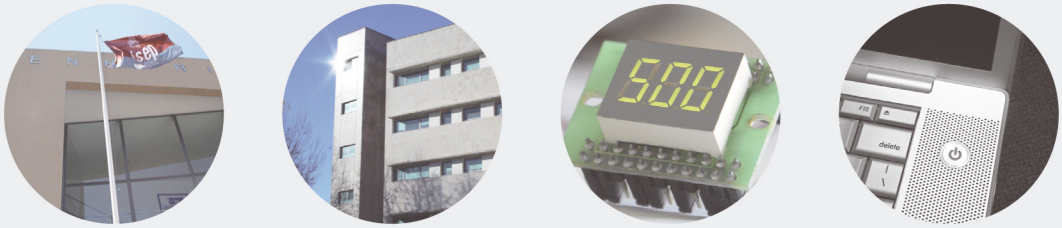
INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO
MESTRADO EM ENGENHARIA ELECTROTÉCNICA E DE COMPUTADORES



Sistema de recolha e análise de movimentos para fisioterapia.

FLAVIO DANIEL GONÇALVES RAMOS
novembro de 2016

POLITÉCNICO
DO PORTO

Instituto Superior de Engenharia do Porto

Sistema de recolha e análise de movimentos para fisioterapia

Tese/Dissertação de Mestrado

Para Obter Grau de Mestre Em
Mestrado em Engenharia Electrotécnica e de Computadores
Sistemas Autónomos
16 de Novembro de 2016

Flávio Daniel Gonçalves Ramos

Instituto Superior de Engenharia do Porto,
Porto, Portugal.

Orientador:

Dr. Lino Manuel Baptista Figueiredo

Coorientador:

Prof. António José Matos de Meireles

Relatório elaborado para satisfação parcial dos requisitos da Unidade Curricular de Tese/Dissertação do Mestrado em Engenharia Electrotécnica e de Computadores.

ISBN ++-++++-+++-+

Email do autor: 1110394@isep.ipp.pt

À minha família e amigos.

Agradecimentos

Gostaria de agradecer a todas as pessoas que encontrei durante todo meu percurso académico que me ajudaram em todos os níveis a alcançar os objetivos propostos. Ao estado português porque sem ele o meu trajeto no ensino superior não seria possível. E por fim, um obrigado a minha família e amigos pelo suporte que me tem dado ao longo dos anos e a ajudar-me a tornar uma melhor pessoa cada dia que passa.

Resumo

Este projeto consiste na recolha de impulsos elétricos produzidos quando um músculo fica ativo, para uma posterior análise de movimentos para a fisioterapia. A leitura destes impulsos tem o nome de eletromiografia (EMG). Este é um projeto que envolve várias áreas de investigação, entre elas a engenharia eletrotécnica, a engenharia informática e a medicina.

O sistema de adquisição de dados consistiu no desenvolvimento de um circuito de condicionamento do sinal, em que o sinal que sai do circuito é lido pelo módulo de conversão analógico/digital, do inglês *analog to digital converter* (ADC) inserido dentro de um microcontrolador, que faz uso do seu módulo de transmissão de dados para enviar esse sinal convertido para um software desenvolvido de raiz onde é possível visualizar graficamente o sinal.

Para validar o sinal adquirido com o sistema desenvolvido, utilizou-se um outro equipamento, que já existe no mercado e que permite ler o sinal EMG para além de outros, chamado de Biopac MP36 Student Lab System, onde foram feitos testes com ambos os equipamentos para comparar os resultados obtidos.

Feita a análise entre os dois sistemas, o sinal obtido pelos dois equipamentos eram bastante semelhantes, validando-se o sistema desenvolvido com uma nota positiva.

Este relatório tem como objetivo de explicar o trabalho efetuado durante a implementação do projeto, onde foram proporcionadas novas perpetivas da aplicação da elétronica em outras áreas de investigação, uma aprendizagem de novos conceitos e aplicação dos mesmos.

Palavras-Chave

EMG, Microcontrolador, Músculos, Fisioterapia, Biopac, Interface gráfica, *bluetooth*, Condicionamento de sinal.

Abstract

This project involves the collection of electrical impulses produced when a muscle is active for a further analysis of movements for physical therapy. Reading these impulses is called electromyography (EMG). This project involves several areas of research, including the electrotechnical engineering, computer engineering and medicine.

The data acquisition system consisted in developing a signal conditioning circuit, where the signal output of the circuit is read by the analog conversion module analog to digital converter (ADC) inserted within a microcontroller, which makes use of its data communication module to send that signal converted to a software developed where it can graphically display the signal.

To validate the acquired signal with the developed system, we used other equipment that already exists in the market that allows you to read the EMG signal in addition to others signals, called MP36 Biopac Student Lab System, where tests were done with both devices to compare the results obtained.

After the analysis of the two systems, the signal obtained by both equipment were quite similar which indicates that the system developed has a positive note.

This report aims to explain the work done during the project implementation, which were provided new perspectives of the application of electronics in other areas of research, and learning of new concepts and application of the same ones.

Key Words

EMG, Microcontroller, Muscles, Physiotherapy, Biopac, Graphical User Interface, Bluetooth, Signal conditioning.

Conteúdo

Lista de Figuras

Lista de Tabelas

Acrónimos

ACh	Acetilcolina
A/D	Analógio-Digital
ADC	Analogic-Digital Converter
AgCl	Coreto de Prata
AO	Amplificador Operacional
ASCII	American Standard Code for Information Interchange
C	Condensador
CMRR	Commum Mode Rejection Ratio
CPU	Central Processing Unit
CTC	Clear Timer on Compare
EEPROM	Eletrical-Erasable Programmble Read-Only Memory
EMG	Eletromiografia
EMGS	Eletromiografia de Superfície
ENG	Eletroneurografia
EPROM	Erasable Programmble Read-Only Memory
ESTSP	Escola Superior de Tecnologia do Porto
I/O	In/Out
ISEP	Instituto Superior de Engenharia do Porto
LCD	Liquid-Crystal Display
MatLab	Matrix Laboratory
MUAPs	Motor Unit Action Potencial
R	Resistência
RAM	Random Acess Memory
RC	Resistência-Condensador
RISC	Reduced Instruction Set Computer

ROM Read-Only Memory

SENIAM Surface EMG for the Non-Invasive Assessment of Muscles

UART Universal Asynchronous Receiver Transmitter

USART Universal Synchronous Asynchronous Receiver Transmitter

USB Universal Serial Bus

1 Introdução

Neste capítulo será feita uma contextualização do projeto que será desenvolvido, como surgiu a ideia e qual será a utilidade deste trabalho no mundo da medicina. Serão também apresentados os objetivos definidos para este trabalho assim como a calendarização das etapas tomadas para a elaboração do projeto. Por fim é apresentada a organização do relatório, explicando como está estruturado e com um pequeno resumo sobre cada um dos capítulos.

