Tele-reabilitação do movimento da mão através de um jogo (HandsOn!) baseado em sinais EMG

Laboratório Integrado em Engenharia Biomédica 2020/2021

Diogo Valente Polónia (201806473), Isabela Marques (201806175), Maria Inês Barros (202003309)

Resumo - O eletromiograma avalia o movimento muscular, através de diferenças de potencial entre dois canais musculares. O objetivo deste trabalho foi criar uma plataforma interativa e gamificada, controlada pelo movimento da mão, para jovens crianças com dificuldades motoras ou em necessidade de fisioterapia e em reabilitação. Neste sentido, foi desenhada uma estratégia de processamento em *Matlab* baseada numa base de dados [1] de sinais digitais correspondentes à repetição do mesmo movimento de aperto um número pré-determinado de vezes. O processamento analógico foi primeiro simulado em *Multisim*, e de seguida implementado numa *breadboard* ligada a um *Arduino Uno* para a simples recolha e conversão A/D. O sinal digital resultante foi processado em *Matlab* mas devido à falta de rapidez na comunicação entre *Matlab* e *Java*, o processamento teorizado para *Matlab* foi substituído por um código de *Python* mais simples porém capaz de comunicar eficientemente com a interface desenvolvida em *Java*. Esta interface permite ao utilizador jogar um pequeno jogo estilo *arcade*, controlando a movimentação de um avatar através do teclado do computador e o disparo de projéteis através do fecho da mão. Deste modo, acumula-se pontos caso os projéteis atinjam os inimigos automaticamente gerados na interface. Para gerir e planear o projeto, 4 ferramentas foram utilizadas – *UML*, *Pencil*, *Redmine* e *Google Drive* –, e para o desenvolvimento de um logótipo, o *Canva*. Foi possível implementar, de forma simples e robusta, um sistema de controlo digital por sinal de eletromiografia (EMG), para fisioterapia, mas que poderá ser usado em diversas aplicações mais complexas, nomeadamente nas áreas de *e-sports*, simulação militar ou qualquer outra que envolva movimentos repetitivos e minuciosos e que possa, portanto, beneficiar da prática destes em ambiente seguro e simulado.

Palavras-Chave – Eletromiografia, sinal, processamento, tele-reabilitação

I. INTRODUÇÃO

A eletromiografia (EMG) é uma técnica importante para registar os potenciais de ação dos músculos. Estes sinais são controlados por estímulos provenientes de neurónios motores, que fazem parte do sistema nervoso central (SNC). Os estímulos são conduzidos pela medula espinhal até às fibras musculares que, por sua vez, provocam a contração do músculo [2]. Sendo uma representação das propriedades anatómicas e fisiológicas dos músculos, os registos de EMG podem ser usados para diagnosticar lesões musculares, lesões nervosas e disfunções musculares que ocorrem devido a distúrbios neurológicos e musculares [3].

A sua importância é reconhecida não apenas na área médica, mas também na engenharia. O estudo desta técnica possibilita, por exemplo, o desenvolvimento de aparelhos de reabilitação motora para fisioterapia e próteses robóticas.

Os sinais EMG encontram-se na ordem dos milivolts e com frequências entre os 2 aos 500 Hz. Estes são medidos usando elétrodos que podem ser elétrodos de agulha ou de superfície (ou de pele). Como os primeiros são considerados um método invasivo, os segundos, não invasivos, tem a vantagem de não necessitar de anestesiar o paciente antes da colocação do elétrodo tornando a aplicação dos sensores simples e indolor [2].

Neste projeto é apresentada uma proposta de utilização de eletromiografia num jogo, maioritariamente direcionado para crianças, cujo objetivo é de motivação à fisioterapia da mão. Assim, utilizando movimentos da mão definidos e prescritos para a sua reabilitação, o paciente poderá efetuar esses movimentos tendo o jogo como incentivo.

Para tal, utilizando elétrodos de superfície, sinais de eletromiografia foram recolhidos, filtrados e amplificados. Posteriormente foi realizado o processamento em *MatLab/Python* e a interface em *Java*.

II. METODOLOGIA

A. Materiais

Sendo o objetivo do projeto a recolha de sinal de EMG e posterior conversão A/D, para que possa ser usado para controlar o jogo, foi projetado e implementado um circuito analógico em laboratório, para o processamento analógico do sinal fisiológico.

A captação do sinal EMG foi efetuado por três elétrodos de superfície que permitem a captação dos potenciais de ação e transformá-los em impulsos elétricos. Posteriormente, estes são amplificados duas vezes, uma anteriormente à filtragem e outra posteriormente à mesma. O esquemático do circuito proposto foi projetado no *Multisim* e encontra-se na figura 1. Seguidamente, para o seu processamento foi utilizado *MatLab* (teoria) e *Python* (prática).

Por fim, a interface foi implementada em *Java* (SDK 16) recorrendo ao *Eclipse IDE*.

B. Planeamento do projeto

1. Estrutras de Acompanhamento e Gestão

O grupo foi estabelecido por 3 estudantes que, desde uma fase inicial, procuraram distribuir as tarefas de forma adequada entre cada membro e tomar decisões em concordância de todos, começando pela escolha do tema, de modo a garantir a criação conjunta do projeto em questão.

Para tal, foram utilizadas ferramentas como a *Google Drive*, para facilitar e acompanhar o cumprimento dos encargos propostos a cada um e o progresso do trabalho. A equipa reuniu regularmente numa conferência do *Google Meet*, devido à impossibilidade de reuniões presenciais.

Existiram reuniões semanais (além das previstas em aula) na terça-feira de tarde, a hora acordada, para fazer um ponto de situação do trabalho e definir próximos passos, assim como assegurar o cumprimento do cronograma. A tabela 1 contém os membros do grupo e respetivas funções.

Nome	Funções
Diogo Valente Polónia	Planeamento de tarefas; Modelação do projeto; Idealização e implementação da interface
Isabela Marques	Pesquisa e documentação do processamento digital; Implementação em *Matlab*
Maria Inês Barros	Pesquisa e documentação do processamento analógico; Implementação do circuito

Tabela 1 - Funções dos elementos do grupo

Visando a planificação e organização do trabalho a desenvolver para a implementação do projeto em causa, apresenta-se, no Anexo 1, um Diagrama de Gantt. Este ilustra as tarefas realizadas, já ajustadas às alterações de datas que se realizaram no decorrer do projeto.

