www.revistaft.com.br

Revista FT | ISSN 1678-0817 | v. 30, n. 164, 2026 | DOI: 10.69849/2kf0k112

ArenaSignal: aplicativo para processamento de sinais eletromiográficos

ArenaSignal: an application for electromyographic signal processing

Leticia Emily Gonçalves de Moraes[1]
Saulo Belarmino de Sousa Junior[2]
Carlos Enrique Lopez Noriega[3]

RESUMO

O processamento digital de sinais eletromiográficos (EMG) é essencial para a análise da atividade muscular em aplicações de reabilitação, biomecânica e interfaces homem-máquina, mas muitas ferramentas disponíveis dependem de licenças comerciais ou apresentam alta complexidade de uso. Este trabalho apresenta o desenvolvimento do ArenaSignal, um aplicativo *desktop* multiplataforma, gratuito e de código aberto, desenvolvido em *Python* 3.11, para processamento e visualização de sinais EMG. O sistema possui arquitetura modular e implementa as principais etapas do *pipeline* de processamento: correção da média, filtragem digital configurável, retificação de onda completa e detecção de *onset*/*offset* por envoltória linear com histerese. A validação foi realizada por meio de testes funcionais manuais e automatizados (suíte de 25 testes com *pytest*), utilizando sinais EMG simulados com parâmetros controlados, além de dados externos de EMG facial adquiridos com MyoTec. Os resultados confirmaram o funcionamento correto das etapas implementadas e a capacidade do sistema de carregar e representar sinais reais. O ArenaSignal constitui uma ferramenta acessível e reprodutível voltada ao ensino e à experimentação em processamento de biossinais.

Palavras-chave: Eletromiografia de superfície; Processamento digital de sinais; Filtro *Butterworth*; Detecção de *onset*/*offset*; *Software* educacional.

ABSTRACT

Digital processing of electromyographic (EMG) signals is essential for analyzing muscle activity in rehabilitation, biomechanics, and human–machine interface applications, but many available tools rely on commercial licenses or are highly complex to use. This work presents the development of ArenaSignal, a free, open-source, cross-platform desktop application developed in Python 3.11 for processing and visualizing EMG signals. The system has a modular architecture and implements the main stages of the processing pipeline: mean correction, configurable digital filtering, full-wave rectification, and onset/offset detection by linear envelope with hysteresis. Validation was performed through manual and automated functional tests (a suite of 25 tests with pytest), using simulated EMG signals with controlled parameters, as well as external facial EMG data acquired with MyoTec. The results confirmed

1 Faculdade Engenheiro Salvador Arena – São Bernardo do Campo – São Paulo – Brasil. ORCID: https://orcid.org/0009-0008-3753-6083

2 Faculdade Engenheiro Salvador Arena – São Bernardo do Campo – São Paulo – Brasil. ORCID: https://orcid.org/0009-0007-8204-5921

3 Faculdade Engenheiro Salvador Arena – São Bernardo do Campo – São Paulo – Brasil. ORCID: https://orcid.org/0000-0001-6262-1177

the correct operation of the implemented stages and the system's ability to load and represent real signals. ArenaSignal constitutes an accessible and reproducible tool aimed at teaching and experimentation in biosignal processing.

Keywords: Surface electromyography; Digital signal processing; Butterworth filter; Onset/offset detection; Educational software.

1 INTRODUÇÃO

O avanço das tecnologias biomédicas tem impulsionado o desenvolvimento de sensores miniaturizados, vestíveis e de baixo consumo energético, ampliando as possibilidades de monitoramento fisiológico contínuo e não invasivo. Sinais cardíacos, cerebrais, musculares e respiratórios podem ser captados em diferentes contextos, gerando dados que necessitam de processamento para sua interpretação. Entre as modalidades de biopotenciais, a eletromiografia (EMG) destaca-se no estudo da função neuromuscular, permitindo a análise da atividade elétrica produzida pelos músculos durante a contração (Reaz; Hussain; Mohd-Yasin, 2006). Sua aplicação abrange áreas como reabilitação, biomecânica, controle de próteses, interfaces homem-máquina e diagnóstico de distúrbios neuromusculares (Chowdhury *et al.*, 2013; Reaz; Hussain; Mohd-Yasin, 2006).

Os sinais EMG apresentam baixa amplitude e são suscetíveis a interferências como *offset* de corrente contínua (componente constante indesejada que desloca a linha de base do sinal), ruídos eletrônicos, artefatos de movimento e interferência da rede elétrica, podendo comprometer a qualidade das informações obtidas (Chowdhury *et al.*, 2013; Reaz; Hussain; Mohd-Yasin, 2006). Dessa forma, o processamento digital é necessário para reduzir esses efeitos e preservar as características de interesse do sinal. Esse processamento envolve etapas como correção da média, filtragem, retificação e detecção de *onset*/*offset*, permitindo melhorar a interpretação da atividade muscular e possibilitar comparações entre sinais obtidos em diferentes condições (Chowdhury *et al.*, 2013; Halaki; Ginn, 2012; Reaz; Hussain; Mohd-Yasin, 2006).

As etapas de processamento podem ser implementadas em sistemas computacionais capazes de automatizar rotinas, testar diferentes parâmetros e permitir a visualização dos sinais ao longo do processamento, favorecendo a validação e a reprodutibilidade (Chowdhury *et al.*, 2013; Reaz; Hussain; Mohd-Yasin, 2006). Entretanto, apesar da disponibilidade de diferentes ferramentas para análise de EMG, ainda existe a necessidade de soluções que reúnam essas etapas em um ambiente modular e de fácil compreensão, especialmente para estudos, validações e experimentação com sinais simulados (Halaki; Ginn, 2012; Klaić *et al.*,

2026). Nesse contexto, justifica-se o desenvolvimento de uma ferramenta que integre processamento e visualização dos sinais EMG, com estrutura modular e suporte à utilização de dados sintéticos.

Ferramentas comerciais, como o MATLAB com suas *toolboxes* especializadas (*Signal Processing Toolbox*, *Bioinformatics Toolbox*), oferecem recursos robustos, porém dependem de licenças pagas (cerca de US$ 8.000/ano para uso comercial ou US$ 500 para estudante). Soluções específicas como Delsys EMGworks são gratuitas, mas limitadas ao *hardware* da marca, enquanto MyoLab custa aproximadamente US$ 300. Alternativas *open source*, como pyEMG e OpenSignals, podem apresentar maior flexibilidade, porém muitas vezes são incompletas ou com documentação escassa (Colombet *et al.*, 2015; Hauer, 2024). Essa lacuna evidencia a importância de uma ferramenta gratuita, modular e de código aberto que atenda tanto ao ensino quanto à pesquisa.

Diante desse contexto, este trabalho tem como objetivo desenvolver um sistema computacional destinado ao processamento digital de sinais eletromiográficos, implementando as principais etapas do processamento digital descritas na literatura (Chowdhury *et al.*, 2013; Reaz; Hussain; Mohd-Yasin, 2006). O sistema busca permitir a análise detalhada de cada etapa do processamento, oferecendo uma interface modular que favoreça a compreensão dos efeitos das técnicas sobre o sinal.

2 REVISÃO DA LITERATURA

Os sinais biomédicos correspondem às manifestações físicas do organismo que podem ser medidas para avaliar diferentes sistemas biológicos. Entre eles, destacam-se os biopotenciais, como o eletrocardiograma (ECG), o eletroencefalograma (EEG) e o eletromiograma (EMG), amplamente empregados em diagnóstico, monitoramento e pesquisa. Sua aquisição envolve amplificação, digitalização e processamento computacional, sendo necessária atenção às interferências externas devido à baixa amplitude desses sinais (Koo, 2025). Segundo Klaić *et al.* (2026), fatores como a interface eletrodo-pele, o acoplamento capacitivo e os artefatos de movimento podem comprometer a relação sinal-ruído, evidenciando a importância de sistemas de aquisição adequadamente projetados.