1.1 Contextualização

Este projeto surgiu de uma parceria entre a "Porto Design Factory" e a Escola Superior de Tecnologia da Saúde do Porto (ESTSP). No momento em que esta tese foi desenvolvida o departamento de fisioterapia do ESTSP, possui um sistema que permite perceber se um indivíduo possui o risco de desenvolver entrose no tornozelo. No entanto, o mecanismo que possuem é bastante manual e não permite retirar as conclusões pretendidas para uma análise mais realista e assertiva. O método de deteção de possível entorse passa por colocar o indivíduo sobre uma caixa, que está apresentada na Figura 1.1, geralmente apoiado em apenas um dos pés.

Fonte: elaborado pelo autor.

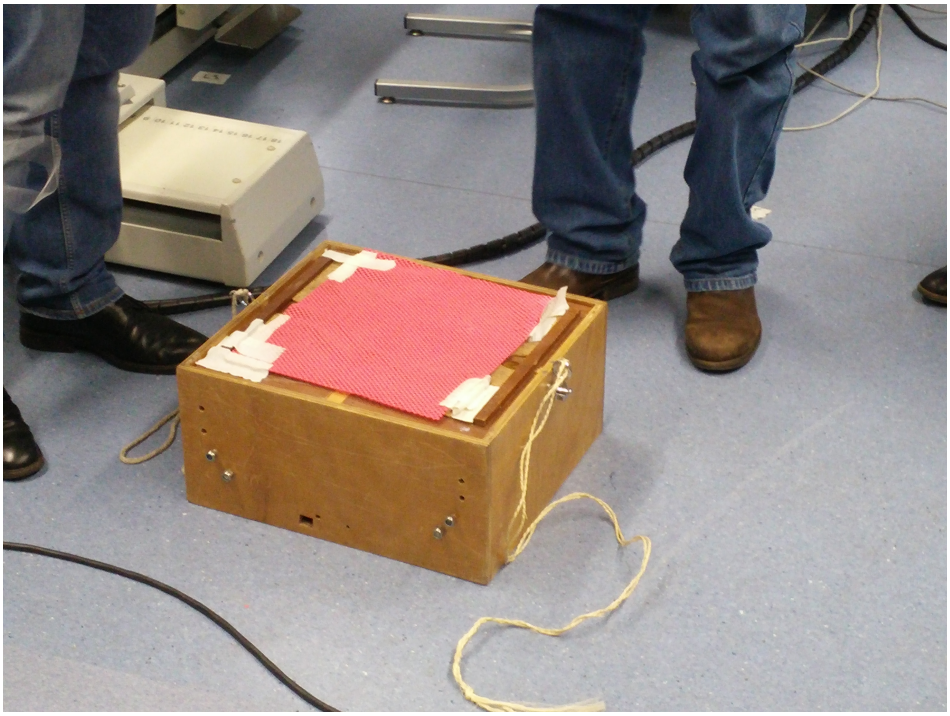


Figura 1.1: *Caixa para o teste de entorse do tornozelo.*

A caixa tem dois parafusos um de cada lado da caixa para permitir a abertura da tampa em cada um dos lados e possui também quatro batentes, dois em cada lado da caixa. Estes dois batentes estão colocados de forma a que quando a tampa do seu lado feche, o ângulo de abertura seja de 30 graus ou de 45 graus. A existência de dois tipos diferentes de ângulos de abertura prende-se pelo facto de ser possível avaliar todos os pacientes de várias faixas etárias, por exemplo, para um adulto o ângulo de abertura estabelecido pelos fisioterapeutas foi de 45 graus, enquanto para um idoso o ângulo de abertura já é mais reduzido, cerca de 30 graus. Estes ângulos foram definidos de forma a ser avaliada a propensão para entorse sem que ocorra uma verdadeira entorse. É por este facto que para os idosos o ângulo de abertura é menor, pois os seus tendões e músculos já não possuem o fortalecimento e elasticidade necessárias para aguentar com um ângulo de abertura maior sem que haja o rompimento dos tendões que levem a entorse do tornozelo.

Para simular o ato de entorse, o paciente é colocado sobre a caixa com um dos pés sobre um dos lados da tampa, com os elétrodos colocados no músculo que se quer

avaliar, de seguida é puxado um fio que está ligado a um dos parafusos, que quando é retirado por força do fio ter sido puxado, a zona da tampa baixa até encontrar um dos batentes. Todo o processo de registar a atividade do músculo é efetuada através de um teste de eletromiografia já existente no laboratório e o momento em que a tampa é aberta é registado com um acelerómetro, que está junto à tampa, onde os valores dos eixos do acelerómetro e da eletromiografia são obtidos na mesma escala de tempo para depois comparar o instante da abertura da tampa com o valor da eletromiografia.

Quando a tampa se abre, o músculo tende a reagir contraindo-se de forma a proteger o tornozelo de sofrer uma entorse. O que se pretende avaliar neste género de testes de entorse é o tempo que o músculo demora a reagir, desde que a tampa foi aberta. Um tempo de reação menor indica que o individuo não corre um risco elevado de sofrer entorse se um dia se deparar com as mesmas condições que efetuou no teste. Com um tempo de reação mais demorado, é indicação que o paciente pode vir a sofrer de entorses com alguma frequência, quando enfrentar as mesmas condições que encontrou no teste.

Com o sistema de eletromiografia é possível conhecer qual o instante em que o músculo reagiu, contudo como a abertura do tampo da caixa é feita de uma forma manual, o paciente, em norma, apercebe-se de quando a tampa vai ser aberta e tenta, involuntariamente, contrair o músculo antecipadamente. Com a abertura da tampa feita manualmente, o teste de perceber quanto tempo é o que o músculo demora a reagir a quando da abertura da tampa é inválido, pois o músculo já tinha reagido antes da tampa ser baixada. Com este problema em mente, o intuito deste projeto, relativamente a parte da eletrónica, passa por desenvolver um sistema que consiga ler o sinal EMG, amostrá-lo numa interface gráfica e registar o sinal EMG com um tempo de amostragem na ordem dos 1 ms.

Relativamente à caixa, os engenheiros mecânicos estão a desenvolver uma mecanismo que permita que a tampa da caixa seja aberta, sem fios, em que o controlo de abertura seja realizado por parte da interface gráfica, para assim haver um sincronismo entre o instante que a tampa foi aberta e o início da leitura do sinal EMG. Com a parceria destas duas engenharias prevê-se que este teste para conhecer se o paciente pode vir a sofrer entrose no tornozelo, se torne mais válido, já que possuirá um maior rigor no sincronismo entre a abertura da tampa e registo do momento em que o músculo reagiu.