2. Planeamento Global do Projeto em UML

Para melhor projetar a comunicação entre as ferramentas escolhidas, os requisitos e funcionalidades desejadas, bem como a lógica base do projeto, considerou-se importante desenvolver um diagrama de atividade, em linguagem *UML*, permitindo um melhor planeamento dos métodos a usar para alcançar todos os requerimentos de desenvolvimento (Anexo 2).

C. Captação do sinal: Circuito analógico

1. Colocação dos elétrodos

Uma vez que o sinal de EMG que se pretendia captar correspondia ao movimento da mão a abrir e fechar, os elétrodos foram colocados no interior do antebraço. Este era proveniente do músculo flexor superficial dos dedos, cuja função principal é de flexão das articulações metacarpofalângicas e interfalângicas do segundo ao quinto dedos.

2. Primeiro andar: Amplificação

Para o primeiro andar e por se tratar de um sinal bastante pequeno, o sinal foi primeiramente amplificado num amplificador de instrumentação INA126 permitindo-nos ter um ganho e impedância de entrada elevados. O ganho, dado pela expressão (1), foi estabelecido em 175.2 V/V, colocando a resistência R_G com um valor de 470Ω.

$$A_d = 5 + \frac{80k\Omega}{R_G} \qquad (1)$$

3. Segundo andar: Filtro passa baixo

De seguida, e considerando que as frequências dos sinais provenientes dos elétrodos não têm frequência superior a 500Hz, foi implementado um filtro passa baixo com ganho unitário. A frequência de corte do filtro, dada por

$$f_H = \frac{1}{2\pi\sqrt{R_1 R_2 C_1 C_2}} \qquad (2)$$

foi estabelecida nos 500Hz. Recorrendo à plataforma *Filter Wizard*, o filtro apresentado na figura 2 foi implementado.

4. Terceiro andar: Filtro passa alto

Após o segundo andar, o sinal segue para o filtro passa alto cuja frequência de corte, expressa na equação (2), foi imposta em 2Hz. Este encontra-se representado na figura 3.

Ambos os filtros são de segunda ordem do tipo *Sallen Key*.

De acordo com a expressão e substituindo os valores de acordo com o esquemático obtido através da plataforma *Filter Wizard*, as frequências de corte tomam os valores de, aproximadamente, 486Hz e 1.9Hz para o filtro passa baixo e filtro passa alto, respetivamente. Já que o circuito proposto pela plataforma está condicionado em termos das séries E12 – para as resistências – e E6 – dos condensadores – disponíveis, os valores adotados na prática acabam por não concretizar a frequência de corte teórica. Ainda assim, o valor considerado foi uma boa aproximação com vista a concretização do propósito da criação destes filtros.

5. Quarto andar: Amplificação

Por fim, e uma vez que o sinal filtrado perde alguma da sua amplitude, para melhor tratamento e visualização deste, foi novamente implementado um andar de amplificação. Desta vez o amplificador implementado foi não inversor e foi imposto um ganho de 22 V/V.

D. Processamento

O tratamento do sinal tem como objetivo permitir a deteção da realização do movimento. Para isto, escolheu-se uma base de dados onde o número de vezes que o movimento era realizado já estava pré-estabelecido. A base de dados utilizada, *sEMG for Basic Hand movements Data Set* [1], inclui diferentes apertos de mão. A frequência de amostragem do *dataset* é de 500Hz. Este também passou por um filtro passa banda analógico com frequências de corte de 15Hz e 500Hz e um filtro *notch* a 50Hz para eliminar artefactos da linha de energia.

Os dois canais apresentados no Anexo 3 pertencem a um sujeito masculino realizando o movimento "cilíndrico" 100 vezes, isto é o movimento de pegar um objeto cilíndrico.

A análise teórica realizada em *MatLab* consiste, primeiramente, em abrir o canal certo e obter o valor absoluto do sinal, depois de remover qualquer possível *offset* através da remoção da média do sinal original. A seguir, um filtro passa baixo *Butterworth* foi aplicado, onde a frequência de corte desejada é de 20Hz, esta ajustada para cima em 25% uma vez que o filtro será passado duas vezes (de frente para trás e de trás para frente). Esse ajuste assegura que com a atenuação de frequência de -3dB, após passarmos duas vezes, teremos a frequência de corte desejada. O valor de 20Hz foi escolhido pois mantém as frequências mais relevantes para o processamento pretendido. A frequência de amostragem é dividida por dois para termos a frequência de *Nyquist*, que será usada na construção dos filtros [4].

Um filtro digital *notch* a 50Hz também foi inserido no código, porém como o *dataset* analisado possuía esta frequência filtrada analogicamente, não houve praticamente nenhum efeito no sinal analisado.

Depois foi aplicado um filtro de média móvel com uma janela de 200, obtendo o último gráfico de ambas as imagens do Anexo 3. Após o filtro, o sinal aparece mais estável e com menos ruído em comparação ao sinal anterior, uma vez que esta função previne o ruído instantâneo. O valor da janela tem de ser ideal. Se este é muito pequeno a intensidade do sinal flutua muito e é difícil obter características em cada medição, se é muito grande, o sinal fica atenuado e sua forma original é perdida [5].

A última etapa é obter os picos, onde cada um representa a realização do movimento. Para isto foi utilizado a função *islocalmax* no *MatLab*, que encontra o máximo local. Entretanto, um parâmetro que precisou ser ajustado foi a *Minprominence*, que mede o quanto um pico se destaca devido a sua altura intrínseca e sua localização em relação a outros picos. Um pico baixo isolado pode ser mais proeminente que outro de valor maior, se o pico mais alto for membro de uma sequência de valores maiores onde este não se destaca. O valor da proeminência é ajustado para cada canal experimentalmente.

Com esse processamento foi identificado 101 (um falso positivo) no primeiro canal e 100 picos no segundo. Com isto podemos perceber que, no geral, a estratégia de processamento é eficiente e precisa em identificar quando o movimento está a ser realizado.

