SEI-SICITE 2021
Pesquisa e Extensão para um mundo em transformação

XI Seminário de Extensão e Inovação
XXVI Seminário de Iniciação Científica e Tecnológica
08 a 12 de Novembro - Guarapuava/PR

UTFPR
UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
CAMPUS GUARAPUAVA

Software de processamento de eletromiografia de superfície em linguagem Python

Surface electromyography processing software in Python language

Guido Margonar Moreira *, Daniel Prado de Campos †

2021

RESUMO

Considerando a necessidade de uma ferramenta de fácil uso para a visualização e processamento de sinais de Eletromiografia de superfície(sEMG), sensores de sinais elétricos com eletrodos colados à pele, este projeto tem por objetivo pesquisar e desenvolver um software capaz dessas funcionalidades, para isso a linguagem de programação escolhida foi o python por conta de sua simplicidade e da familiaridade do desenvolvedor, o Ambiente de Desenvolvimento Integrado(IDE) utilizado foi o pycharm, as janelas do programa foram desenvolvidas no Qt Designer que exporta arquivos no formato .ui que podem ser lidos pela biblioteca pyqt5, após a leitura dos dados o programa realiza uma suavização utilizando o método de média fixa com um intervalo de 100 em 100 dados, o conteúdo é armazenado em um vetor e os procedimentos de análise são disponibilizados ao usuário, permitindo a detecção de sinais Verdadeiros-Positivos(onde houve pulso elétrico por conta da atividade muscular) tanto de maneira automática através do cálculo do limiar, ou manual com o usuário definindo onde os sinais se encontram marcando cada posição, por fim possibilitando a exportação dos dados no formato .txt, todas essas funcionalidades foram alcançadas demonstrando que o python poderá continuar sendo utilizado no desenvolvimento de outras funcionalidades desejadas por este projeto.

Palavras-chave: sEMG. python. pyqt5. qtdesigner.

ABSTRACT

Need the need for an easy-to-use tool for the visualization and processing of Surface Electromyography (sEMG) signals, electrical signal sensors with electrodes glued to the skin, this project aims to research and develop a software capable of these characteristics, for this the programming language chosen was python because of its simplicity and developer familiarity, the Integrated Development Environment (IDE) used for pycharm, as program windows were developed in Qt Designer that exports files in .ui format that can be read by the pyqt5 library, after reading the data the program performs a smoothing using the fixed average method with an interval of 100 out of 100 data, the content is stored in a vector and the analysis procedures are made available to the user, allowing the detection of True-Positive signals (where there was an electrical pulse due to muscle activity) either tom ic through the calculation of the threshold, or manual with the user defining where the signals marking each position, finally enabling the export of data in .txt format, all these characteristics were achieved demonstrating that python can continue to be used in the development of others desired by this project.

Keywords: sEMG. python. pyqt5. qtdesigner.

* Engenharia da Computação; guidomoreira@utfpr.edu.br; https://orcid.org/0000-0002-6480-8864.
† coenc-ap; danielcampos@utfpr.edu.br; https://orcid.org/0000-0001-6233-6077.

1 INTRODUÇÃO

Este projeto busca a criação de uma ferramenta simples e fácil para a análise de sinais de eletromiografia de superfície(sEMG), desenvolvida em 1944, por Inman, Saunders e Abbot, trata-se da obtenção de sinais elétricos produzidos pelo corpo humano sem a necessidade de uso de agulhas ou oferecer qualquer risco e incômodo ao sujeito de onde os sinais são coletados, a obtenção desses dados utiliza eletrodos de superfícies em direto contato com a pele e devido a essa simplicidade sem riscos o procedimento não requer licença médica para ser realizado, essas amostras podem ser utilizadas na verificação do comportamento neuromuscular em lesões e doenças, além de permitir o estudo de padrões musculares sem grandes dificuldades, assim este projeto busca descobrir como criar uma aplicação que permita a análise dos dados de sEMG de forma prática e dinâmica?.

Para responder esta pergunta foram analisadas as linguagens de programação com as quais os membros do grupo já possuíam experiência prévia, além das capacidades e limitações de cada uma, para a escolha foram consideradas a simplicidade e legibilidade do código, além da familiriade do desenvolvedor, isso levou a escolha do python, que pode ser estudado por meio de sites como STACKOVERFLOW. . . (2008), e com base do livro (MUELLER, 2020), assim este projeto busca descobrir se esta linguagem permitirá as funções básicas que o programa virá a executar como leitura, processamento e visualização dos sina is de sEMG, usando algumas das funções em (ALEXANDRE, 2009), e como o desenvolvimento de sensores físicos foi adiados por conta da pandemia os sinais nos exemplos vieram da base de dados MEGANEPRO. . . (2016).