1.2 Objetivos

O principal objetivo deste projeto é de construir um sistema capaz de realizar de uma forma prática e competente a leitura do sinal EMG, com um protótipo que seja de fácil utilização, portátil e de preço reduzido. O segundo objetivo passa por desenvolver uma interface gráfica para apresentar o sinal EMG para os médicos fisioterapeutas conseguirem efetuar a análise do teste realizado. As várias etapas que se realizaram até chegar ao protótipo final foram:

- Estudo do sistema nervoso e dos músculos;
- Estudo dos diferentes métodos para a aquisição do sinal eletromiográfico;
- Pesquisa de vários microcontroladores que existem no mercado, as suas características, os módulos de ADC e ambientes de programação;
- Desenvolvimento de um algoritmo para melhorar o sinal eletromiográfico obtido;
- Estudo sobre programas que permitem criar uma interface gráfica, como programar e como visualizar graficamente o sinal EMG;
- Elaboração de uma aplicação gráfica para interface com o sistema operativo *Windows* e *Ubuntu*.

1.3 Calendarização

O trabalho foi desenvolvido de uma forma continua com uma duração de 6 meses, em que o tempo de cada uma das etapas para a elaboração deste projeto está descrito na tabela 1.1. O tempo passado a desenvolver este produto foi divido nas seguintes fases: estudo sobre o sistema nervoso, funcionamento dos músculos, eletromiografia, circuitos de aquisição do sinal EMG, microcontroladores, *software* para desenvolver interfaces gráficas. Depois da fase do estudo sobre os tópicos sobre a biologia e eletrónica associada ao projeto, passou-se a uma fase de testes com o sistema biopac para ganhar sensibilidade para o tipo de sinal que era suposto obter. Com todo o conhecimento reunido com as fases anteriores passou-se para a implementação do projeto, montagem do circuito para aquisição do sinal, programação do microcontrolador para converter e enviar o sinal e por último, a criação de uma interface gráfica para amostrar graficamente o sinal. Com a implementação realizada, a última fase foram efetuados testes ao sistema desenvolvido.

Tabela 1.1: *Calendarização do projeto*

	Semana																						
Tarefas	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**
Estudo dos músculos	■	■	■																				
Estudo do módulo de condicionamento de sinal		■	■	■																			
Estudo de microcontroladores				■	■																		
Realização dos testes do Biopac						■																	
Montagem do circuito de condicionamento de sinal							■	■	■	■	■	■											
Programação dos microcontroladores												■	■	■									
Criação da interface gráfica														■	■	■	■	■	■				
Realização de testes																			■	■	■		
Elaboração do relatório	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■

1.4 Organização do relatório

Este relatório está estruturado em sete capítulos. No 1º capítulo, "Introdução"é feita a contextualização do projeto a desenvolver, quais os objetivos definidos, a calendarização e a como está estruturado o relatório. O 2º capítulo é referente a pesquisa bibliográfica efetuada para melhor intender o processo que seria posto em prática para obter os resultados desejados. No 3º capítulo é apresentado os testes realizados com o sistema biopac. O 4º capítulo é referente a arquitetura geral do sistema, introduzindo ao leitor qual a solução escolhida para o projeto a desenvolver. No 5º capítulo é demonstrada como foi feita a implementação do hardware para obter os resultados desejados. O 6º capítulo mostra os resultados obtidos, comparando-os com os resultados do sistema biopac e é feita uma análise com uma avaliação sobre os resultados. No 7º capítulo, são apresentadas as conclusões retiradas com a implementação deste projeto e quais as expetativas para o futuro deste produto. Existe ainda a secção dos anexos, com informações adicionais sobre o desenvolvimento do projeto, nomeadamente o código implementado e o esquema elétrico do sistema desenvolvido.

2
Estado da Arte

Neste capítulo serão abordados os tópicos estudados que são fulcrais para a ajudar a perceber melhor como será feita a implementação deste projeto. Em primeiro lugar são apresentados os temas ligados à biologia, mais concretamente, o que é e como ocorre uma entorse do tornozelo, o que é a fisioterapia, o funcionamento do sistema nervoso, é mencionado o sistema muscular e por fim é apresentada a eletromiografia. Depois de perceber como funciona o sistema nervoso, a constituição dos músculos, como surge uma entorse, o que é a eletromiografia e como funciona, passou-se ao estudo da eletrónica associada a este tipo de projeto. Relativamente à eletrónica, são apresentadas as varias configurações possíveis para ler um sinal EMG, como é realizada a filtragem do sinal e que filtros existem, a conversão de um sinal analógico para digital e que microcontroladores existem no mercado que possam ser usados para este projeto. Por fim, são apresentados sistemas que já existem no mercado que conseguem realizar eletromiografia.

2.1 Entorse do Tornozelo

A entorse de tornozelo pode ser definida como uma lesão articular que gera dor, limitação funcional (perda de movimentos) e instabilidade articular, podendo ou não estar associada a um rompimento dos ligamentos. Na região de articulações do tornozelo temos 7 ligamentos (3 laterais e 4 centrais) que juntamente com músculos e estruturas ósseas são responsáveis pela estabilidade do tornozelo, ver Figura 2.1.

Fonte: Academia de futebol, com edição do autor[1].

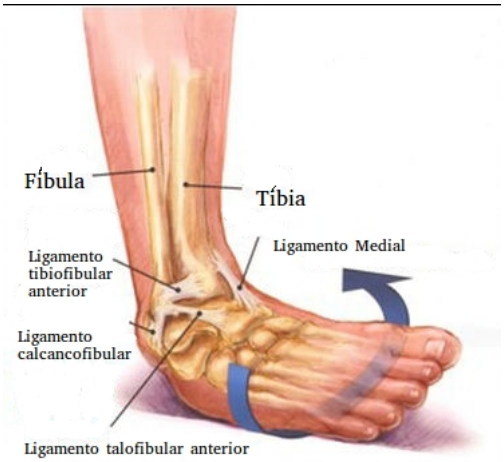


Figura 2.1: *Anatomia do tornozelo.*

Esta é uma das principais lesões ligadas ao desporto, comum a atletas de diversas modalidades, tendo o atletismo, basquetebol, voleibol e futebol como os desportos com maior incidência. As entorses são classificadas de acordo com sua gravidade: grau I-leve (leve estiramento de ligamentos), grau II-moderado (lesões parciais de ligamentos), grau III-grave (lesões totais/rupturas completas dos ligamentos). As entorses também recebem classificação quanto ao tipo, podendo ser, entorse por inversão (Quando durante a torção o pé vira para dentro) e entorse por eversão (Quando durante a torção o pé vira para fora). As entorses por inversão são as mais comuns representando 90% do total de lesões, e está geralmente associada a alguma lesão dos ligamentos. As entorses por eversão são mais incomuns e podem estar associadas com alguma fratura óssea.