E. Interface

1. Modelação e Desenvolvimento

a) Planeamento em Pencil

A interface em *Java* partiu de um primeiro esboço em *Pencil*, que foi desenvolvido com base nos 5 princípios do *Design*, garantindo que todos os componentes de interação são visíveis (*1. Visibility*), que o resultado de todas as ações são óbvias e prontamente observáveis (*2. Feedback*), que são impostas limitações às interações para prevenção de erros (*3. Constraints*), que o desenho, cores e funcionalidades são consistentes e fáceis de aprender (*4. Consistency*), e por fim, que todos os atributos dão prioridade à facilidade de compreensão da sua funcionalidade, face à estética (*5. Affordance*).

Além disso, sendo o jogo destinado a jovens crianças, foi importante utilizar formas juvenis e conhecidas pelo que se adaptou, por exemplo, os avatares do popular jogo "*Among Us*". No Anexo 4, é possível comparar as interfaces desenvolvidas em *Pencil* e o resultado final em *Java*.

b) Desenvolvimento em Java

Com base no modelo em *Pencil*, e recorrendo ao *plug-in WindowsBuilder*, foram criadas 2 *frames*, uma correspondente ao menu inicial, o qual permite iniciar o jogo, sair da aplicação e alterar as definições, abrindo a 2ª *frame* correspondente à janela que permite fazer o *reset* ao *highscore*, alterar as dimensões da tela de jogo e alterar a dificuldade. Quando o utilizador clica no botão para iniciar uma tentativa na *frame* principal, é criado um objeto da classe *Game*, que gera uma nova *Jframe*, implementada sem recurso ao *WindowsBuilder* e inicia a lógica do jogo. Esta *Jframe* permite não só visualizar as componentes em movimento (o avatar que corresponde à mão do utilizador), as balas disparadas e os inimigos, mas também contém uma barra com a vida do jogador, o seu *score* atual, e o *highscore*, bem como as instruções mais relevantes (por exemplo a instrução de pressionar o a tecla "M" para voltar ao Menu).

3. Comunicação Processamento <> Interface

Após a inicialização do modo de jogo, o procedimento *signalFileListener* procura o ficheiro *signal.txt* na pasta do projeto, 60 vezes por segundo, i.e., a cada *tick* (procedimento explicado mais à frente). Assim, quando o gesto de encerramento do punho é detetado, o *script* de *Matlab* (ou, no caso da implementação real, de *Python*) cria o ficheiro *signal.txt*, prontamente detetado pelo *Java*. Este cria um novo objeto *Bullet* na *frame* do jogo, e apaga, de seguida, o ficheiro, permitindo a comunicação em tempo real entre as várias componentes do projeto e ainda que o "disparo" digital ocorra instantaneamente ao encerramento do punho do utilizador.

4. Adpatação do Processamento

O processamento digital desenvolvido em *Matlab*, embora eficaz na identificação das ocorrências do movimento através da análise do sinal após a recolha completa do mesmo, revelou-se ineficiente para aplicação em tempo real do modelo. Isto deve-se quer graças ao tempo demorado que leva a processar o sinal quer pelas dificuldades que impõe na comunicação com outros sistemas comparativamente a outras soluções como *Python*.

Este problema foi identificado nas últimas fases do trabalho desenvolvido, apenas após a conclusão da construção do circuito físico e da integração das várias componentes que compõem o projeto, pelo que se optou, para efeitos práticos, substituir o modelo criado em *Matlab* por um *script* de *Python* (*signalFileCreator.py)* baseado numa técnica diferente de processamento, bastante mais simples e, portanto, mais suscetível a erros, mas, ainda assim, suficiente para obter os resultados pretendidos.

O *script* lê diretamente os valores de tensão impressos pelo *Arduino* no *Serial Port* de comunicação e recorre à identificação de picos no sinal acima de um *threshold* (3.5V) determinado experimentalmente. Para ultrapassar a oscilação do sinal adquirido, sem recorrer a métodos mais complexos de processamento, por motivos de gestão de tempo, optou-se por desenvolver um algoritmo que contabiliza o número de picos (*NOnsets*) que ocorrem antes de serem detetados 5 (*NOffsets*) sinais seguidos abaixo do *threshold* (caso isso aconteça *NOnsets* é reposto a 0). Quando *NOnsets* = 5, o *script* envia o comando de criação do ficheiro *signal.txt* que origina o disparo em *Java*. Os critérios para *NOnsets* e *NOffsets* foram obtidos por experimentação, de tal forma a minimizar o erro obtido e nos testes efetuados, o movimento de fecho de mão foi reconhecido e originou o disparo corretamente na grande maioria vezes. Na figura A é possível analisar o fluxograma correspondente ao algoritmo utilizado.

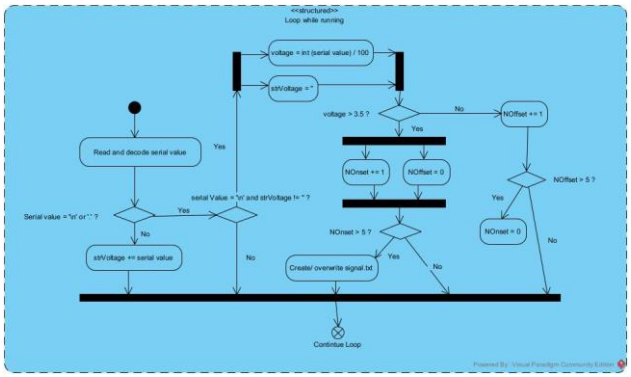


Figura A - Fluxograma algoritmo processamento sinal em Python

2. Recursos e Dependências

a) Composição do Java Project

A pasta do projeto é composta por uma pasta *res*, que contem todos os recursos externos utilizados, desde as imagens, os ficheiros *readAnalogVoltage.ino* que controla o *Arduino*, o ficheiro que guardas as definições (*settings.txt*). Já o ficheiro gerado pelo script de *Python* para sinalizar ao *Java* o comando de disparo (*signal.txt*) e o ficheiro *signalFileCreator.py* estão na pasta de projeto. Está também presente uma pasta *src*, que contém todos os ficheiros *.java* dentro do pacote criado *com.game.src.main* que, por sua vez, contém as classes utilizadas. Estas incluem a classe principal *frmMain* que controla todo o projeto, e o pacote *entities* que abarca as *interface classes*, correspondente aos 2 tipos de entidade existentes (A e B) implementadas pelas classes *Player*, *Enemy* e *Bullet*, para gerir o sistema de colisão. Objetos da mesma entidade (A: *Player* e *Bullet*; B: *Enemy*) não colidem entre si, ou seja, o jogador não é afetado por uma bala, mas a objetos de entidades diferentes podem colidir.

