': 'desligada'})
```

Fonte: Autoria própria (2023).

O servidor possui os métodos necessários para renderizar as páginas *Web*, a partir das rotas definidas nas anotações acima delas. Os métodos são mostrados abaixo, sendo, respectivamente, a rota para a página principal e a rota para o gráfico. Vale ressaltar que, além de renderizar a página principal, o método do Código 12 ainda passa o valor dos parâmetros para ela.

Código 11 – Função verificarEstadoFES()

```
let estadoAnterior = null;

    window.addEventListener('DOMContentLoaded', function () {
        verificarEstadoFES();
    });

    function verificarEstadoFES() {
        fetch('/FES')
            .then(response => response.json())
            .then(data => {
                const novoEstado = data.FES;
                if (estadoAnterior !== null && estadoAnterior === '
   desligada' && novoEstado === 'ativada') {
                    showBalloonMessage('FES Ativada!', 3000);
                }
                estadoAnterior = novoEstado;
            })
            .catch(error => {
                console.error('Ocorreu um erro ao chamar a rota FES:'
   , error);
            });
        setTimeout(verificarEstadoFES, 1000);
    }
```

Fonte: Autoria própria (2023).

Código 12 – Método index()

```
@app.route('/')
    def index():
        return render_template('index.html', threshold=th,
   tempo_ativacao=ta, intensidade=i, tempoPos=tempoPos)
```

Fonte: Autoria própria (2023).

Código 13 – Método chart()

```
@app.route('/chart')
    def chart():
        return render_template('chart.html')
```

Fonte: Autoria própria (2023).

Foram ainda implementados dois métodos para exportar um arquivo CSV com os dados em função do tempo. Os dados que serão exportados são os dados brutos, ou seja, antes de qualquer processamento no *backend*. Para isso, foram utilizadas as bibliotecas *io* e *csv*. Então, o método do Código 14 define uma rota para realizar o *download* do arquivo, configurada para receber solicitações GET. ele ainda está associado ao roteamento */download_csv* que é chamada quando o usuário pressiona o botão 'Download CSV', gerando como resposta a utilização do método do Código 15. Além disso, são definidos os cabeçalhos da resposta para o navegador, onde em *response.headers['Content-Type'] = 'text/csv'* é indicado o tipo de conteúdo

como CSV e em *response.headers['Content-Disposition'] = 'attachment; filename=dados.csv'* é especificado que o navegador deve fazer o download do arquivo com nome "dados.csv".

Por fim, o método do Código 15 é responsável por criar o conteúdo do arquivo. Primeiramente, é criado um objeto *output* do tipo *StringIO*, que atua como um *buffer* de *string* temporário. Na sequência, utilizando a biblioteca *csv* é criado o escritor. O cabeçalho das coluna é definido como [tempo, dado]. Por fim, os *buffers* que possuem os dados e o tempo são percorridos escrevendo cada par de valores no arquivo.

Código 14 – Método download_csv()

```
@app.route('/download_csv', methods=['GET'])
def download_csv():
    response = make_response(generate_csv())
    response.headers['Content-Type'] = 'text/csv'
    response.headers['Content-Disposition'] = 'attachment; filename=
   dados.csv'

    return response
```

Fonte: Autoria própria (2023).

Código 15 – Método generate_csv()

```
def generate_csv():
    output = io.StringIO()
    writer = csv.writer(output)
    writer.writerow(['tempo', 'dado'])

    for i in range(len(dados)):
        writer.writerow([time_buffer[i], dados[i]])