Pé cavo, diminuição de força dos músculos que agem sobre o tornozelo, frouxidão dos ligamentos, défice proprioceptivo[1], joelhos em valgo[2], joelhos em varo[3] e mais alterações rotacionais dos membros inferiores são alguns dos fatores intrínsecos que podem gerar as entorses de tornozelo. Os traumas desportivos, quedas de escadas e degraus e as condições precárias das vias publicas são fatores extrínsecos que podem contribuir para as entorses de tornozelo.

O tratamento varia dependendo da gravidade do trauma e das possíveis lesões associadas. O objetivo do tratamento da lesão do tornozelo é o retorno às atividades diárias, com remissão da dor, inchaço e inexistência de instabilidade articular. Para

[1]Capaz de receber estímulos originados nos músculos, tendões ou outros órgãos internos.
[2]Dobrado ou dirigido para fora.
[3]Que está voltado ou virado para dentro.

isso diversas medidas podem ser adotadas, como por exemplo, usar gelo, uma faixa elástica, medicação adequada, fortalecimento muscular, treino proprioceptivo, etc.[1] Na Figura 2.2 estão representados os dois tipos de entorses que podem ocorrer.

Fonte: Fisioterapia manual, com edição do autor[2].

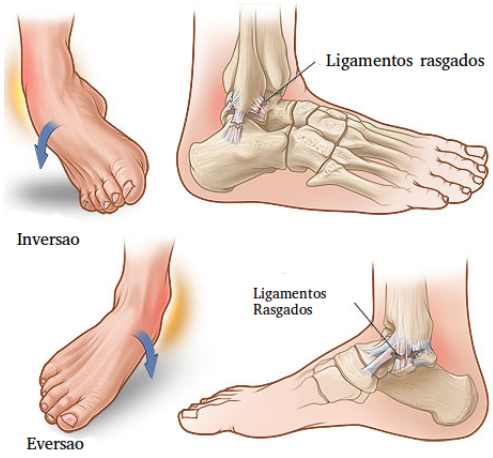


Figura 2.2: *Tipos de entorse do tornozelo.*

Para tratar de entorses, recorre-se a fisioterapia para tentar recuperar o paciente. Fisioterapia é a ciência que estuda, analisa e avalia o movimento e a postura, baseadas na estrutura e função do corpo. Faz uso de estudos educativos e terapêuticos específicos, com base essencialmente no movimento, nas terapias e meios físicos e naturais. Tem como objetivos a finalidade da promoção da saúde e prevenção da doença, da deficiência, da incapacidade e da inadaptação e de tratar, habilitar ou reabilitar, indivíduos com disfunções de natureza física, mental, de desenvolvimento ou outras, incluindo a dor, com o objectivo de os ajudar a atingir a máxima funcionalidade e qualidade de vida.

2.2 Descrição do Sistema Nervoso

A maioria dos animais interage com o meio graças à constante circulação de mensagem no seu organismo. Esta rede de mensagens é assegurada pelo sistema nervoso que é constituído por um conjunto de órgãos em estreita ligação ente si.[3][4]

O sistema nervoso coordena e regula todos os atos conscientes e inconscientes dos indivíduos. No sistema nervoso central, encéfalo e medula espinal, dá-se a integração, isto é, a interpretação dos estímulos provenientes quer do meio interno ou externo e a preparação de respostas adequadas aos estímulos recebidos. Os estímulos e

as respostas circulam então, através dos nervos do sistema nervoso periférico, que chegam, respetivamente, de e a todas as partes do corpo. Os órgãos do sistema nervoso central ficam assim ligados aos diferentes sistemas de órgãos, como músculos, órgãos dos sentidos, etc. Os elementos centrais desta verdadeira rede de comunicação são as células nervosas, ou neurónios, células especializadas que variam de tamanho e de forma, mas que mantêm um padrão comum. Existem três tipos de neurónios, os neurónios sensitivos, que transitem informações dos recetores sensoriais para os centros nervosos. Os neurónios motores que transmitem ordem dos centros nervosos para os órgãos efetores, como por exemplo, os músculos. Os interneurónios, como o nome sugere, integram a informação que chega dos neurónios sensitivos e preparam a mensagem para os neurónios motores, os interneurónios estão localizados no encéfalo ou na medula espinal.

Num neurónio em repouso, isto é, que não está a conduzir uma mensagem, o potencial de membrana[4] pode ser de -70 mV, a este potencial chama-se potencial de repouso. Este valor negativo significa que o interior da célula próxima da membrana tem uma carga negativa relativamente ao fluido extra-celular. Contudo os neurónios são células excitáveis, respondendo a estímulos. Qualquer mudança no meio, interno ou externo, como o som, a luz ou uma substancia química, funciona como um estimulo que, ao ser captado ou transformado, pode alterar o potencial de repouso, uma vez que alterar a permeabilidade da membrana dos neurónios aos iões, gerando um potencial de ação. Este traduz numa inversão rapida das cargas elétricas de uma porção da membrana de um neurónio. Uma vez iniciado este potencial de ação propaga-se ao longo do axónio[5]. Os potencias de ação são conduzidos ao longo axónio a uma velocidade que pode ser superior a 100 metros por segundo. Depois de passar o potencial de ação, o potencial da membrana rapidamente regressa ao seu valor de repouso neste caso -75 mV. A diferença de carga elétrica entre as zonas em repouso no axónio e as zonas em atividade gera uma corrente elétrica. Apenas as células nervosas, células musculares e alguns tipos de células endócrinas[6] e de células do sistema imunitário podem vir a possuir um potencial de ação e repouso, as restantes células apenas possuem o potencial de repouso.

[4]Quantidade de energia gerada pela diferença de cargas elétricas entre o interior e o exterior da membrana.

[5]Parte do neurónio responsável pela condução dos impulsos elétricos.

[6]Células que produzem hormonas que libertam na circulação sanguínea.

2.3 Sistema Muscular

Os músculos são os órgãos que geram a força que permite o movimento, conseguido à custa da capacidade que as fibras musculares têm de se contrair e alongar. Esse deslizamento entre as fibras musculares produz movimento. No entanto para que tal seja possível, os músculos têm necessariamente que estar ligados aos ossos, ligação que se faz através de tecido fibroso denominado tendão. Em resumo, é a actividade produzida pelos músculos, ligados aos ossos pelos tendões, com ajuda das articulações que funcionam como dobradiças, que permite o movimento. Posto este conceito de capacidade de movimento, existe um outro que é necessário reter para que se perceba a verdadeira capacidade dos músculos, esse conceito é o de tónus muscular[7]. Todos os músculos funcionam da mesma maneira. A contração dá-se com a chegada de um sinal nervoso. Quando este cessa, o músculo relaxa e sob o efeito de músculos antagonistas (os que fazem executar o movimento contrário), volta a distender-se. Enquanto que a contração é ativa, porque é determinada por um sinal preciso, a distensão é passiva. O músculo é constituído por feixes de fibras musculares paralelas. Uma fibra muscular é uma célula especial que contem cordeos de proteínas, as miofibrilas. As miofibrilas apresentam uma sucessão de bandas claras e escuras alternadas. Estas bandas devem-se à presença de moléculas de miosina, que formam filamentos espessos e moléculas de actina, que formam filamentos delgados. No músculo distendido os filamentos de actina estão distantes uns dos outros. Na contração, os filamentos de actina aproximam-se entre si, deslizando entre os filamentos de miosina. A contração de um músculo inteiro deve-se à soma de muitos milhares de deslizes de moléculas de actina. Esses deslizes produzem-se graças aos movimentos das pequenas cabeças que existem nas moléculas de miosina [4]. Podemos então dizer que os músculos: mantêm e facilitam posições, permitem movimentos e produzem calor pela sua contracção que liberta energia sob a forma de calor.