Foi necessário criar um novo *class path* para se poder referenciar a pasta *res*, e foram utilizadas várias bibliotecas do sistema *Java*, que podem ser encontradas no Anexo 5.

3. Lógica e Algoritmia

A classe *Game* contém a função *start()* e *stop()* que começam e terminam a *thread* na qual se dá o processamento do jogo. A função *init()* responsável pela inicialização do objeto *Game* e das suas propriedades e o procedimento *run()*, no qual está inserido o ciclo principal do jogo (*Game Loop*), enquanto ciclo *while true*. Para se iniciar o jogo, é necessário, após a criação do objeto, iniciar a *thread* (*game.start()*). Esta função chama o procedimento *run()* que inicializa o jogo e dá inico ao *Game Loop*. O *loop* é a componente mais importante de qualquer jogo estabelecendo os tempos de controlo (*tick*) e de renderização das imagens (*render*). Seguindo a prática comum, estabeleceu-se uma frequência de execução de 60 vezes por segundo (através da colocação de *timers* dentro do *Game Loop*) e não se impôs restrições à renderização, ou seja o procedimento *render()* é chamado a cada iteração do ciclo pelo que depende apenas da capacidade de processamento do dispositivo utilizado. A função *game.tick()* chama o *filelistener()*, que permite a reinicialização de inimigos e executa as funções *player.tick()*, que possibilita a movimentação do jogador e escuta as interações do utilizador e a função *controller.tick()*, responsável por todos os comandos que não dependem do utilizados (balas, inimigos).

Importa, também, referir a utilidade das classes *BufferedImageLoader*, *Textures* e *Spritesheet*, que permitem uma renderização eficiente das imagens, que são criadas digitalmente através de um único ficheiro que é lido apenas uma vez (*spritesheet.png*). Esta imagem contém todas as outras utilizadas que são separadas digitalmente e chamadas através de um objeto *textures*.

III. DISCUSSÃO

1. Principais Resultados

O produto final deste trabalho revelou-se bastante satisfatório, tendo sido possível concretizar e atingir os principais objetivos definidos inicialmente. Não só se procedeu à criação de um jogo totalmente funcional, com armazenamento de *highscores*, como também se conseguiu implementar um circuito que permitisse a captação viável do sinal eletromiográfico e um processamento capaz de eficazmente detetar quando o utilizador se encontra com o punho fechado, permitindo, desta forma, controlar o avatar do jogo e disparar projéteis em tempo real.

No entanto, é de notar que se pretendia, à *priori*, detetar diferentes movimentos, através da captação de sinal por mais canais, porém por restrições de material e de tempo apenas foi possível implementar a leitura de um único gesto: o encerramento do punho. No entanto, o jogo está preparado para ler até 8 movimentos atualmente, contendo os avatares correspondentes (é possível visualizá-los clicando na tecla *enter*). O *Java* distinguiria os gestos através do nome do ficheiro *.txt* criado, por exemplo.

2. Dificuldades

As restrições impostas pelo regime *online* promoveram as dificuldades maiores no desenvolvimento deste projeto que se materializaram principalmente nas últimas semanas de trabalho, uma vez que apenas nessa altura foi possível testar o circuito desenhado, o qual teve de ser iterado algumas vezes para funcionar como previsto.

Além disso, notou-se uma discrepância entre os resultados e testes de processamento teóricos estudados *online*, utilizando bases de dados de sinais, e o que foi possível fazer e obter na realidade, principalmente no que toca ao processamento em *Matlab*, que foi preciso substituir para implementação do processamento em tempo real.

Outra barreira notável ao progresso do projeto foi a baixa fiabilidade dos componentes utilizados, que por vezes punham em causa a correta leitura do sinal, sendo, portanto, uma fonte grande de incerteza na análise do circuito construído.

Por outro lado, houve algum atrito na adesão ao *Redmine* gerada pela fraca simplicidade e eficiência da ferramenta, sendo que se tornou mais prático gerir o projeto utilizando ferramentas como o *Google Sheets* (planeamento e gestão) e a *Google Drive* (armazenamento), e assim foi possível organizar todas as tarefas, ideias e documentos envolvidos no trabalho.

3. Análise SWOT e Potencial de Melhoria

De forma a avaliar o sucesso do produto desenvolvido, foi realizada uma análise *SWOT* (*Strengths*, *Weakness*, *Opportunities*, *Threats*). Como principais forças destacam-se a facilidade de uso, sendo necessário colocar apenas 3 elétrodos no local correto e clicar no botão *Start Game*, e pode-se começar a jogar instantaneamente. É também um jogo bastante interativo e, sobretudo, divertido, tornando o fortalecimento muscular dos mais jovens num exercício dinâmico e entusiasmante. Além disso pretende-se atacar um mercado de nicho e com bastante facilidade de penetração.

Por outro lado, um dos pontos menos fortes mais pertinentes do produto é a necessidade de vir acompanhado com uma componente de *hardware*, o que poderá dificultar a adesão ao mesmo.

As ameaças mais relevantes à sua comercialização são a retoma dos serviços presenciais onde a probabilidade da fisioterapia feita presencialmente pode ser privilegiada pelos utilizadores. Há ainda a dificuldade de penetração no mercado de controladores por eletromiografia devido há existência de bastantes produtos semelhantes no mercado dos dispositivos biomédicos (no entanto, note-se que há poucas aplicações dos mesmos em jogos).

Por fim, diversas oportunidades de melhoria foram identificadas. No que toca ao processamento de sinal, salienta-se a possibilidade de junção da eficácia do processamento teorizado em *Matlab* com a eficiência do *script* em *Python*, através da utilização de bibliotecas *Python* já preparadas para o processamento EMG, como a *BioSPPy*, e a integração de um *multiplexer* no circuito que permita a captação de vários sinais em simultâneo e o seu processamento, multiplicando o número de movimentos passíveis de análise. Relativamente ao jogo, destacam-se a implementação de níveis de dificuldade adaptados a exercícios e condicionantes específicas de tratamento diferentes, o desenvolvimento de maiores componentes de *gamification* através de recompensas e de estratégias de progressão no jogo e, ainda, a capacidade de ter vários utilizadores a jogar ao mesmo tempo.