A eletromiografia (EMG) consiste no monitoramento da atividade elétrica das membranas excitáveis e permite avaliar a função muscular por meio dos sinais elétricos gerados durante a contração (Konrad, 2005; Marchetti; Duarte, 2006). A EMG de superfície constitui um método não invasivo amplamente utilizado em contextos clínicos e de pesquisa,

cuja qualidade depende de fatores como seleção e posicionamento dos eletrodos, preparação da pele e parâmetros de filtragem (Duarte *et al.*, 2011; Marchetti; Duarte, 2006). O sinal está relacionado à atividade das unidades motoras, formadas por um motoneurônio, seu axônio e as fibras musculares por ele inervadas (Araujo, 1998). Dessa forma, a EMG possui aplicações no diagnóstico neuromuscular, reabilitação, biomecânica, fisiologia do exercício e ergonomia (Araujo, 1998; Duarte *et al.*, 2011; Konrad, 2005).

A EMG facial permite analisar a atividade dos músculos responsáveis pelas expressões faciais, destacando-se o corrugador do supercílio (*corrugator supercilii*) e o zigomático maior (*zygomaticus major*), associados, respectivamente, a respostas de valência negativa e positiva. A técnica possibilita investigar emoções, microexpressões, mímica facial, empatia e respostas implícitas em contextos de interação social e neuromarketing. Também apresenta aplicações em Interfaces Humano-Computador, nas quais a atividade muscular pode contribuir para sistemas capazes de responder a estados emocionais (Klaić *et al.*, 2026). Nesse contexto, *datasets* públicos, como RAVDESS (*Ryerson Audio-Visual Database of Emotional Speech and Song*) e PeakAffectDS, são relevantes para desenvolvimento, comparação e validação de abordagens.

O *pipeline* envolve, de forma geral, a remoção do *offset* de corrente contínua, a filtragem, a retificação, a detecção de *onset*/*offset* e a normalização (Ferreira; Guimarães; Silva, 2010; Wu *et al.*, 2022). A correção do *offset* ajusta a linha de base do sinal, calculando a média do sinal e subtraindo-a de cada amostra:

$$x_c(n) = x(n) - \underline{x} \tag{1}$$

onde $x(n)$ é o sinal original e $\bar{x}$ é sua média (Wu *et al.*, 2022). A filtragem reduz componentes indesejados e preserva a faixa de interesse. Para a EMG de superfície, utiliza-se comumente filtros Butterworth passa-banda com frequências de corte entre 20 e 450 Hz, além de filtro *notch* para rejeição de 50/60 Hz (Chowdhury *et al.*, 2013; Ferreira; Guimarães; Silva, 2010; Marchetti; Duarte, 2006; Shields *et al.*, 2026). O filtro Butterworth é escolhido por sua resposta plana na banda passante, evitando ondulações que distorcem a amplitude, inclusive em sinais de baixa magnitude (Mello *et al.*, 2007). A aplicação da filtragem deve ser criteriosa para evitar alterações na amplitude e na fase; nesse sentido, a técnica de filtragem bidirecional com *filtfilt* proporciona resposta de fase zero, preservando o alinhamento temporal dos eventos (Ferreira; Guimarães; Silva, 2010; Virtanen *et al.*, 2020). A retificação de onda completa converte o sinal bipolar em unipolar por meio do valor absoluto:

$$x_r(n) = |x(n)| \tag{2}$$

preservando a informação de amplitude (Ferreira; Guimarães; Silva, 2010; Reaz; Hussain; Mohd-Yasin, 2006; Wu *et al.*, 2022). A detecção de *onset*/*offset* identifica os períodos de atividade muscular a partir de uma envoltória linear do sinal retificado, obtida com filtro Butterworth de 2ª ordem, passa-baixa, corte em 6 Hz, aplicado com *filtfilt* para preservar a fase. A linha de base (*baseline*) corresponde às amostras da envoltória abaixo do percentil 25, e seu desvio-padrão recebe um piso de 20% da média, para que o limiar não caia dentro da flutuação normal do ruído. Seguindo o método de envoltória linear com histerese de Hodges e Bui (1996), a detecção emprega dois limiares, na razão de 0,5:

$$Limiar_{início} = \mu_{baseline} + 3\sigma_{baseline} \quad (3)$$

$$Limiar_{fim} = \mu_{baseline} + 1{,}5\sigma_{baseline} \quad (4)$$

onde $\mu_{baseline}$ e $\sigma_{baseline}$ são média e desvio-padrão da linha de base. Um *burst* inicia quando a envoltória ultrapassa o limiar de início e só termina quando ela cai abaixo do limiar de fim, evitando a oscilação em torno de um único limiar (Ferreira; Guimarães; Silva, 2010; Wu *et al.*, 2022). *Bursts* separados por menos de 100 ms são fundidos, e os que permanecem com duração inferior a 100 ms são descartados como ruído. Por fim, a normalização possibilita comparações entre músculos e indivíduos, contribuindo para a reprodutibilidade das análises (Halaki; Ginn, 2012; Silva Júnior, 2013).

O avanço das ferramentas computacionais ampliou as possibilidades de processamento de EMG, mas permanecem desafios relacionados à acessibilidade, reprodutibilidade e complexidade metodológica (Adlbi, 2026; Hauer, 2024). Ferramentas comerciais, como MATLAB, oferecem recursos especializados, porém dependem de licenças, enquanto soluções *open source* podem apresentar maior flexibilidade e integração (Hauer, 2024; Silva *et al.*, 2023). O *Python* destaca-se como alternativa aberta, com ampla disponibilidade de bibliotecas e possibilidade de desenvolvimento de aplicações voltadas à reprodutibilidade e à redução da barreira de entrada (Hauer, 2024; Silva *et al.*, 2023). Além da linguagem, a modularidade constitui um aspecto importante, permitindo separar componentes e incorporar novas técnicas de forma independente (IEEE, 2000; Sommerville, 2011). Essa abordagem reforça a necessidade de ferramentas abertas, adaptáveis e reprodutíveis para o processamento de EMG e outros biossinais (Adlbi, 2026; Colombet *et al.*, 2015).

3 METODOLOGIA

A crescente utilização da eletromiografia (EMG) em contextos clínicos e de pesquisa tem ampliado a demanda por ferramentas computacionais acessíveis para processamento e

análise de sinais (Chowdhury *et al.*, 2013; Reaz; Hussain; Mohd-Yasin, 2006). Entretanto, parte das soluções disponíveis é proprietária e de alto custo, enquanto alternativas *open source* podem exigir conhecimento avançado em programação ou apresentar arquiteturas pouco modulares (Adlbi, 2026; Hauer, 2024). Nesse contexto, este trabalho desenvolve uma ferramenta computacional gratuita e de código aberto para processamento e visualização de sinais EMG, com interface gráfica voltada ao ensino e à experimentação no projeto EMG/EEG da Faculdade Engenheiro Salvador Arena.

Quanto à natureza, a pesquisa é aplicada, pois busca solucionar um problema prático específico: a carência de ferramentas livres e modulares para processamento de sinais EMG (Gil, 2002; Prodanov; Freitas, 2013). Quanto aos procedimentos técnicos, caracteriza-se como desenvolvimento de *software*, envolvendo concepção, implementação e validação de um sistema computacional para processamento de biossinais (Hauer, 2024). A abordagem é quantitativa, uma vez que o sistema realiza operações matemáticas sobre dados numéricos, incluindo filtragem e transformação dos sinais (Creswell, 2010).

O objetivo consiste em desenvolver, implementar e validar uma ferramenta para processamento e visualização de sinais EMG utilizando *Python* e seu ecossistema científico. Busca-se ainda disponibilizar o sistema como projeto *open source* no GitHub, favorecendo a reprodutibilidade, colaboração e utilização didática (Hauer, 2024; Prodanov; Freitas, 2013; Silva *et al.*, 2023). A metodologia contempla o ambiente de desenvolvimento, a arquitetura do *software*, a implementação das técnicas de processamento, os testes funcionais e a validação por meio de sinais EMG simulados e dados externos.