Sabendo para que servem e como basicamente funcionam, interessa agora classificar os músculos, uma vez que nem todos são iguais. Essa classificação baseia-se na capacidade do músculo ser ou não movimentado voluntariamente, isto é pela vontade própria de um indivíduo. Uma vez que, para cada músculo contrair tem que haver um estímulo produzido pelo sistema nervoso. O que se pretende classificar é o facto de esse estímulo nervoso ter sido ou não produzido por vontade própria.

[7]Por tónus muscular entende-se basicamente a rigidez muscular, ou seja a capacidade que o músculo tem de adquirir determinada forma e posição.

Músculo Esquelético

O músculo esquelético liga-se aos ossos e permite movimentos voluntários. É constituído por fibras musculares mais compridas (Figura 2.3).

Fonte: Sua vida em forma[5].

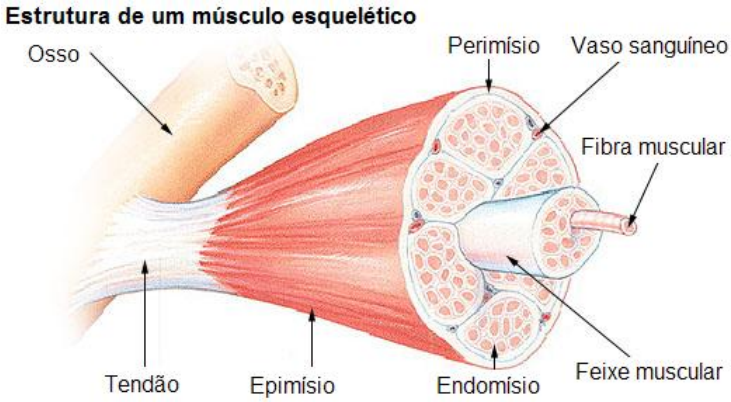


Figura 2.3: *Músculo Esquelético.*

Músculo Liso

Relativamente ao músculo liso, este é mais curto, a sua acção não depende da vontade própria do individuo, é involuntário. Por exemplo a camada muscular dos intestinos, do estômago ou do útero. Na Figura 2.4 está representada a descrição de um músculo liso.

Fonte: Cola da web[6].

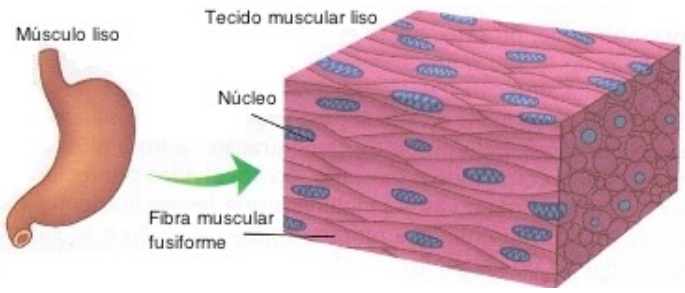


Figura 2.4: *Músculo Liso.*

Músculo Cardíaco

Por último o músculo cardíaco é constituído por fibras que se ramificam umas nas outras e a sua acção é involuntária e rítmica, quer isto dizer que a grande diferença é o facto de para além de o músculo cardíaco não poder ser controlado voluntariamente, tem a capacidade de ser automático isto é, pode produzir, em caso de necessidade, sem interferência do sistema nervoso um estímulo que permita a sua contracção. A Figura 2.5 ilustra a estrutura do músculo cardíaco.

Fonte: Apresentação sobre o músculo cardíaco, com edição do autor[7].

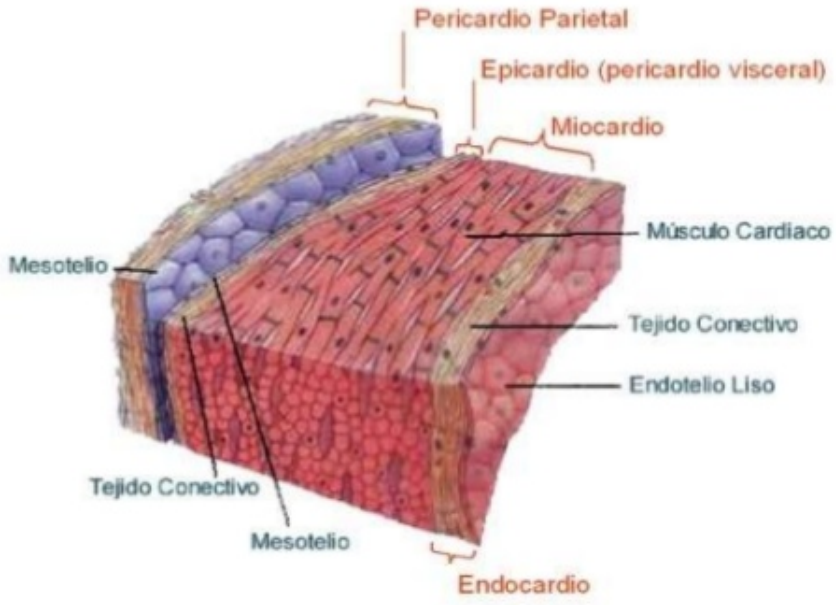


Figura 2.5: *Músculo Cardíaco.*

2.3.1 Músculos do Tornozelo

O tornozelo exerce uma função primordial na locomoção do ser humano. Além de sustentar o peso do corpo, deve adaptar-se para absorver forças e adaptar-se face a superfícies irregulares. Com esta quantidade de tensão, o corpo humano possui um grande sistema de apoio muscular, ajudando os ligamentos a estabilizar a articulação. Vários músculos apoiam o tornozelo e o sistema esquelético do pé inteiro. Estes músculos são responsáveis pela atividade normal do tornozelo e pela sua estabilização. Dois tipos de músculos atuam sobre o pé e o tornozelo: os músculos intrínsecos e os músculos extrínsecos.