4. Comercialização do Produto

A estratégia de comercialização idealizada teve em vista salientar as forças e oportunidades do produto bem como ultrapassar as fraquezas e ameaças do mesmo.

Desde já, o público alvo principal são crianças em fase de reabilitação dos movimentos da mão, mas, de modo a ultrapassar a dificuldade imposta pelo facto de ser necessário vender uma parte física, pretende-se comercializar o produto em 2 partes, a aplicação (*HandsOn!*) e a parte física, doravante designada como *MyoController*. O jogo pode ser jogado sem recurso ao *MyoController*, sendo possível disparar projéteis utilizando a tecla *space*, pelo que se pode promover a comercialização da aplicação *HandsOn!* diretamente ao consumidor final (jovens), gratuitamente, e o *MyoController* (fonte de rendimento principal, aliada à publicidade presente no jogo) apenas a clínicas de fisioterapia que poderão, depois, fornecê-lo aos seus utentes, "receitando" níveis específicos do *HandsOn!* criando, assim, uma cultura de utilização integrada e dinâmica neste mercado.

Além disso, existe a possibilidade de disponibilizar a aplicação para *Android* e *IOS*, de modo a aumentar o mercado disponível. Note-se que em 2020, foram transferidos 80 milhões de jogos *mobile* que totalizaram 100 mil milhões de dólares de receita, dos quais cerca de 5.16% (correspondente 5 mil e 160 milhões de dólares) são atribuídos ao género *arcade*, no qual se insere o jogo desenvolvido [6]. Nesse mesmo ano foram criados mais de 2 milhões de anúncios pagos para o mesmo tipo de jogos [7]. Estes dados mostram o potencial de monetização através de publicidade, sem imputar um custo de transferência da aplicação. Há, ainda, a possibilidade de dinamizar o *MyoController* integrado noutros produtos com já bastante mercado, como por exemplo o *The Myo Armband*.

O nome atribuído ao produto criado foi *HandsOn! A Physio Game*, transmitindo de forma direta o intuito do jogo, acompanhado de um logótipo com um *design* simples e *clean*. A integração do punho fechado visa por um lado salientar a capacidade de controlo muscular através da execução desse gesto e, e por outro, explicitar a vertente reabilitante do jogo que promove o fortalecimento muscular do utilizador, sugerindo, através do punho cerrado, o seu empoderamento.

Foi, ainda, desenvolvido um *Pitch Deck* com o intuito de apresentar o produto a potenciais investidores que possam contribuir para o financiamento inicial de finalização do produto e de arranque da campanha de publicidade.

Figura B - Logótipo desenvolvido em 2 formatos

Figura C - Primeiro diapositivo do Pitch Deck desenvolvido

IV. CONCLUSÕES

Neste trabalho, mostrou-se como a integração de conhecimentos sobre sensores, processamento de sinal, criação de circuitos e programação é essencial na criação de um produto comercial para uso biomédico. Além disto, com a situação atual decorrente do Covid-19, salienta-se a experiência de coordenar um projeto à distância, no qual, por um lado, ferramentas como o *Google Drive* e *Google Meet* possibilitaram o contacto entre os membros do grupo, e por outro, tivemos a inspiração da criação do *HandsOn!* no seguimento do problema quanto à realização da fisioterapia à distância, uma vez que é uma atividade de maior risco devido à proximidade. A inclusão das diversas ferramentas

mencionadas, incluindo aquelas apresentadas nas aulas teóricas como *Java* e *Pencil*, permitiu-nos também ter um melhor conhecimento entre a conectividade dos diversos processos para o funcionamento de um jogo à base de um sinal biológico como o de EMG. Por fim, acredita-se que o projeto teve um processo enriquecedor e um resultado final satisfatório apesar das dificuldades enfrentadas.

V. REFERÊNCIAS

II. BIBLIOGRAPHY

[1] Sapsanis, C. (2013). *Recognition of Basic Hand Movements Using Electromyography, Diploma Thesis*. University of Patras - Dataset.

[2] Chowdhury, R. H. (13 de setembro de 2013). Surface electromyography signal processing and classification techniques. *Sensors (Basel, Switzerland) vol. 13,9 12431-66*.

[3] Mehendale, Gohel, V., & Ninad. (10 de novembro de 2020). *Review on electromyography signal acquisition and processing.* Obtido de National Library of Medicine: https://doi.org/10.1007/s12551-020-00770-w

[4] Rose, W. (2019). *KAAP 686: Mathematics for Biomechanics, Electromyogram analysis.* University of Delaware, Department of Kinesiology and Applied Physiology.

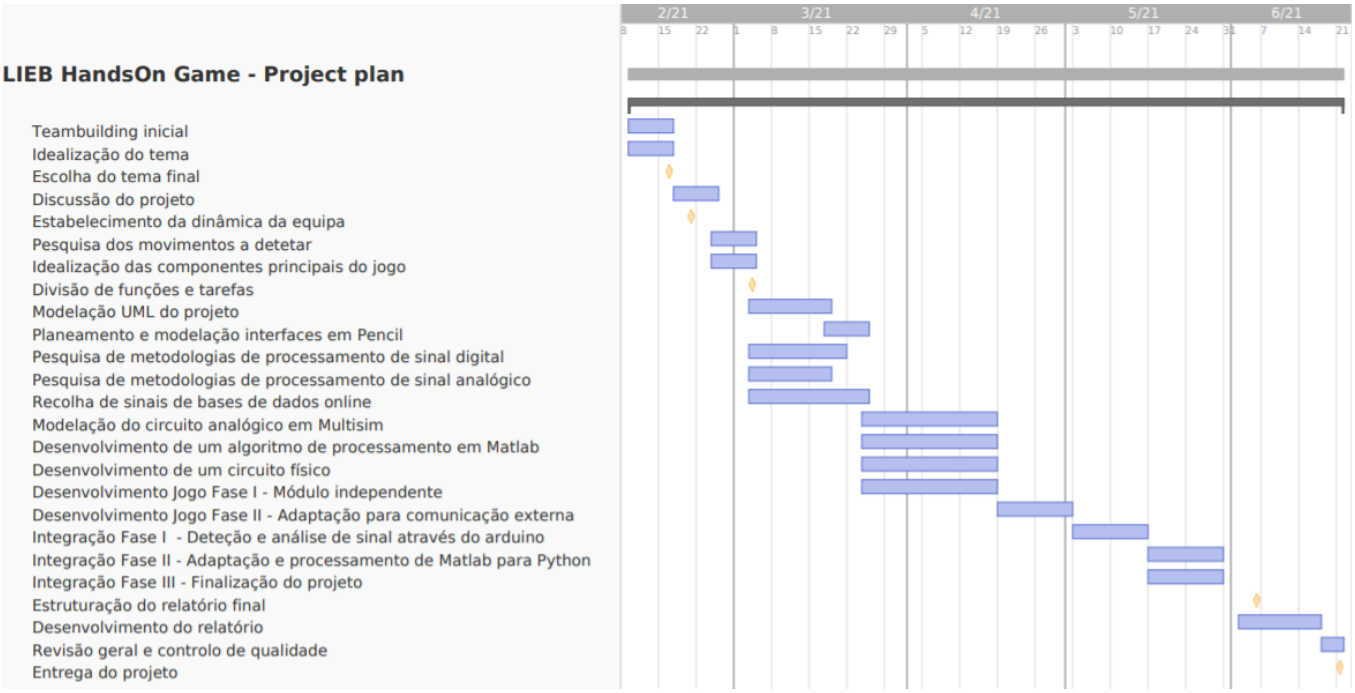
[5] Takeshi Tsujimura, S. Y. (2012). Hand Sign Classification Employing Myoelectric Signals of Forearm. *Computational Intelligence in Electromyography Analysis-A Perspective on Current Applications and Future Challenges, vol. 13*, pp. 309-336.