O sistema foi desenvolvido como aplicação *desktop* em *Python* 3.11, com suporte multiplataforma (Windows e macOS). A linguagem foi escolhida devido à sua ampla utilização em computação científica, disponibilidade de bibliotecas especializadas e facilidade de integração entre módulos (Rossum *et al.*, [20--]). Seu caráter *open source* também está alinhado à proposta de acessibilidade e reprodutibilidade do projeto (Silva *et al.*, 2023).

O Quadro 1 sintetiza as principais ferramentas empregadas no desenvolvimento e suas respectivas finalidades.

Quadro 1 – Tecnologias utilizadas no desenvolvimento do sistema.

Tecnologia	Finalidade
Python 3.11	Linguagem de programação principal
NumPy	Manipulação de vetores e operações matemáticas
SciPy	Processamento digital de sinais (filtros Butterworth, notch, filtfilt e espectro via `welch`)
Matplotlib	Geração e visualização de gráficos
Tkinter	Desenvolvimento da interface gráfica
`FigureCanvasTkAgg` / `NavigationToolbar2Tk`	Visualizador de figuras embutido na interface, com navegação
venv	Gerenciamento de ambiente virtual
Git/GitHub	Controle de versão e hospedagem do código
pytest	Testes automatizados das funcionalidades

Fonte: Elaboração própria dos autores (2026).

O NumPy (Harris *et al.*, 2020) foi utilizado para armazenamento dos sinais em *arrays* e operações matemáticas, incluindo o cálculo e a remoção do *offset*. A SciPy (Virtanen *et al.*, 2020) foi empregada na implementação dos filtros Butterworth (por meio da função *butter*) e *notch* (por meio da função *iirnotch*), bem como na filtragem bidirecional com *filtfilt*. A Matplotlib (Hunter, 2007) foi utilizada na geração dos gráficos. A interface gráfica foi desenvolvida com Tkinter (Moore, 2018), permitindo executar o processamento e visualizar os resultados sem interação direta com a linha de comando.

Para garantir o isolamento das dependências, utilizou-se o ambiente virtual *venv*/ (Python Software Foundation, 2026b). O controle de versão foi realizado com Git (Preissel; Stachmann, 2014) e o código disponibilizado no GitHub (Hauer, 2024; Preissel; Stachmann, 2014). O desenvolvimento utilizou as IDEs Spyder e Visual Studio Code, enquanto um *script setup-win.bat* automatiza a configuração do ambiente e a instalação das dependências.

A estrutura do projeto separa os *scripts* didáticos das funcionalidades da aplicação principal, mantendo os códigos experimentais em *core-codes-with-coments/* e a aplicação em *project-with-frontend/src/*. Os gráficos são armazenados em *project-with-frontend/src/output_images/*, caminho ancorado à pasta do próprio *script* para que o local das figuras independa de como a aplicação é iniciada, enquanto *requirements.txt* e *README.md* concentram, respectivamente, as dependências e a documentação do projeto.

A arquitetura foi organizada de forma modular, separando a interface e o controle da aplicação das rotinas de processamento. Essa organização segue a prática de dividir um sistema em componentes menores e especializados, o que reduz o acoplamento entre eles e facilita a manutenção e a inclusão de novas funcionalidades (IEEE, 2000; Sommerville,

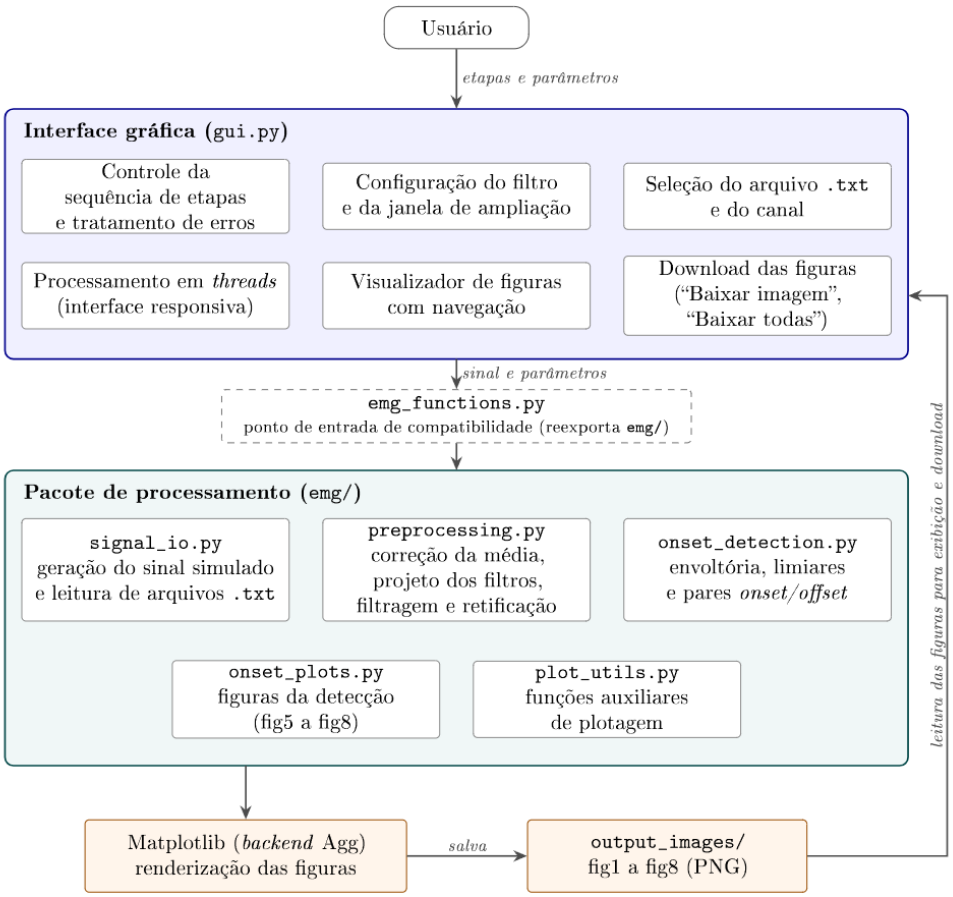
2011). A camada de interface, implementada em *gui.py*, é responsável pela interação com o usuário e pelo controle da sequência de execução. As rotinas de processamento estão organizadas no pacote *emg/*, dividido em módulos por responsabilidade: geração e leitura dos sinais (*signal_io.py*), correção da média, projeto dos filtros e retificação (*preprocessing.py*), detecção de *onset*/*offset* (*onset_detection.py*), geração das figuras de detecção (*onset_plots.py*) e funções auxiliares de plotagem (*plot_utils.py*). O módulo *emg_functions.py* atua como ponto de entrada de compatibilidade, reexportando o pacote *emg/*, cujo *__init__.py* fixa o *backend Agg* do Matplotlib, necessário porque a interface processa os sinais em *threads*.

O fluxo de execução inicia-se com a geração ou carregamento do sinal EMG e segue, de forma sequencial, pelas etapas de correção da média, filtragem, retificação e detecção de *onset*/*offset*. Para arquivos .txt, o sistema detecta automaticamente o delimitador (vírgula ou espaço/tabulação), a presença de cabeçalho, o separador decimal por vírgula e a presença ou ausência de coluna de tempo, aceitando arquivos com uma única coluna de amplitude, com tempo mais N canais, ou com N canais sem coluna de tempo, por meio das funções *load_txt_signal* e *peek_txt_channels*. A taxa de amostragem de 1000 Hz é o padrão, usado no sinal simulado e em arquivos sem coluna de tempo; quando o .txt traz coluna de tempo, a taxa é inferida como o inverso da média das diferenças entre amostras. Os resultados são encaminhados ao módulo de visualização e apresentados na interface, podendo posteriormente ser exportados em formato PNG. A interface também controla a ordem de execução das etapas, impedindo que uma operação seja realizada antes da conclusão da etapa anterior.