Os músculos intrínsecos são os mais curtos, só se fixam nos ossos do pé e são responsáveis, em parte, pela massa muscular na região da planta do pé. Dos 20 músculos individuais do pé, 14 estão localizados na face da planta do pé, 2 estão na face dorsal e 4 são intermediários. Na Figura 2.6 são visíveis os músculos do pé.

Fonte: Aula de anatomia[8].

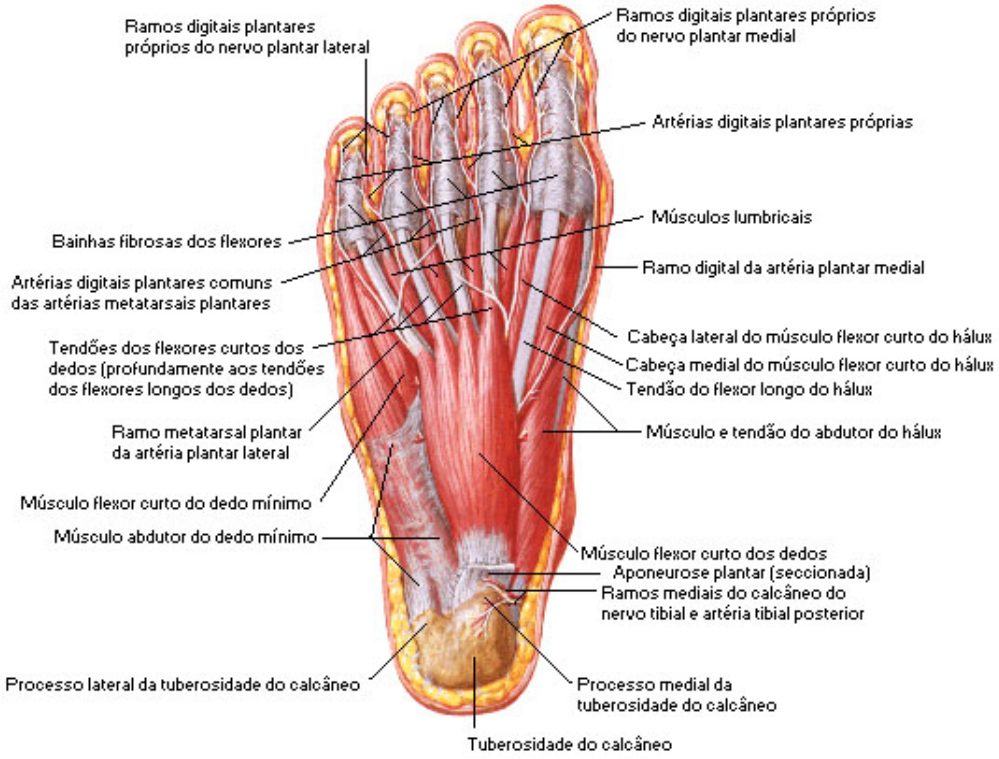


Figura 2.6: *Músculos Intrínsecos do Pé.*

Relativamente aos músculos extrínsecos, são os músculos que se fixam em outros ossos dos membros inferiores para além dos que constituem o pé, tal como: a tíbia, o perónio e o fémur. São todos poliarticulares[8] e atuam sobre o tornozelo e o pé. Este subdividem-se em 3 grandes grupos: os anteriores, os laterais e os posteriores.

[8]Que possuem várias articulações

Músculos Extrínsecos do pé, região anterior

Três músculos longos encontram-se situados na região anterior da perna. Os tendões curvam-se antes do tornozelo onde são mantidos por uma "fita" de ligamentos: o retináculo inferior dos músculos extensores. São eles: Tibial Anterior, extensor longo da hálux, extensor longo dos dedos e fibular terceiro (Figura 2.7).

Fonte: Apresentação sobre o complexo articular do tornozelo[9].

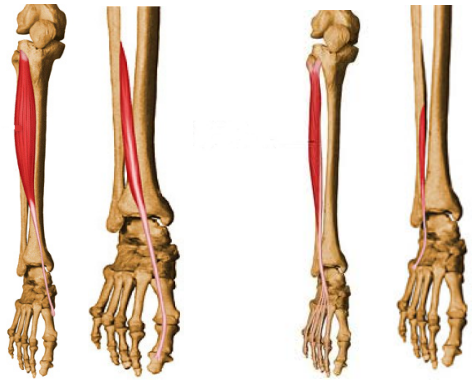


Figura 2.7: *Por ordem de apresentação(Da esquerda para a direita) Tibial Anterior, extensor longo da hálux, extensor longo dos dedos e fibular terceiro.*

Músculos Extrínsecos do pé, região lateral

Na região lateral da perna encontram-se dois músculos que se fixam na face lateral do perónio. São eles: Fibular Longo e Fibular Curto (Figura 2.8).

Fonte: Apresentação sobre o complexo articular do tornozelo[9].

Figura 2.8: *Por ordem de apresentação(Da esquerda para a direita) Fibular Longo e Fibular Curto.*

Músculos Extrínsecos do pé, região posterior

O grupo posterior dos músculos da perna é o mais importante. Consta de duas camadas. A camada mais profunda é constituída por três músculos situados um ao lado do outro nas faces posteriores da tíbia e do perónio. São eles: Flexor longo dos dedos, Tibial Posterior e Flexor longo do Hálux (Figura 2.9).

Fonte: Apresentação sobre o complexo articular do tornozelo[9].

Figura 2.9: *Por ordem de apresentação(Da esquerda para a direita) Flexor longo dos dedos, Tibial Posterior e Flexor longo do Hálux.*

A camada superficial do grupo muscular posterior é constituída pelo músculo Plantar e pelo Tríceps Sural. Este músculo, o mais forte da perna é formado por três partes musculares que tem uma mesma inserção: o tendão do calcâneo (Tendão de Aquiles), o qual se encontra na face posterior do calcâneo. A parte mais profunda é ocupada pelo Sóleo. Este é recoberto por partes mais superficiais: as cabeças medial e lateral do Gastrocnémio (Figura 2.10).

Fonte: Apresentação sobre o complexo articular do tornozelo[9].