    return output.getvalue()
```

Fonte: Autoria própria (2023).

5.3 Discussões

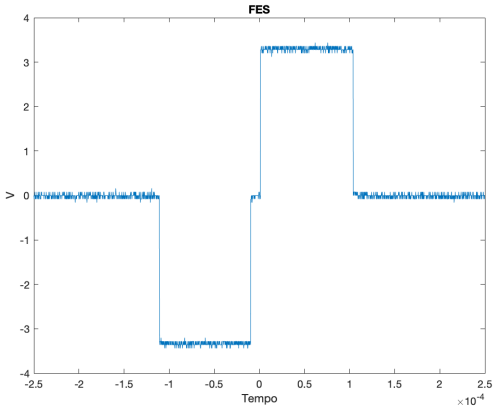
Para verificar o funcionamento do sistema foram realizados testes de obtenção de dados de sEMG, ativação da FES manualmente e ativação da FES através das condições especificadas na equação 3 juntamente com o sensor de vibração. Os testes realizados obtiveram dados parecidos com o dado apresentado na Figura 11. Nela é possível ver que o músculo estava em repouso e houve uma contração muscular voluntária, onde há a intenção e a realização do movimento, que neste caso foi a contração do músculo extensor radial curto do carpo, causando uma extensão do pulso.

O teste de verificação da FES foi dividido em duas etapas. A primeira delas foi o teste do botão 'Ativar FES' presente no *dashboard*. Como não havia um sistema de FES foi conectado

o osciloscópio nos pinos de saída da FES no ESP32. Assim, obtivemos como saída o dado da Figura 13.

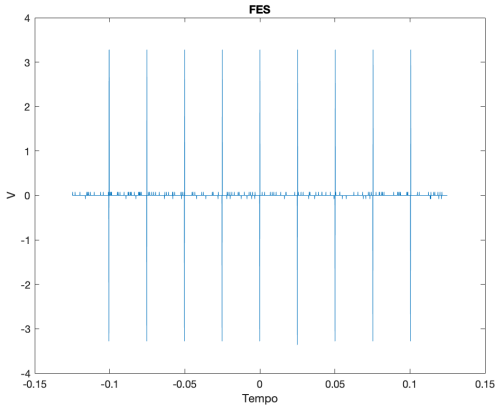
A Figura 13 mostra um único pulso emitido pelo módulo ESP32, onde há um degrau de 0 a -3,3 V, um pausa denominada interpulso e, na sequência, o pulso de 0 a 3,3 V. A FES é ativada aumentando gradativamente, mantendo-se ativada e depois diminuindo gradativamente. A Figura 14 mostra o momento em que a FES permanece ativada. A quantidade de pulsos varia de acordo com o valor do tempo de ativação. Essa figuras foram obtidas como saída em ambos os testes da FES, tanto no teste do botão (ativação forçada) quando no teste do vencimento das condições para ativação.

Figura 13 – Pulso único da saída da FES



Fonte: Autoria própria (2023).

Figura 14 – Período da FES ativada



Fonte: Autoria própria (2023).

Durante a realização dos testes, foram detectados alguns problemas e algumas melhorias futuras. O primeiro problema consiste na existência de picos que estouram o limite de 3.3 V no gráfico do EMG, provavelmente relacionados ao circuito de aquisição de biopotenciais. O segundo problema consiste na ocorrência de algumas desconexões do *socket Bluetooth* após um período grande de tempo. Isso ocorreu duas vezes durante a realização dos teste, sendo que após um longo período havia a perda da conexão, talvez pela inatividade.

6 CONCLUSÃO

O sistema MES-FES foi idealizado para ser um sistema capaz de adquirir sinais de eletromiografia de superfície com taxa de amostragem constante, ter pós-processamento num *backend* e alteração de parâmetros e visualização facilitada, utilizando tecnologias como o *Python* e um microcontrolador.

Sendo assim, o sistema foi desenvolvido com a maioria das funcionalidades que foram previstas anteriormente. As funcionalidades que o sistema inclui são: a aquisição de dados sEMG com taxa de amostragem constante, pós-processamento em *backend* na linguagem *Python* que retifica e filtra os dados, possibilita a alteração de parâmetros, possibilita a visualização dos dados em um gráfico ao vivo, possibilita parar a coleta, reiniciar a coleta, ativar a FES manualmente e salvar os dados em um arquivo CSV.

Algumas funcionalidades como a possibilidade de alterar o tamanho da janela que define MES_i e MES_{i-1}, o *threshold* de amplitude e o cálculo do período de incremento e decremento da FES, que determina o quão rápido a intensidade da estimulação elétrica irá aumentar ou diminuir não foram desenvolvidas, ficando para um trabalho futuro.

Foram detectados dois problemas durante a realização dos testes, sendo a presença de picos no gráfico do EMG, muito possivelmente vindos da placa de aquisição de biossinais baseado em AD8232, e duas desconexões do *socket Bluetooth* após um longo tempo de uso. Porém esse problemas e as funcionalidades que não foram implementadas, ficam para um trabalho futuro. As melhorias futuras incluem o aumento dos parâmetros alteráveis, como o período de subida e descida da FES e a implementação de rotinas que evitem a desconexão do *socket Bluetooth*.

Mesmo sem essas funcionalidades, o sistema é uma ferramenta capaz de fornecer o necessário para os profissionais da área, pois com ela conseguiram tratar algumas deficiências relacionadas ao movimento e também aos músculos. É importante ressaltar que além de terem a visualização dos dados ao vivo, a alteração dos parâmetros da FES e o botão de pausa na ferramenta, também será possibilitado ao usuário a manipulação desses dados da melhor maneira para cada caso, visto que será possível obter os dados brutos num arquivo CSV.