A correção da média (Eq. 1) é implementada com *np.mean*. A filtragem permite configurar o tipo (passa-baixa, passa-alta, passa-banda, *notch*) e as frequências de corte, projetadas com *scipy.signal.butter* e aplicadas com *filtfilt* para fase zero; a ordem (4) e o fator de qualidade do *notch* (Q = 30) são fixos no código. Para validação, usou-se filtro passa-banda de 4ª ordem, 20-450 Hz. A retificação (Eq. 2) usa *np.abs*. A detecção de *onset*/*offset* segue o algoritmo descrito nas Eq. 3 e 4.

A Figura 1 ilustra a arquitetura do sistema, evidenciando seus principais componentes e o fluxo de dados entre eles.

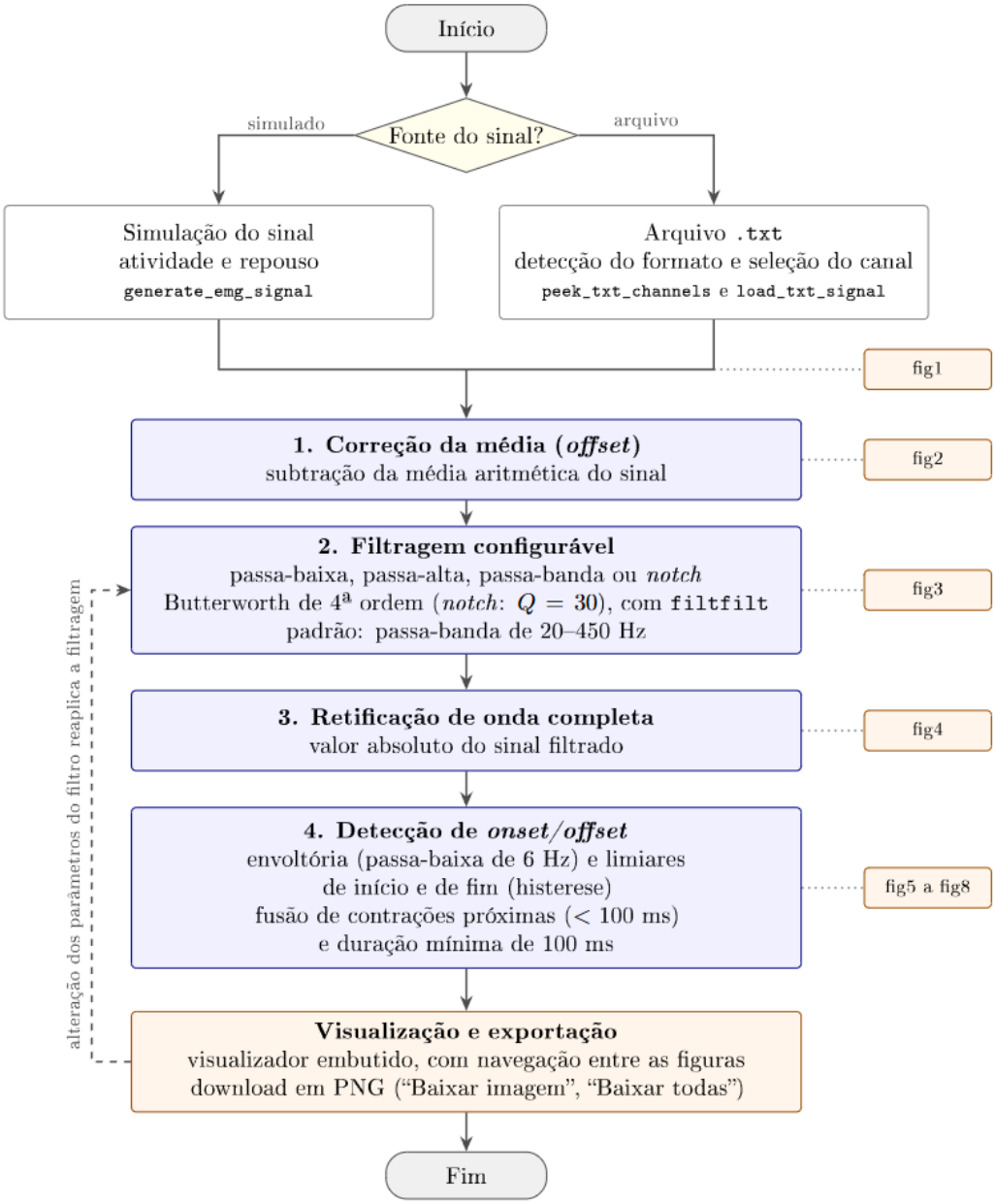
Figura 1 – Diagrama da arquitetura do sistema.



Fonte: Elaboração própria dos autores (2026).

A Figura 2 sintetiza o fluxo completo de processamento implementado no sistema, desde a geração do sinal simulado ou o carregamento de um arquivo .txt até a exibição e a exportação das figuras.

Figura 2 – Diagrama de fluxo do sistema de processamento de EMG.



Fonte: Elaboração própria dos autores (2026).

A verificação do sistema foi dividida em duas frentes: testes funcionais, destinados a verificar o funcionamento da aplicação, e validação do processamento, destinada a verificar se as técnicas implementadas produzem resultados compatíveis com o comportamento esperado para sinais EMG. Essa distinção é importante, pois a execução de uma rotina sem erros não garante, por si só, a correção científica do resultado (Pressman; Maxim, 2020; Sommerville, 2015).

Os testes funcionais foram realizados de forma sequencial, acompanhando o fluxo de utilização da aplicação: geração do sinal, correção da média, filtragem, retificação, detecção

de *onset*/*offset*, visualização e exportação. Foram verificadas a execução das funções, a atualização da interface, a geração dos gráficos, o salvamento das figuras e o controle da ordem de execução. Também foram avaliadas situações de uso incorreto, como a tentativa de executar uma etapa antes da conclusão da anterior. Essa abordagem corresponde aos testes funcionais ou de caixa-preta, voltados à verificação dos requisitos operacionais do *software* (Myers; Sandler; Badgett, 2012; Pressman; Maxim, 2020; Sommerville, 2015).

Além dos testes manuais, foi utilizada uma suíte automatizada com *pytest*, aplicada às funções do módulo de processamento e aos componentes da aplicação. A suíte reúne 25 testes, sendo 13 referentes às funções de processamento (incluindo os quatro tipos de filtro, parametrizados) e 12 aos componentes da interface; na execução realizada em 30/10/2026, todos os 25 testes passaram, em 9,11 s. Os testes utilizaram entradas controladas e compararam as saídas obtidas com valores previamente definidos, permitindo identificar falhas durante o desenvolvimento. Os casos de teste foram considerados satisfatórios quando as funções foram executadas sem erros, os sinais processados apresentaram o comportamento esperado, os gráficos foram corretamente gerados e a interface respeitou a sequência estabelecida.

A validação do processamento foi realizada em duas frentes: com sinais EMG simulados, com parâmetros conhecidos e controlados, e com dados externos obtidos em experimento complementar. Para os sinais simulados, cada etapa foi avaliada com critérios específicos: na correção da média, verificou-se a remoção do *offset* e o posicionamento da linha de base em torno de zero; na filtragem, a aplicação do filtro Butterworth de 4ª ordem, a redução visual dos componentes fora da faixa de 20-450 Hz e a ausência de distorções temporais; na retificação, a conversão dos valores negativos para positivos e a preservação relativa da amplitude (Chowdhury *et al.*, 2013; Ferreira; Guimarães; Silva, 2010; Reaz; Hussain; Mohd-Yasin, 2006; Wu *et al.*, 2022); na detecção de *onset*/*offset*, a identificação correta dos períodos de atividade, com pares *onset*/*offset* coerentes e geração adequada da figura correspondente.

Para a validação com dados externos, foram utilizados sinais EMG faciais adquiridos com o dispositivo MyoTec, considerando músculos associados à expressão facial (corrugador do supercílio e zigomático maior). O *software* de aquisição do MyoTec gera arquivos no formato .edf, os quais foram convertidos para .txt utilizando o MATLAB. Esse arquivo .txt foi então carregado no sistema desenvolvido por meio da funcionalidade *load_txt_signal*, e a representação gráfica foi gerada. Essa abordagem permitiu verificar a capacidade do sistema de carregar e representar graficamente sinais EMG reais a partir do formato .txt.