Figura 2.10: *Por ordem de apresentação(Da esquerda para a direita) Plantar, Sóleo e Gastrocnémio.*

2.3.2 Funcionamento Muscular

Assim como os nervos, as fibras musculares também produzem um distúrbio elétrico: isso causa-lhes uma contração. Potencias de ação em motoneurónios[9] resultam da libertação do neurotransmissor acetilcolina (ACh) na terminação da fibra nervosa, a junção neuromuscular. A ACh provoca a despolarização da membrana da fibra muscular na junção, e uma onda de despolarização espalha-se ao longo da membrana da fibra muscular. É essa onda de despolarização que causa toda a contração da fibra muscular. Um único motoneurónio inerva diversas fibras musculares, isto é, faz sinapses com muitas fibras musculares, e elas serão ativadas como um grupo. Uma unidade motora consiste em um único motoneurónio junto com fibras musculares supridas por ele, que podem ir de varias centenas num músculo grande para apenas algumas em um musculo menor, envolvidos com controlo e movimentos precisos. O sistema nervoso central tem mais controlo de unidades motoras que fibras musculares individuais. Uma característica fascinante dos músculos é que eles são extremamente bem desenhados para produzir contrações precisas e leves, independentemente da força exigida. Para contrações de baixa intensidade, unidades motoras com um numero pequeno de fibras musculares são ativadas numa baixa frequência. Para o aumento da produção de força, nervos motores disparam em frequências elevadas, e mais unidades motoras são ativadas. Conforme aumenta a necessidade de força, unidades motoras com grande numero de fibras motoras são recrutadas.

Quando muitas fibras musculares se contraem, o sinal elétrico produzido é muito grande, maior do que aquelas das fibras dos motoneurónios que as ativam. Isso ocorre simplesmente porque um motoneurónio inerva muitas fibras musculares. A atividade eléctrica do músculo pode ser medida na superfície do corpo. Atividades de músculos esqueléticos podem ser detetadas usando elétrodos colocados na pele ou agulhas, e então amplificadas, sendo assim exibidas, como por exemplo num eletromiograma ou EMG. O tamanho dos sinais registados depende do número de fibras musculares ativas. Há, portanto, variações consideráveis no tamanho do sinal do EMG, dependendo do tamanho do músculo e da proporção das fibras musculares ativadas. [10]

2.4 Eletromiografia

A eletromiografia (EMG) é uma técnica de monitorização da atividade elétrica muscular. Uma EMG pode ser invasiva ou não invasiva, sendo as eletromiogra-

[9]Neurónio capaz de fazer um músculo entrar em atividade.

fias não invasivas também conhecidas como eletromiografia de superfície (EMGS). O sinal eletromiográfico é o somatório de todos os sinais elétricos detectados numa determinada região do corpo, que pode ser afetado por diversos fatores, como propriedades anatómicas e fisiológicas da região investigada e pela própria instrumentação. A eletromiografia é uma importante ferramenta para a análise clínica da força muscular, podendo fornecer informações como o tempo para a ativação muscular, a duração e intensidade da ativação, avaliar a coordenação motora[11], cálculo da velocidade de condução da fibra muscular[12], estudar *biofeedback*[13], as caracteristicas de estimulação das unidades motoras[14], identificação de doenças neuromusculares específicas[15] e alterações neuromusculares devido à idade[16], ao nível de exercício físico praticado pelo individuo[17] e pela falta de uso de certos musculos[18]. Embora amplamente conhecida e utilizada, trabalhar com sinais EMG não é uma tarefa trivial. Esses sinais têm característica aleatória, variando em amplitude de 0 a 6 mV e em frequência de 0 a 500 Hz, com predominância na faixa de 20 a 250 Hz. Além disso, várias fontes de ruído elétrico causam interferência no sinal EMG, tais como os ruídos de origem fisiológicas, de fontes de energia elétrica e de artefato. O ruído de artefato é causado pelo movimento do elétrodo na pele e é considerado um dos problemas mais críticos na aquisição desse tipo de sinal. Assim, para fazer a aquisição adequada do sinal EMG, a primeira medida a tomar passa por definir qual tipo de informação que se deseja extrair do sinal. Com base na característica desse sinal é possível definir os parâmetros necessários para a construção da instrumentação apropriada, que em geral são: frequência de amostragem, elétrodos, amplificadores de instrumentação, filtros, conversor analógico para digital e sistema de armazenamento de dados.

2.4.1 Tipos de eletromiografia

A eletromiografia consiste em registar a actividade elétrica recorrendo a um sistema de captação do sinal biológico (placa de aquisição dos sinais, amplificador, sistema de canais, elétrodos) e um software para processamento do sinal. Relativamente a eletromiografia, existem dois métodos: a eletromiografia invasiva e a eletromiografia não-invasivas ou eletromiografia de superfície (EMGS).

EMG invasivas

A electromiografia invasiva é a técnica mais usada. O processo de realização de uma EMG envolve a inserção de um elétrodo de agulha (elétrodo) no músculo a estudar, de forma a medir a actividade eléctrica. Esta actividade é visualizada num osciloscópio e é também avaliada através do áudio com um microfone. Dado

que os músculos esqueléticos são geralmente extensos, poderá nalguns casos ser necessário a inserção da agulha em mais do que um local de modo a obter uma EMG mais eficaz e informativa. Depois da inserção da agulha e do estudo em repouso, é pedido ao paciente para contrair o músculo (por exemplo, dobrar a perna ou o braço). A realização de uma EMG envolve dor e inequivocamente algum desconforto na altura da inserção da agulha. Na Figura 2.11 está representado o método de electromiografia invasiva.

Fonte: Classroom orange[19].

Figura 2.11: *Electromiografia invasiva.*

Eletromiografia de Superfície

A electromiografia de superfície, ou electromiografia não-invasiva, é feita através de elétrodos que são usados na superfície da pele (daí o nome electromiografia de superfície), em vez de serem introduzidos directamente no músculo. Os elétrodos de superfície (ativo, de referência e terra) são capazes de registar de forma mais generalizada a actividade de um maior número de fibras musculares, quando são ativadas em condições de esforço mínimo, médio e máximo. O elétrodo activo deve estar situado sobre a região a ser estudada, o eléctrodo de referência pode ser colocado sobre um grupo muscular distinto a ser estudado e o eléctrodo terra em qualquer outro lugar que não seja entre o elétrodo de referência e o eléctrodo activo. O eléctrodo activo é o que capta a actividade do músculo. O eléctrodo de referência serve para distinguir o conjunto dos músculos que irão ser estudados e o terra reduzir o ruído inerente a operação a ser efetuada e também para dar segurança ao paciente. Com

este método o nível de desconforto do utente é muito reduzido. Atualmente é usada uma variação deste método que se baseia na colocação de vários elétrodos, em forma de matriz ou malha, de forma a que seja possível ler os impulsos numa área maior do corpo em vez de apenas num músculo. É possível visualizar, na Figura 2.12, como o método da electromiografia de superfície é aplicado no paciente.

Fonte: Laboratório de biomecânica[20].