[6] Knezovic, A. (5 de maio de 2021). *Mobile Gaming Statistics*. Obtido de Udonis: https://www.blog.udonis.co/mobile-marketing/mobile-games/mobile-gaming-statistics

[7] Socialpeta. (24 de março de 2021). *Mobile gaming industry statistics and trends for 2021*. Obtido de Business of Apps: https://www.businessofapps.com/insights/mobile-gaming-industry-statistics-and-trends-for-2021/

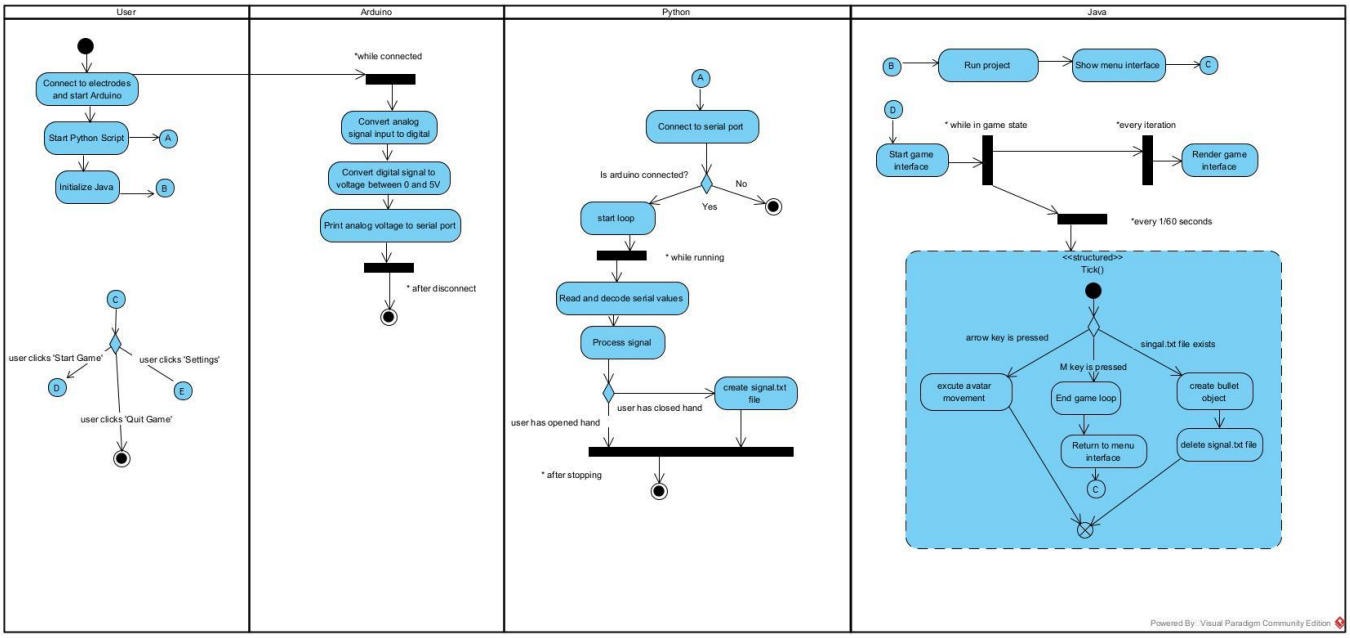
Anexos

Anexo 1 – Planeamento de tarefas em formato Gantt Chart



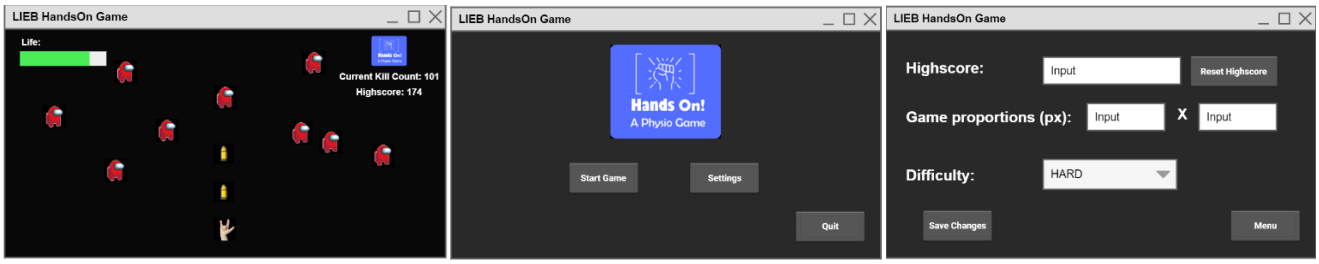
Anexo 1 - Gantt Chart

Anexo 2 – Diagrama de Atividade Inicial do Projeto

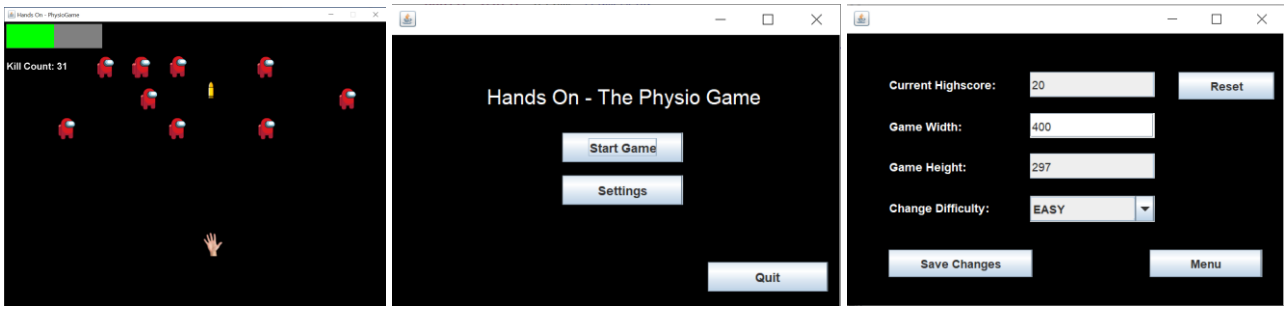


Anexo 2 - Diagrama de Atividade do Projeto

Anexo 4 – Comparação do modelo Pencil com a interface Java



Anexo 4.1 - Modelação da interface em Pencil



Anexo 3.2 - Interface Java

Anexo 5 – lista de bibliotecas importadas no projeto de Java

java.awt.Canvas;	java.awt.Graphics;	javax.swing.border.EmptyBorder;
java.awt.Color;	java.awt.image.BufferedImage;	javax.swing.DefaultComboBoxModel;
java.awt.Dimension;	java.awt.image.BufferStrategy;	javax.swing.event.CaretEvent;