O procedimento de validação compreendeu a geração do sinal simulado, a aplicação sequencial das técnicas de processamento, a geração dos gráficos e a execução dos testes funcionais e automatizados. Quando foram identificadas inconsistências, estas foram corrigidas e os testes foram novamente executados para verificar a efetividade das correções. A análise foi baseada principalmente na inspeção dos sinais antes e depois de cada etapa e no comportamento da aplicação durante sua execução.

A validação apresenta algumas limitações. Os testes foram realizados predominantemente com sinais EMG simulados, e a validação com dados externos restringiu-se à verificação do carregamento e da representação gráfica do arquivo .txt, sem uma comparação quantitativa do processamento. Não foram utilizadas métricas como RMS, SNR ou correlação cruzada, e não houve comparação com equipamentos comerciais de eletromiografia. Embora o sistema disponha de funcionalidade para carregamento de arquivos .txt, permitindo a utilização de sinais multicanais, a validação apresentada foi realizada com um conjunto limitado de dados. Essas limitações são compatíveis com o objetivo proposto para o sistema, voltado ao ensino e à experimentação, e delimitam a interpretação dos resultados obtidos.

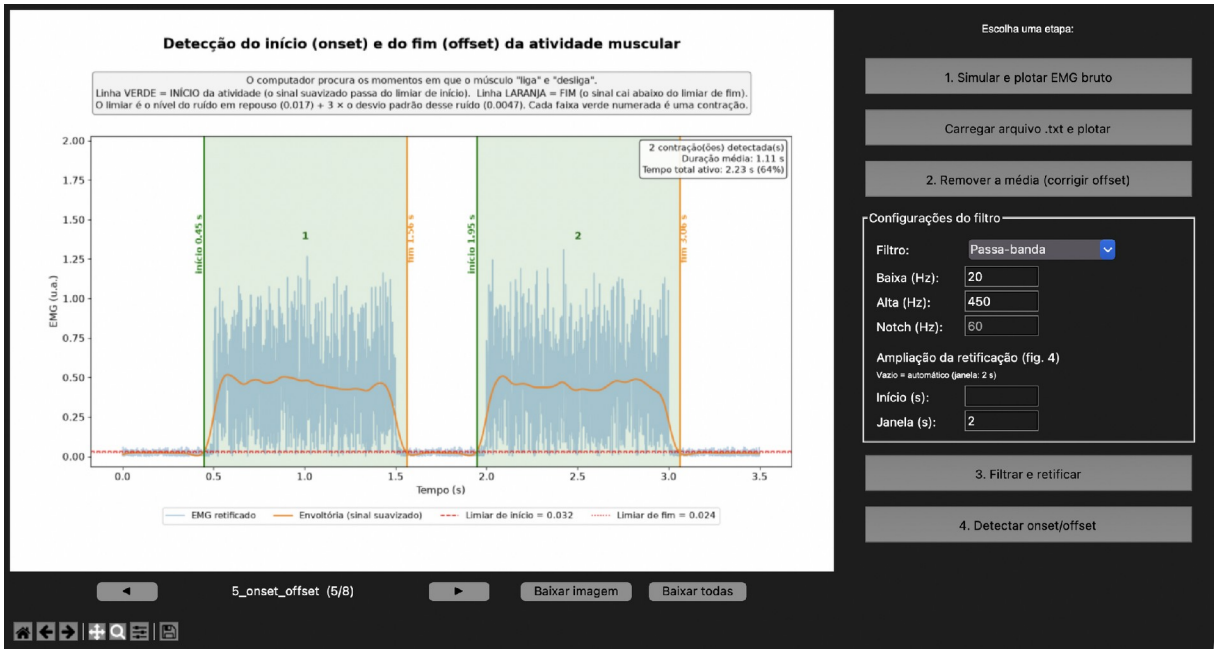
4 RESULTADOS E DISCUSSÃO

O sistema desenvolvido foi submetido aos procedimentos de teste e validação descritos na metodologia, tanto com sinais EMG simulados quanto com dados externos. Os resultados são apresentados a seguir, organizados de acordo com as etapas de processamento e validação.

A interface gráfica desenvolvida com Tkinter mostrou-se responsiva e de navegação intuitiva, permitindo ao usuário executar as etapas de processamento de forma sequencial, com *feedback* visual imediato. As figuras geradas são apresentadas em um visualizador embutido na própria interface, com navegação entre as figuras e indicação da posição atual; os botões "Baixar imagem" e "Baixar todas" copiam as figuras em resolução original para a pasta escolhida pelo usuário. Ao carregar um arquivo .txt com mais de um canal, a interface exibe uma caixa de diálogo para seleção do canal a ser processado. O controle de sequência impediu a execução de etapas fora de ordem, emitindo mensagens adequadas ao usuário. A interface também apresenta os parâmetros configuráveis para os diferentes tipos de filtro (passa-baixa, passa-alta, passa-banda e *notch*), permitindo ao usuário ajustar as frequências de corte; alterações nessa configuração após a etapa de filtragem reaplicam o filtro automaticamente. A

Figura 3 mostra a tela principal do aplicativo, com o visualizador de figuras, os botões de cada etapa e as configurações do filtro.

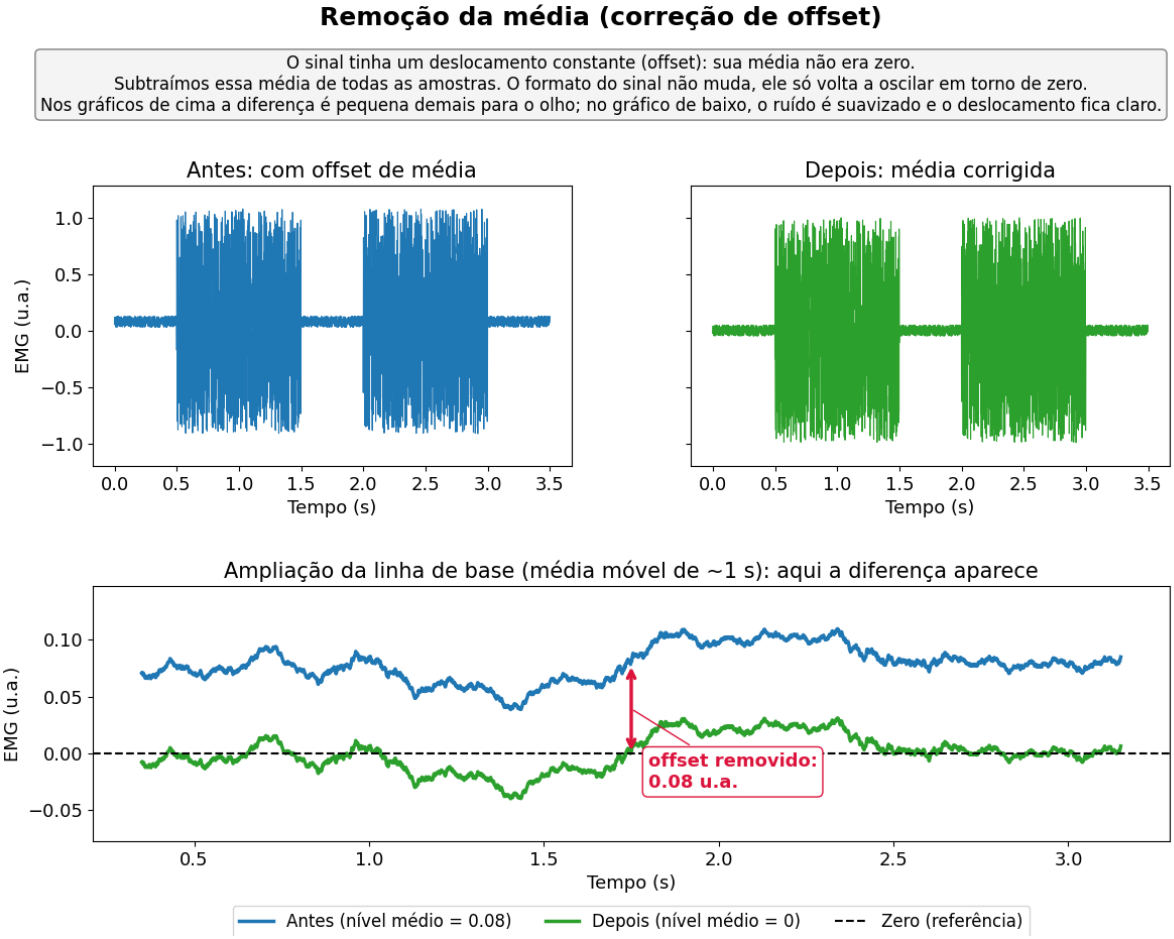
Figura 3 – Interface gráfica do sistema após a detecção de *onset*/*offset*, com o visualizador de figuras (navegação e botões de *download*) à esquerda e, à direita, os botões de cada etapa e as configurações do filtro.