2 MÉTODO (OU PROCEDIMENTOS OPERACIONAIS DA PESQUISA)

Para a criação do programa foi escolhida a linguagem de programação python versão 3.6, criado originalmente em 1989 por Guido Van Rossum, o python visa a simplicidade e organização do código, facilitando a leitura e entendimento por parte do programador, foi escolhida devido a familiaridade com a linguagem, e facilidade no aprendizado das funcionalidades básicas,a IDE utilizada para criação e compilação dos códigos foi o pycharm(que possui licença gratuita para estudantes), plataforma criada em 2010, permite fácil instalação de bibliotecas de forma automática e dinâmica, e pré compila os códigos detectando erros na escrita, a IDE foi instalada diretamente no computador pessoal do desenvolvedor que fez tudo de maneira digital sem sair de casa começando no final de 2020.

A criação da interface foi possível através do uso da biblioteca pyqt5 versão 5.12.3, que permite a criação da janela por meio das funções na Fig. 1, após a inicialização da janela a interface é carregada pela função **loadui('nomeArquivo.ui')**, e por fim o botão **bt_abrirArq** é associado a função **openArq** responsável por ler o nome digitado pelo usuário e tentar abrir um arquivo no formato requisitado.

Para facilitar o desenvolvimento das interfaces que o programa viria a utilizar foi escolhida a ferramenta Qt designer, nela é possível criação de janelas de maneira intuitiva e interativa, arrastando os widgets de uma lista e em seguida ajustando suas propriedades diretamente na interface, depois os botões, labels, caixas de texto, e demais ferramentas estão nomeadas e nas posições desejadas, basta salvar o arquivo no formato **.ui** que como anteriormente citado pode ser carregado pela biblioteca pyqt.

Como os sensores para coleta em tempo real ainda estão em desenvolvimento os dados analisados pelo programa foram baixados de bancos de dados da internet, a análise dos arquivos no formato **.mat** foi feita com a biblioteca **scipy**,versão 1.5.4, foi quem permitiu por meio da função **loadmat('nomeArquivo.mat')**, dessa maneira os dados são armazenados dentro de uma variável do tipo dictionary, já para leitura do formato **.csv**

Figura 1 – Função criando a janela principal

```
# Janela do programa
class MatplotlibWidget(QMainWindow):
    #Inicializa janela do programa
    def __init__(self):
        QMainWindow.__init__(self)

        #Define qual tela esta sendo mostrada
        self.m = "B"

        #Carrega UI para abrir arquivos
        loadUi("AbrirArquivos.ui", self)

        #Associa o botão bt_abrirArq com a função openArq
        self.bt_abrirArq.clicked.connect(self.openArq)
```

Fonte: Autoria própria (2021).

foi a biblioteca **pandas** usando a função **read( )csv('NomeArquivo.csv', delimiter='<Simbolo utilizado para delimitar dados no arquivo>')** que por fim salva os dados em uma matriz. Independente do formato original, os dados armazenados agora podem ser facilmente manipulados pelo programa e salvos em um vetor para facilitar as operações matemáticas. Com os dados na sua forma bruta armazenados pelo processador faz-se necessária a

Figura 2 – Lendo dados arquivo .mat

```
#Abrir arquivos .mat de nome armazenado na string Arq
def openMAT(Arq):
    #Arquivo
    file = sio.loadmat(Arq)
```

Fonte: Autoria própria (2021).

suavização por meio da Retificado seguida de uma média que pode ser fixa ou móvel de acordo com a escolha do usuário, na média fixa as amostras são divididas e é tirada a média resultando em um sinal bem menor que o original Eq. (1), já na média móvel o arredondamento é feito através da convolução uma entrada de 100 impulsos, começando com zero de deslocamento indo até o -99, com o dado bruto, seguindo a fórmula Eq. (2), quando essa convolução tenta-se interagir com um sinal fora do vetor, é considerado o valor $Sinal[i] = 0$, assim o vetor **suavizado[]** resultante possui uma quantidade de sinais a mais dependente do valor da variável **Ared** que define quantos dados são usados para cada média aritmética.

$$Suavizado[k] = \sum_{i=k*100}^{\text{k*100+100}} \frac{Sinal[i]}{100}, K \in [0, \frac{TamanhoVetor}{100}] \tag{1}$$

$$Suavizado[k] = \sum_{i=k-99}^{\text{k}} \frac{Sinal[i]}{100}, K \in [0, TamanhoVetor + 99] \tag{2}$$