java.awt.event.ActionEvent;	java.awt.Rectangle;	javax.swing.event.CaretListener;
java.awt.event.ActionListener;	java.io.BufferedReader;	javax.swing.JButton;
java.awt.event.KeyAdapter;	java.io.BufferedWriter;	javax.swing.JComboBox;
java.awt.event.KeyEvent;	java.io.File;	javax.swing.JFrame;
java.awt.event.MouseEvent;	java.io.FileReader;	javax.swing.JLabel;
java.awt.event.MouseListener;	java.io.FileWriter;	javax.swing.JPanel;
java.awt.event.WindowEvent;	java.io.IOException;	javax.swing.JTextField;
java.awt.event.WindowListener;	java.util.LinkedList;	javax.swing.SwingConstants;
java.awt.EventQueue;	java.util.Random;	
java.awt.Font;	javax.imageio.ImageIO;	

Tabela 2 - Lista de bibliotecas Java utilizadas

Anexo figuras

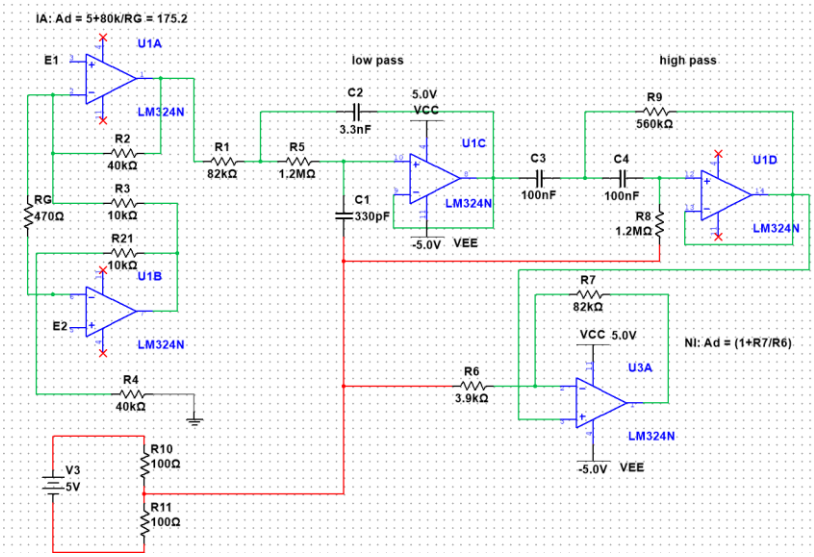


Figura 1 – Esquemático do circuito analógico projetado na plataforma Multisim

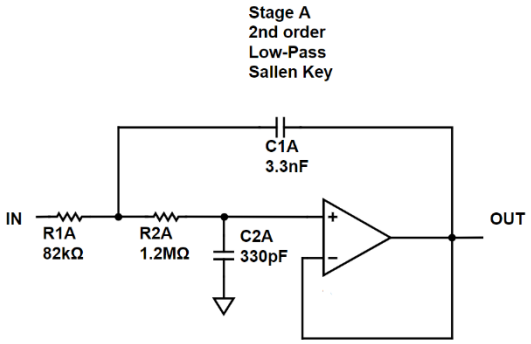


Figura 2 – Esquemático do filtro passa baixo projetado na plataforma Filter Wizard

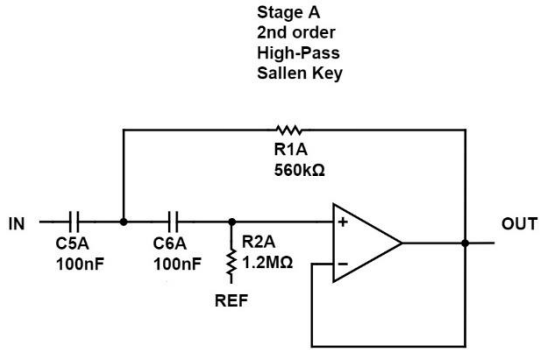


Figura 3 – Esquemático do filtro passa alto projetado na plataforma Filter Wizard