Fonte: Elaboração própria dos autores (2026).

A aplicação da correção da média sobre o sinal EMG simulado resultou na remoção do *offset* DC, com o sinal passando a oscilar em torno de zero. A inspeção visual dos gráficos antes e depois da correção (Figura 4), incluindo o painel de média móvel que evidencia numericamente o valor do *offset* removido, confirmou o deslocamento da linha de base para o valor zero, conforme esperado. Esse comportamento é consistente com a literatura, que destaca a importância da remoção do *offset* para análises subsequentes (Ferreira; Guimarães; Silva, 2010; Wu *et al.*, 2022). A média do sinal corrigido foi numericamente próxima de zero, validando a implementação.

Figura 4 – Comparação do sinal EMG antes e após a correção da média, com painel de média móvel destacando o *offset* removido e a linha de base ajustada para zero.



Fonte: Elaboração própria dos autores (2026).

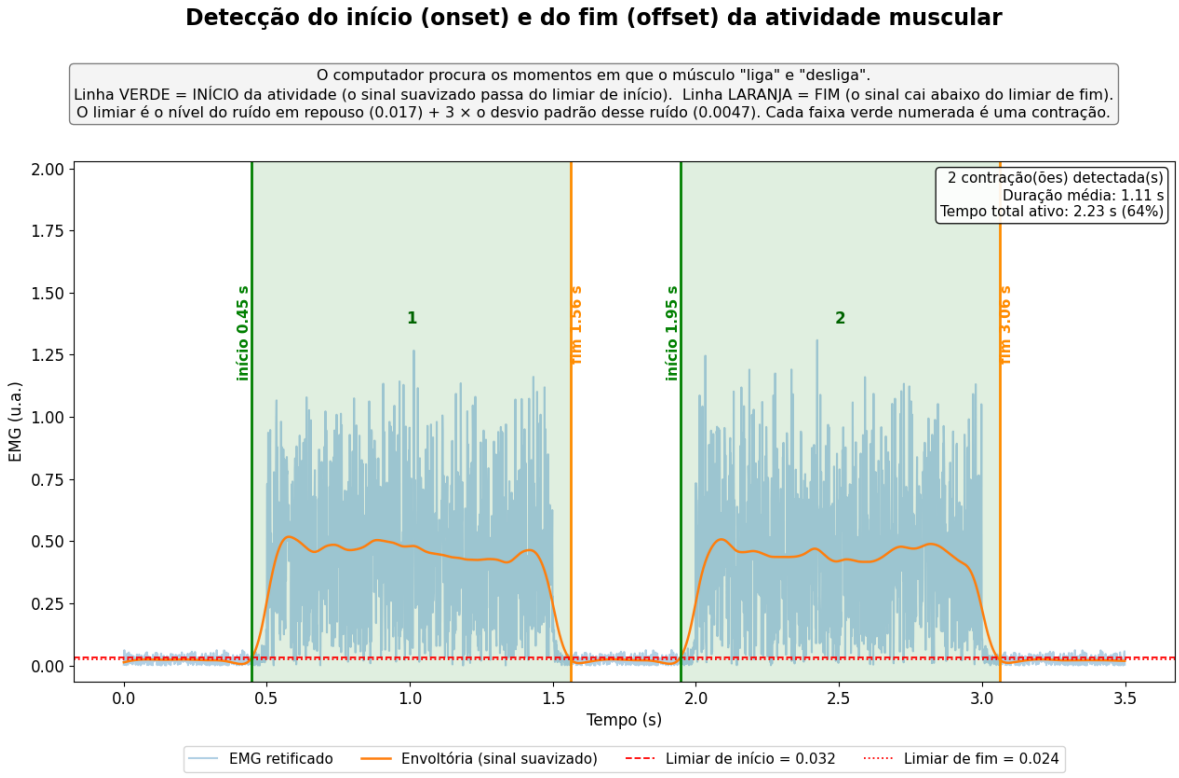
A filtragem foi aplicada utilizando o filtro Butterworth passa-banda de 4ª ordem com frequências de corte de 20 e 450 Hz. A inspeção visual do sinal filtrado no tempo e no espectro de frequências revelou a atenuação de componentes fora da faixa passante, com a região sombreada e as frequências de corte marcadas evidenciando o efeito do filtro. A utilização do *filtfilt* garantiu a ausência de distorção de fase, mantendo o alinhamento temporal dos eventos. Embora a validação tenha utilizado uma configuração específica, o sistema oferece ao usuário a flexibilidade de escolher entre diferentes tipos de filtro e ajustar seus parâmetros conforme a necessidade da análise.

A aplicação da retificação converteu o sinal bipolar em unipolar, preservando a informação de amplitude. A inspeção dos gráficos, incluindo a janela ampliada e as barras com o valor médio antes e depois da retificação, confirmou que os valores negativos foram convertidos para positivos e que a amplitude média deixou de ser próxima de zero, passando a refletir a intensidade da contração. Esse resultado está alinhado com o esperado para a

retificação de onda completa (Ferreira; Guimarães; Silva, 2010; Reaz; Hussain; Mohd-Yasin, 2006; Wu *et al.*, 2022), e a implementação utilizando np.abs() mostrou-se correta.

A detecção de *onset*/*offset* foi aplicada ao sinal retificado, resultando na identificação dos períodos de atividade muscular. A envoltória linear, obtida por filtragem passa-baixa a 6 Hz, permitiu uma representação suavizada da atividade, enquanto os dois limiares com histerese mostraram-se adequados para distinguir os *bursts* de atividade do ruído de fundo, evitando a oscilação em torno de um limiar único. Os pares *onset*/*offset* foram corretamente identificados, e a figura gerada (Figura 5) apresentou os períodos de atividade destacados e numerados, com a contagem de contrações, a duração média e o tempo total ativo. O critério de fusão e duração mínima de 100 ms eliminou eficazmente falsos positivos decorrentes de picos isolados de ruído. Esse resultado demonstra a eficácia do algoritmo implementado para a segmentação do sinal EMG em períodos de atividade e repouso.

Figura 5 – Detecção dos pontos de *onset* e *offset* dos *bursts* de atividade muscular, com a envoltória e os limiares de início e de fim sobrepostos e cada contração numerada.