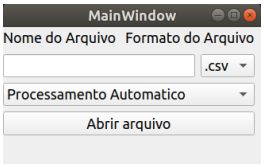
Com os sinais suavizados calculados o programa pode facilmente acessar os resultados no vetor para realizar o processamento desejado pelo usuário, no caso para a detecção automática de ocorrências de Verdadeiros positivos é necessario o uso da formula de Eq. (3) para cálculo do limiar usando ruído de base, a média aritmética é realizada com valores que o programa tenha adquirido em momentos de Verdadeiros Negativos, assim com o valor apropriado para k o programa consegue traçar uma reta para diferenciar ruído de sinais Verdadeiros Positivos, assim valores de módulo maior que o limiar são considerados possíveis sinais.

$$Limiar = \mu + \sigma * k \tag{3}$$

3 RESULTADOS

A interface então foi criada totalmente no qt designer, a primeira janela que é aberta com a execução do código Fig. 3 contém uma caixa de texto para que seja digitado localização e nome do arquivo, ao lado um widget do tipo combobox permite que seja definido o formato do arquivo que será aberto(.mat, .csv e .txt), logo em baixo se encontra outra combobox que permite o usuario escolher entre "Processamento automático"e "intervalos manuais", por fim o botão **bt_abrirArq** onde está escrito "Abrir Arquivo", quando ele é pressionado a função **openArq** é executada, as configurações selecionadas são chamadas pelo programa que tenta abrir o arquivo na localização indicada, caso algum erro ocorra durante o processo(como por exemplo o nome do arquivo ser digitado corretamente) o programa simplesmente cancela a execução da função e se mantém na tela para abertura do arquivo.

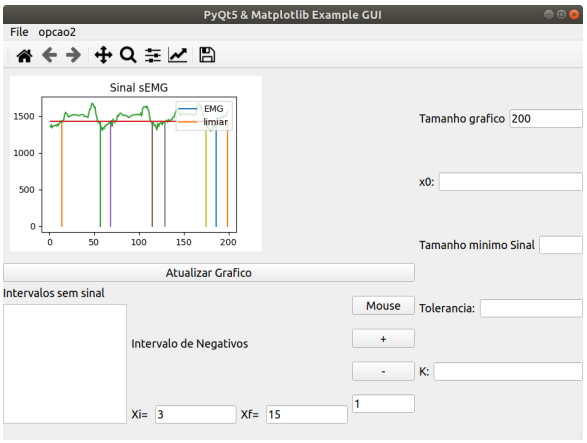
Figura 3 – Janela para abertura de arquivos



Fonte: Autoria própria (2021).

Caso o arquivo seja aberto com sucesso os dados do primeiro sensor é armazenado em um vetor, e o cálculo da média fixa e realizado e salvo no array **suavizado**, a partir dai o programa inicializa uma nova janela, carregando a interface e associando as funções aos respectivos botões, no caso do processamento automático a janela aberta é Fig. 4.

Figura 4 – Janela exemplo processamento automático



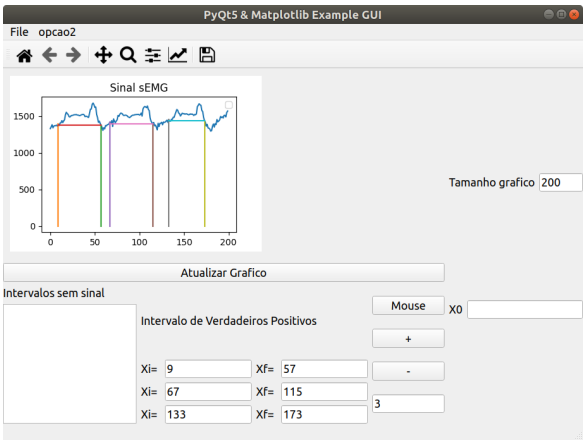
Fonte: Autoria própria (2021).

Na coluna direita existem vários labels(textos) acompanhados de Widgets QlineEdit(caixas de texto) para serem preenchidas, elas servem para preencher os valores de variáveis usadas pelo programa no processamento, caso algo invalido seja digitado ou sejam deixados vazios o programa simplesmente utiliza um valor default, os dados são: **tamanho gráfico**, que determina quantos dados serão mostrados no gráfico que vai aparecer na interface, **x0** que determina de qual posição será obtido o primeiro valor do gráfico, **tamanho mínimo sinal**, caso um sinal se mantenha acima do valor do limiar um número de vezes menor que o valor definido nesse limite o sinal é considerado Falso Positivo, a **Tolerancia** representa o número máximo de vezes que um sinal pode

voltar para baixo do limiar sem que seja considerado como finalizado, e por fim o **K** é usado diretamente na Eq. (3) para cálculo do limiar.