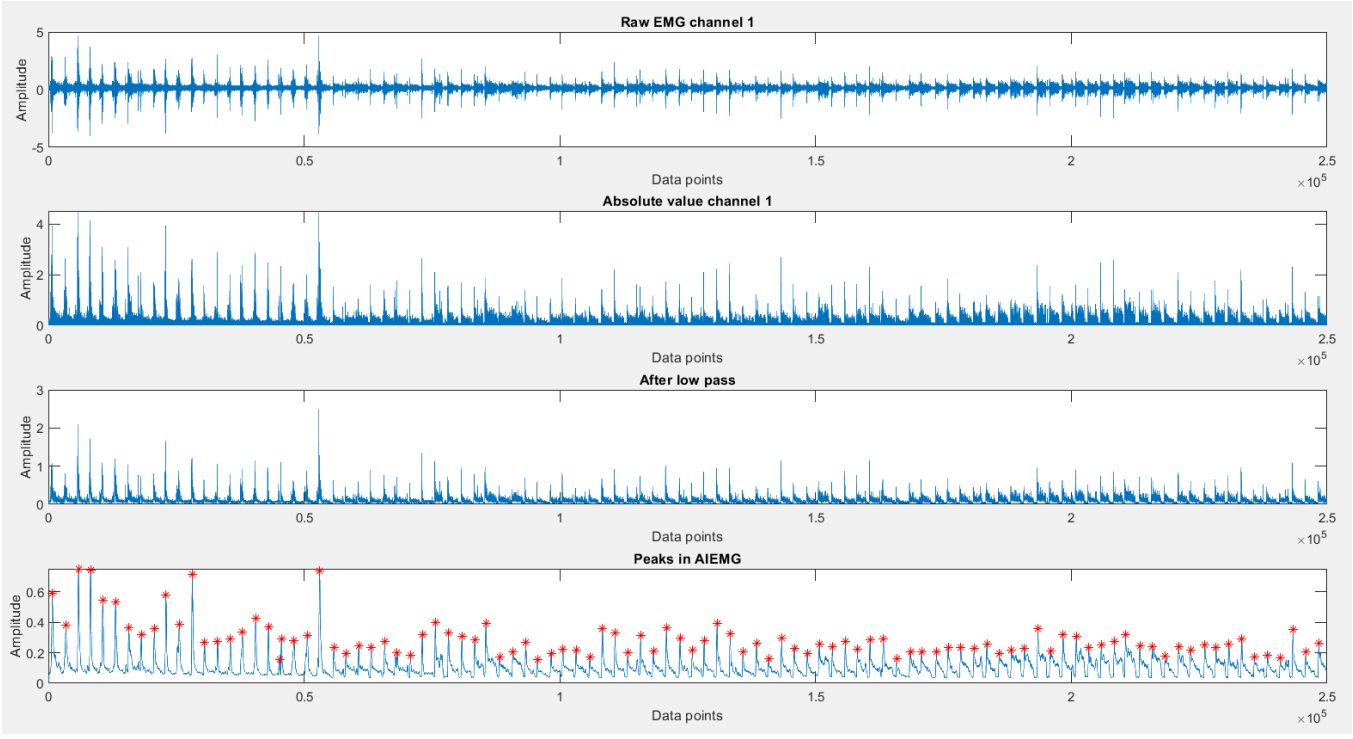


Figura 4 - Diversas etapas de processamento do canal 1 do dataset

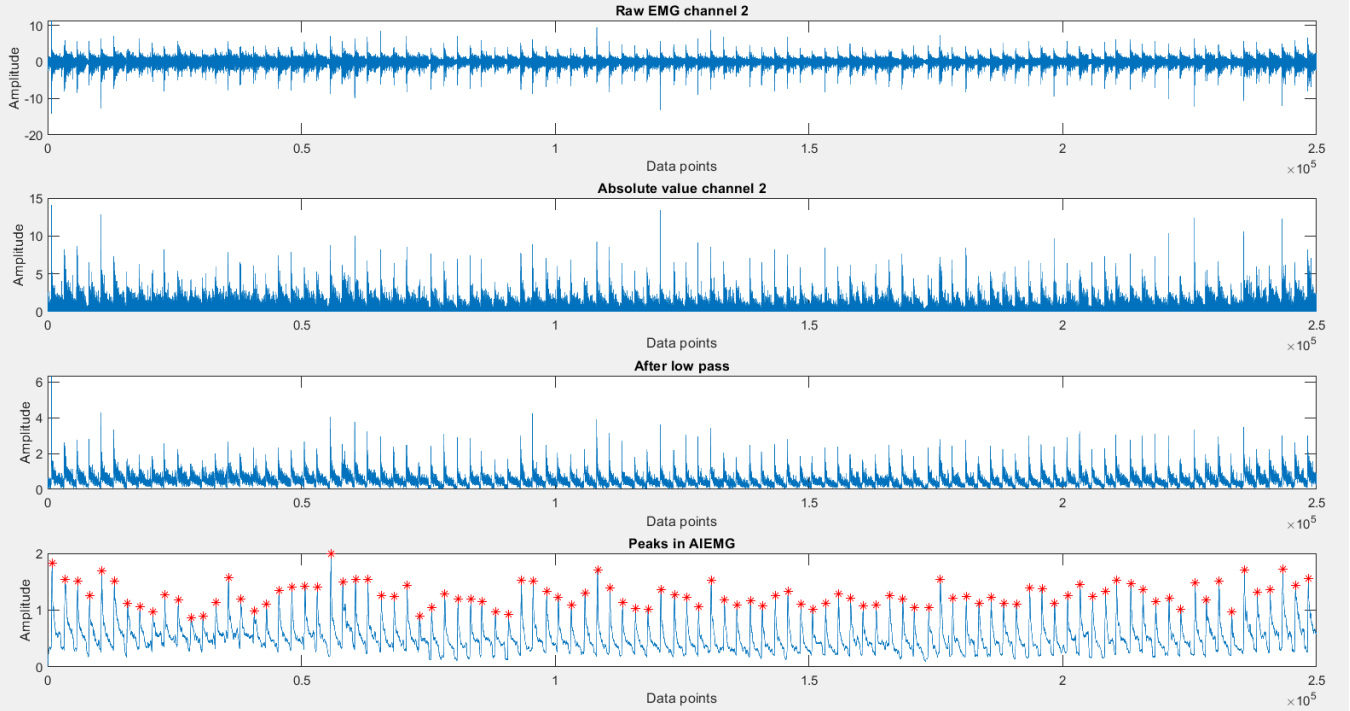


Figura 5 - Diversas etapas de processamento do canal 2 do dataset