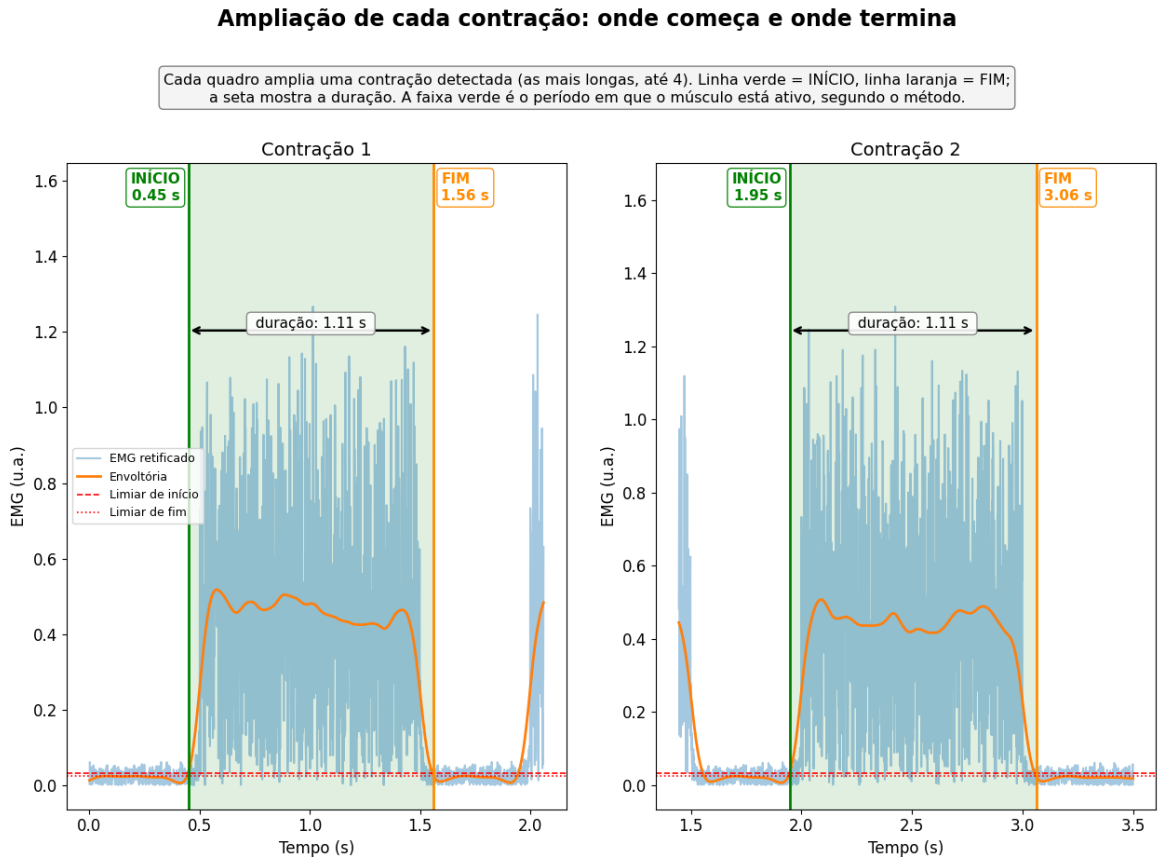


Fonte: Elaboração própria dos autores (2026).

Além das figuras que documentam cada etapa do *pipeline*, o sistema gera três figuras voltadas ao uso didático da ferramenta. A primeira (Figura 6) amplia até quatro das contrações detectadas, indicando a duração de cada uma. A segunda reúne, em um único

painel, os três passos do método de detecção (o sinal retificado, a envoltória e a decisão pelos limiares), permitindo acompanhar visualmente como o algoritmo chega ao resultado. A terceira apresenta uma janela ampliada com o sinal filtrado, a envoltória e os limiares sobrepostos, destacando a faixa de repouso e de atividade. Essas figuras complementam a Figura 5, detalhando o funcionamento do algoritmo de detecção para fins de ensino e experimentação.

Figura 6 – Ampliação de cada contração detectada, com indicação do início, do fim e da duração.



Fonte: Elaboração própria dos autores (2026).

Para complementar a validação com sinais simulados, foram utilizados dados EMG faciais adquiridos com o dispositivo MyoTec, considerando os músculos corrugador do supercílio e zigomático maior, associados a respostas emocionais. O *software* de aquisição do MyoTec gera arquivos no formato .edf, os quais foram convertidos para .txt utilizando o MATLAB. Esse arquivo .txt foi então carregado no sistema desenvolvido por meio da funcionalidade *load_txt_signal*, e a representação gráfica foi gerada. A plotagem obtida demonstrou que o sistema é capaz de carregar e representar graficamente sinais EMG reais a

partir do formato .txt, confirmando o funcionamento da etapa de entrada de dados da ferramenta proposta.

A validação do sistema foi realizada em duas frentes complementares. A primeira, com sinais simulados, permitiu verificar o comportamento das quatro etapas do *pipeline* de processamento (correção da média, filtragem, retificação e detecção de *onset*/*offset*) em condições controladas. A segunda, com dados externos, avaliou a capacidade do sistema de carregar e representar graficamente sinais EMG reais a partir de arquivos .txt obtidos pela conversão de arquivos .edf do MyoTec utilizando o MATLAB, confirmando a funcionalidade de entrada de dados.

Os testes funcionais manuais e automatizados foram executados com sucesso. A suíte de testes com *pytest*, composta por 25 testes, verificou as funções do módulo de processamento, incluindo a geração de sinal, correção da média, filtragem, retificação e detecção de *onset*/*offset*, bem como os componentes da interface; todos os 25 testes passaram na execução registrada. Os testes com entradas controladas produziram saídas compatíveis com os valores esperados, confirmando a correção das implementações. Os testes de interface validaram a navegação, a atualização dos gráficos e o controle de sequência, demonstrando que o sistema atende aos requisitos funcionais estabelecidos.

5 CONSIDERAÇÕES FINAIS

Este trabalho apresentou o desenvolvimento de um sistema computacional gratuito e de código aberto, denominado ArenaSignal, para processamento digital de sinais eletromiográficos, com interface gráfica voltada ao ensino e à experimentação. O sistema contempla as principais etapas de processamento de EMG descritas na literatura, incluindo correção da média, filtragem digital configurável, retificação de onda completa e detecção de *onset*/*offset*, organizadas em uma arquitetura modular que facilita a manutenção e a implementação de novas funcionalidades. A validação, realizada por meio de testes funcionais e automatizados com sinais simulados, demonstrou resultados compatíveis com o comportamento esperado para cada técnica. Adicionalmente, testes com sinais EMG faciais reais, adquiridos com o equipamento MyoTec e arquivos .edf convertidos para .txt utilizando MATLAB, confirmaram a capacidade do sistema de carregar e representar graficamente sinais reais.

Como limitações, destacam-se a predominância de sinais simulados na validação e a ausência de análises quantitativas, como RMS, SNR e correlação cruzada, além da falta de

comparações com ferramentas consolidadas e equipamentos comerciais. A utilização de sinais reais restringiu-se à verificação do carregamento e da representação gráfica, aspectos coerentes com a proposta educacional e experimental do sistema. Como trabalhos futuros, pretende-se ampliar a validação com sinais EMG reais, incorporar métricas quantitativas, expandir os formatos de arquivo suportados e implementar técnicas adicionais, como a normalização do sinal. Por fim, a disponibilização do código-fonte no GitHub contribui para a reprodutibilidade, a colaboração e o desenvolvimento de ferramentas abertas voltadas ao processamento de biossinais. Os autores agradecem à Fundação Engenheiro Salvador Arena pelo apoio institucional, bem como ao Núcleo de Pesquisas (NUPE) pelo suporte ao desenvolvimento deste projeto.

REFERÊNCIAS

ADLBI, Qussai. **EMG analysis engine (Module A)**: an open-source, IEEE/ISEK-compliant platform for reproducible EMG signal processing and feature extraction. [S. l.]: engrXiv, 2026. Pré-publicação. DOI: https://doi.org/10.31224/6627.

ARAUJO, Rubens Corrêa. **Utilização da eletromiografia em análise biomecânica do movimento humano**. 1998. Tese (Doutorado em Biodinâmica do Movimento Humano) – Escola de Educação Física e Esporte, Universidade de São Paulo, São Paulo, 1998. DOI: https://doi.org/10.11606/T.39.1998.tde-08112023-094433.

CHOWDHURY, Rubana H. *et al.* Surface electromyography signal processing and classification techniques. **Sensors**, [S. l.], v. 13, n. 9, p. 12431-12466, 2013. DOI: https://doi.org/10.3390/s130912431.

COLOMBET, Bruno *et al.* AnyWave: a cross-platform and modular software for visualizing and processing electrophysiological signals. **Journal of Neuroscience Methods**, [S. l.], v. 242, p. 118-126, 2015. DOI: https://doi.org/10.1016/j.jneumeth.2015.01.017.