Portanto, o objetivo de proporcionar uma ferramenta para estudos na área da eletromiografia foi atingido.

REFERÊNCIAS

BANG, W.-S. *et al.* Electrical stimulation promotes functional recovery after spinal cord injury by activating endogenous spinal cord-derived neural stem/progenitor cell: an in vitro and in vivo study. **The Spine Journal**, 2023. ISSN 1529-9430. Disponível em: https://www.sciencedirect.com/science/article/pii/S1529943023034575.

CORREIA, V. **Neuropróteses ainda são alvo de debates entre especialistas**. 2017. Disponível em: https://www.correiobraziliense.com.br/app/noticia/tecnologia/2017/07/10/interna_tecnologia,608247/neuroproteses-ainda-sao-alvo-de-debates-entre-especialistas.shtml.

COSTA ANDRÉIA SIAS RODRIGUES, R. C. C. T. A. T. V. K. D. Investigação preliminar sobre interfaces de usuário em produtos de tecnologia assistiva. **ENPOS**, 2016.

Departamento de Atenção Especializada e Temática. **Manual de Boas Práticas de Gestão das Órteses, Próteses e Materiais Especiais (OPME)**. Brasília, 2016.

FIORIN, F. A. *et al.* The learning curve of people with complete spinal cord injury using a ness-fess interface in the sitting position: Pilot study. **Eng**, v. 4, n. 2, p. 1711–1722, 2023. ISSN 2673-4117. Disponível em: https://www.mdpi.com/2673-4117/4/2/97.

FOUNDATION, P. S. **Pyserial**. 2020. Disponível em: https://pypi.org/project/pyserial/.

GARCIA, A. M. D. P. V. E. N. Desafios contemporÂneos: O uso da tecnologia assistiva como instrumento facilitador da aprendizagem. **Linguagens, Educação e Sociedade**, n. 40, sep 2018.

GRINBERG, M. **Flask web development: developing web applications with python**. [*S.l.*]: "O'Reilly Media, Inc.", 2018.

GUPTA, S. *et al.* A survey of human-computer interaction (hci) natural habits-based behavioural biometric modalities for user recognition schemes. **Pattern Recognition**, v. 139, p. 109453, 2023. ISSN 0031-3203. Disponível em: https://www.sciencedirect.com/science/article/pii/S003132032300153X.

IBGE. **PESSOAS COM DEFICIÊNCIA**. 2010. Disponível em: https://educa.ibge.gov.br/jovens/conheca-o-brasil/populacao/20551-pessoas-com-deficiencia.html.

Instituto Federal do Rio Grande do Sul. **Tecnologia Assistiva**. 2021. Disponível em: https://cta.ifrs.edu.br/tecnologia-assistiva/conceito/.

JÚNIOR, J. M. *et al.* Ad8232 to biopotentials sensors: Open source project and benchmark. **Electronics**, v. 12, p. 833, 02 2023.

KILGORE, K. 13 - sensors for motor neuroprostheses. **Woodhead Publishing Series in Biomaterials**, p. 401–436, 2013. ISBN 9781845699871, https://doi.org/10.1533/9780857096289.3.401.

LI, G. *et al.* Continuous dynamic gesture recognition using surface emg signals based on blockchain-enabled internet of medical things. **Information Sciences**, v. 646, p. 119409, 2023. ISSN 0020-0255. Disponível em: https://www.sciencedirect.com/science/article/pii/S0020025523009945.

LOVELY, D. F. The origins and nature of the myoelectric signal. *In*: ____. **Powered Upper Limb Prostheses: Control, Implementation and Clinical Application**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004. p. 17–33. ISBN 978-3-642-18812-1. Disponível em: https://doi.org/10.1007/978-3-642-18812-1_2.

LYNCH, C. L.; POPOVIC, M. R. Functional electrical stimulation. **IEEE Control Systems Magazine**, v. 28, n. 2, p. 40–50, 2008.

MARQUEZ-CHIN, C.; POPOVIC, M. R. Functional electrical stimulation therapy for restoration of motor function after spinal cord injury and stroke: a review. **BioMedical Engineering OnLine**, v. 19, 2020. Disponível em: https://api.semanticscholar.org/CorpusID:218873586.

PEREIRA, N. F. *et al.* Efetividade da interface mess-fess na dorsiflexão de tornozelo após avc por cavernoma: estudo de caso. **Revista Neurociências**, v. 30, p. 1–26, set. 2022. Disponível em: https://periodicos.unifesp.br/index.php/neurociencias/article/view/13579.

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

WANG, H. P. *et al.* A wireless wearable surface functional electrical stimulator. **International Journal of Electronics**, v. 104, p. 1–13, 04 2017.

ZHANG, Q. *et al.* Fes-induced torque prediction with evoked emg sensing for muscle fatigue tracking. **Mechatronics, IEEE/ASME Transactions on**, v. 16, p. 816 – 826, 11 2011.

APÊNDICE A – Código Fonte da Aplicação

O código fonte da aplicação está disponível no link github.com/Joao-Pedro-ML/TCC-MES-FES