No caso de a abertura por intervalos manuais ser escolhida a janela aberta é Fig. 5, Bastando serem definidos os valores de posição inicial(X0) e a quantidade de dados que serão mostrados(Tamanho gráfico)

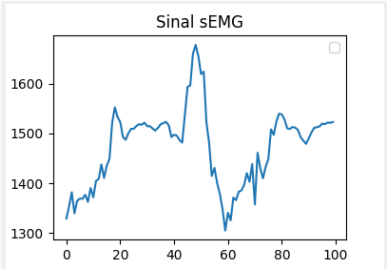
Figura 5 – Janela exemplo de Intervalos manuais



Fonte: Autoria própria (2021).

Em ambos os casos para a visualização dos dados foi utilizado o widget matplotlib (Fig. 6), ele é atualizado sempre que o botão "Atualizar gráfico"é pressionado, a manipulação da informação que aparece no dado é acessada por meio da referência **self.MplWidget.canvas.axes** seguida pelo comando desejado **plot(x,y)** para exibir os gráficos dos vetores, **legend** e **set_title** permitem modificar algumas das configurações de exibição, por fim **draw** mostrava tudo no widget.

Figura 6 – Widget para visualização de gráficos

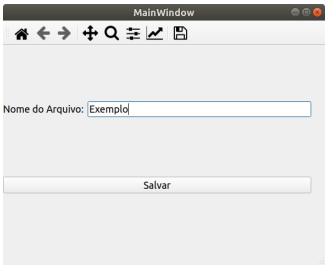


Fonte: Autoria própria (2021).

Para obtenção de intervalos de dados que o programa utiliza nas operações matemáticas, Verdadeiros Negativos e Verdadeiros Positivos, para Processamento Automático e intervalos Manuais respectivamente, foi utilizada a estrutura na centro inferior de Fig. 5, onde o botão **+** adiciona um widget contendo caixas de texto e labels onde está escrito **Xi** e **Xf** permitindo que o usuário digite os intervalos que serão processados de acordo com o modo de abertura, para deletar um intervalo já existente basta digitar o índice(de cima para baixo) na caixa de texto no canto inferior direito e em seguida apertar no botão **-**, o botão **Mouse** permite que o usuário clique diretamente no gráfico para definir o Xi e Xf com o primeiro e segundo cliques, os valores são transferidos para o intervalo com o Índice na caixa de texto, Sempre que alterações são feitas aos intervalos o gráfico é atualizado Marcando os intervalos diretamente no caso de "Intervalos Manuais"na Fig. 5 e usando eles para cálculo do limiar Eq. (3) no "Processamento Automático"na Fig. 4.

Com todos os parâmetros definidos e a análise dos dados completa o programa permite que o usuário exporte as informações adquiridas em um arquivo de formato .txt, para realizar isso é necessário posicionar o mouse sobre a opção File que se encontra na extremidade superior direita da janela e em seguida clicar na opção "exportar", o programa então abre a janela Fig. 7, Em seguida o usuário pode pressionar o botão **Salvar** e um arquivo com o digitado pelo usuário seguido da sigla EMG será criado com o formato txt, dentro dele se encontram todas as posições do vetor em que ocorrem intervalos com e sem sinal.

Figura 7 – Janela Exportar Arquivo



Fonte: Autoria própria (2021).

4 CONCLUSÕES

Com base nos resultados obtidos foi possível concluir que o python foi uma escolha eficiente sendo capaz de realizar a maioria das tarefas buscadas pelo projeto, como leitura dos dados, aplicação de filtros, processamentos de sinais, visualização por gráficos e exportação das informações analisadas, isso significa que o desenvolvimento mais a frente do programa pode continuar no python sem maiores complicações, sendo o processamento de sinais em tempo real um dos futuros objetivos deste projeto, embora a interação entre o python e a linguagem da placa ainda não tenha sido iniciado, mas a manipulação de arquivos alcançada já consegue garantir que de uma maneira ou outra a interação será possível.

AGRADECIMENTOS

Agradeço a Fundação Araucária por fornecer a bolsa para o desenvolvimento deste projeto, ao meu Orientador Daniel sempre auxiliando na busca de soluções dos problemas durante o desenvolvimento e a UTFPR pela oportunidade.

REFERÊNCIAS

ALEXANDRE, N. J. **Introdução ao Processamento Digital de Sinais. Um foco no desenvolvimento de aplicações**. [S.l.]: Grupo GEN, out. 2009. ISBN 9788521626152. Acesso em: 13 out. 2021.

MEGANEPRO. Set. 2016. Disponível em: . Acesso em: 21 set. 2021.

MUELLER, J. P. **Começando a Programar Em Python Para Leigos 2Ed**. [S.l.]: Editora Alta Books, out. 2020. ISBN 9786555202298. Acesso em: 16 out. 2021.

STACKOVERFLOW. Set. 2008. Disponível em: . Acesso em: 21 set. 2021.