CRESWELL, John W. **Projeto de pesquisa**: métodos qualitativo, quantitativo e misto. 3. ed. Porto Alegre: Artmed, 2010.

DUARTE, Marcelo Marques *et al.* Revisão sobre materiais e métodos: eletromiografia. **EFDeportes.com, Revista Digital**, Buenos Aires, v. 16, n. 160, set. 2011. Disponível em: https://www.efdeportes.com/efd160/materiais-e-metodos-eletromiografia.htm. Acesso em: 12 nov. 2026.

FERREIRA, Arthur de Sá; GUIMARÃES, Fernando Silva; SILVA, Julio Guilherme. Aspectos metodológicos da eletromiografia de superfície: considerações sobre os sinais e processamentos para estudo da função neuromuscular. **Revista Brasileira de Ciências do Esporte**, [S. l.], v. 31, n. 2, p. 11-30, 2010. Disponível em: https://www.rbceonline.org.br/index.php/RBCE/article/view/300. Acesso em: 12 nov. 2026.

GIL, Antonio Carlos. **Como elaborar projetos de pesquisa**. 4. ed. São Paulo: Atlas, 2002.

HALAKI, Mark; GINN, Karen. Normalization of EMG signals: to normalize or not to normalize and what to normalize to? In: NAIK, Ganesh R. (org.). **Computational**

intelligence in electromyography analysis: a perspective on current applications and future challenges. Rijeka: InTech, 2012. p. 175-194. DOI: https://doi.org/10.5772/49957.

HARRIS, Charles R. *et al.* Array programming with NumPy. **Nature**, [S. l.], v. 585, n. 7825, p. 357-362, 2020. DOI: https://doi.org/10.1038/s41586-020-2649-2.

HAUER, Arthur. **Desenvolvimento de um framework Python orientado à contribuição para análise e processamento de biossinais**: exemplo de aplicação para eletroencefalogramas. 2024. Dissertação (Mestrado em Engenharia Elétrica e Informática Industrial) – Universidade Tecnológica Federal do Paraná, Curitiba, 2024. Disponível em: http://repositorio.utfpr.edu.br/jspui/handle/1/36087. Acesso em: 12 nov. 2026.

HODGES, P. W.; BUI, B. H. A comparison of computer-based methods for the determination of onset of muscle contraction using electromyography. **Electroencephalography and Clinical Neurophysiology**, [S. l.], v. 101, n. 6, p. 511-519, 1996. DOI: https://doi.org/10.1016/S0013-4694(96)95190-5.

HUNTER, John D. Matplotlib: a 2D graphics environment. **Computing in Science & Engineering**, [S. l.], v. 9, n. 3, p. 90-95, 2007. DOI: https://doi.org/10.1109/MCSE.2007.55.

IEEE. **IEEE recommended practice for architectural description for software-intensive systems**. IEEE Std 1471-2000. New York: IEEE, 2000. Disponível em: https://standards.ieee.org/ieee/1471/2187/. Acesso em: 12 nov. 2026.

KLAIĆ, Luka *et al.* A systematic review of design of electrodes and interfaces for non-contact and capacitive biomedical measurements: terminology, electrical model, and system analysis. **Sensors**, [S. l.], v. 26, n. 4, art. 1374, 2026. DOI: https://doi.org/10.3390/s26041374.

KONRAD, Peter. **The ABC of EMG**: a practical introduction to kinesiological electromyography. Versão 1.4. Scottsdale: Noraxon, 2006.

KOO, Nahmil. Design of low power and low noise instrumentation amplifier for biopotential acquisition: a review. **IEEE Access**, [S. l.], v. 13, p. 23359-23370, 2025. DOI: https://doi.org/10.1109/ACCESS.2025.3537998.

MARCHETTI, Paulo Henrique; DUARTE, Marcos. **Instrumentação em eletromiografia**. São Paulo: Laboratório de Biofísica, Escola de Educação Física e Esporte, Universidade de São Paulo, 2006.

MELLO, Roger G. T.; OLIVEIRA, Liliam F.; NADAL, Jurandir. Digital Butterworth filter for subtracting noise from low magnitude surface electromyogram. **Computer Methods and Programs in Biomedicine**, [S. l.], v. 87, n. 1, p. 28-35, 2007. DOI: https://doi.org/10.1016/j.cmpb.2007.04.004.

MERLETTI, Roberto; MUCELI, Silvia. Tutorial. Surface EMG detection in space and time: best practices. **Journal of Electromyography and Kinesiology**, [S. l.], v. 49, art. 102363, 2019. DOI: https://doi.org/10.1016/j.jelekin.2019.102363.

MOORE, Alan D. **Python GUI programming with Tkinter**: develop responsive and powerful GUI applications with Tkinter. Birmingham: Packt Publishing, 2018.

MYERS, Glenford J.; SANDLER, Corey; BADGETT, Tom. **The art of software testing**. 3. ed. Hoboken: John Wiley & Sons, 2012.

PREISSEL, René; STACHMANN, Bjørn. **Git**: distributed version control: fundamentals and workflows. [S. l.]: Brainy Software, 2014.

PRESSMAN, Roger S.; MAXIM, Bruce R. **Software engineering**: a practitioner's approach. 9. ed. New York: McGraw-Hill Education, 2020.

PRODANOV, Cleber Cristiano; FREITAS, Ernani Cesar de. **Metodologia do trabalho científico**: métodos e técnicas da pesquisa e do trabalho acadêmico. 2. ed. Novo Hamburgo: Feevale, 2013.

PYTHON SOFTWARE FOUNDATION. **The Python language reference**. Python 3.12 Documentation, [s. d.]a. Disponível em: https://docs.python.org/3.12/reference/index.html. Acesso em: 12 nov. 2026.

PYTHON SOFTWARE FOUNDATION. **venv**: criação de ambientes virtuais. Python 3.12 Documentation, [s. d.]b. Disponível em: https://docs.python.org/pt-br/3.12/library/venv.html. Acesso em: 12 nov. 2026.

REAZ, M. B. I.; HUSSAIN, M. S.; MOHD-YASIN, F. Techniques of EMG signal analysis: detection, processing, classification and applications. **Biological Procedures Online**, [S. l.], v. 8, n. 1, p. 11-35, 2006. DOI: https://doi.org/10.1251/bpo115.

SHIELDS, JoCarol E. *et al.* A comparison of popular filters for the electromyography beginner. **Measurement in Physical Education and Exercise Science**, [S. l.], v. 30, n. 2, p. 257-266, 2026. DOI: https://doi.org/10.1080/1091367X.2025.2589872.

SILVA, Robson Rodrigues da *et al.* Ferramentas computacionais para estudo dos canais iônicos utilizando diagramas de estados: MATLAB® vs Python. **Revista Contemporânea**, [S. l.], v. 3, n. 6, p. 5013-5024, 2023. DOI: https://doi.org/10.56083/RCV3N6-018.

SILVA JÚNIOR, Rubens Alexandre da. Normalização EMG: considerações da literatura para avaliação da função muscular. **ConScientiae Saúde**, São Paulo, v. 12, n. 3, p. 470-480, 2013. DOI: https://doi.org/10.5585/ConsSaude.v12n3.4362.

SOMMERVILLE, Ian. **Engenharia de software**. 9. ed. São Paulo: Pearson, 2011.

SOMMERVILLE, Ian. **Software engineering**. 10. ed. Boston: Pearson, 2015.

VIRTANEN, Pauli *et al.* SciPy 1.0: fundamental algorithms for scientific computing in Python. **Nature Methods**, [S. l.], v. 17, n. 3, p. 261-272, 2020. DOI: https://doi.org/10.1038/s41592-019-0686-2.

WU, Tsung-Lin *et al.* pyemgpipeline: a Python package for electromyography processing. **Journal of Open Source Software**, [S. l.], v. 7, n. 72, art. 4156, 2022. DOI: https://doi.org/10.21105/joss.04156.