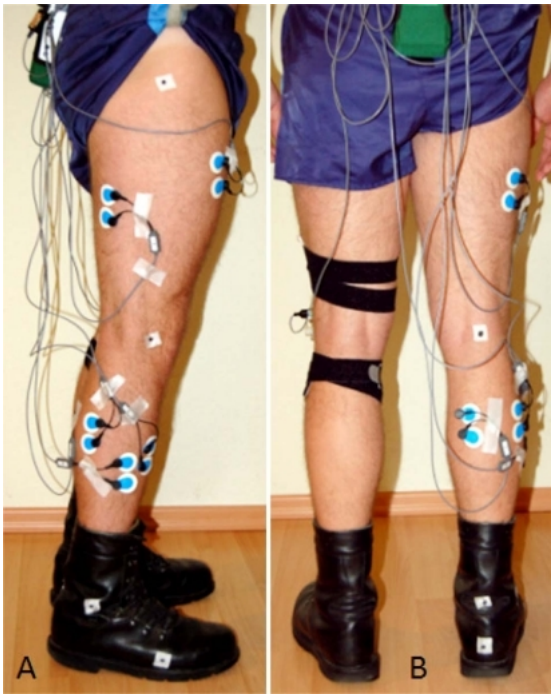


Figura 2.12: *Electromiografia de Superfície.*

2.4.2 Tipos de elétrodos

Existem dois tipos de elétrodos, os elétrodos que são colocados à superfície e os elétrodos de inserção. Os elétrodos de inserção dividem-se em dois tipos, de agulha e de "*fine wire electrodes*". Em relação aos elétrodos de superfície, também são divididos em dois tipos: os que possuem e os que não possuem uma substância em forma de gel. Os quarto tipos de elétrodos (agulha, "*fine wire*"e os de superfície) são apresentados de seguida.

Elétrodo de agulha

Os elétrodos de agulha são amplamente utilizados nos procedimentos clínicos para realizar as avaliações neuromusculares. A extremidade do elétrodo de agulha está vazia, é usada como uma superfície de detecção e possui um fio de isolamento na cânula[10]. Relativamente à qualidade do sinal é comparativamente melhor do que a obtida pelos restantes tipos disponíveis uma vez que realiza a leitura do impulso diretamente do interior do músculo sem a interferência, por exemplo da pele como num método que será mais a frente exposto. Os elétrodos de agulha têm duas vantagens: um é que a sua área de captação é relativamente pequena, o que permite que o elétrodo consiga detectar *Motor Unit Action Potential* (MUAPs) individuais durante as contrações de baixa de intensidade, a outra vantagem é que os elétrodos podem ser convenientemente inseridos dentro do músculo (após a inserção) o que permite uma melhor qualidade do sinal obtido. Três tipos de elétrodos de agulha com diâmetros diferentes são mostrados na Figura 2.13:

Fonte: Neuro base[21].

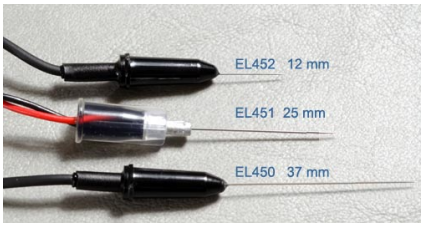


Figura 2.13: *Elétrodo de agulha.*

Fine wire electrodes

Fine wire electrodes são caracterizados pelos seus diâmetros pequenos, por serem altamente não-oxidantes e possuírem um fio isolado. Os *fine wire electrodes* são extremamente finos o que leva a que sejam facilmente implantados e retirados dos músculos, tornado-os geralmente menos dolorosos do que os elétrodos de agulha que estão inseridos no músculo durante toda a duração do procedimento. Um *fine wire electrode* é apresentado na Figura 2.14:

[10]instrumento tubiforme, que se aplica em operações terapêuticas, cirúrgicas, de higiene, etc., por vezes adaptado a seringas e irrigadores.

Fonte: INTECH[22].

Figura 2.14: *Fine wire electrodes.*

Gelled EMG Electrodes

Os *gelled EMG electrodes* contém uma substância eletrolítica gelificada que funciona como uma interface entre a pele e elétrodos. As reações de oxidação e redução ocorrem na junção do elétrodo metálico. A camada de Cloreto de prata (AgCl) permite que a corrente a partir do músculo consiga passar mais livremente através da junção entre o electrólito e o elétrodo. Este método introduz menos ruído elétrico na medição, quando comparado com os elétrodos equivalentes. Devido a este facto, os elétrodos AAgCl são usados em mais de 80% das electromiografias de superfície[23]. Na Figura 2.15 é visível um *gelled EMG electrode*:

Fonte: Bio-medical Instruments[24].

Figura 2.15: *Gelled EMG Electrodes.*

Dry EMG electrodes

Os *Dry EMG electrodes* não necessitam de uma interface de gel entre a pele e a superfície de detecção. Os elétrodos em barra e a agrupação dos elétrodos em forma de uma matriz são exemplos de *dry EMG electrodes*. Estes elétrodos podem conter mais do que uma superfície de detecção e são geralmente mais pesados ($> 20g$), relativamente aos *Gelled EMG Electrodes* ($< 1g$), fazendo com que este aumento da massa inercial possa causar problemas para a fixação dos elétrodos. Um *dry EMG electrode* é mostrado na Figura 2.16.

Fonte: INTECH[22].

Figura 2.16: *Dry EMG electrodes.*

2.4.3 Categorias de elétrodos de superfície

Existem duas categorias de elétrodos de superfície: Passivo e ativos. Uma breve explicação sobre cada um é dada a seguir. Os elétrodos de superfície passivos devem ser ligados a um circuito externo de amplificação para a aquisição correcta do sinal de EMG. Os elétrodos passivos podem ser descartáveis ou reutilizáveis. Os elétrodos de superfície ativos já incluem um circuito pré-amplificador não sendo necessário proceder a montagem de um circuito de amplificação de sinal para melhorar a qualidade do sinal recebido. Estes elétrodos geralmente são do tipo de elétrodos *dry EMG electrodes*.

2.4.4 Regras para a colocação dos elétrodos

Durante algum tempo, os estudos através da eletromiografia de superfície (EMGS) não recebiam tratamento de protocolos quanto às recomendações dos cuidados e normas durante a sua operação, remetendo as pesquisas a um caráter amador e pouco científico. Tentando ultrapassar esta problemática, no final do século passado, alguns cientistas europeus reuniram-se e criaram o consórcio europeu denominado de *Surface EMG for the Non-Invasive Assessment of Muscles (SENIAM)*[25], padronizando assim, os locais de colocação dos elétrodos, tipos de elétrodos e distância entre os mesmos. Neste momento a SENIAM possui recomendações para a colocação dos

sensores em 30 músculos individuais. As recomendações para os músculos individuais estão divididas em cinco partes do corpo em que os músculos estão localizados: Ombros e Pescoço, Tronco e zona baixa das costas, Braço e Mão, Coxa e parte superior da perna e parte inferior da perna e pé. Para efeito de efeciência e objetividade, apenas serão demonstrados nesta tese os locais recomendados relativamente a parte do corpo, parte inferior da perna e pé, as restantes zonas podem ser consulatdas em [25]. Na parte inferior da perna e pé, existem seis músculos onde podem ser colocados os elétrodos, tibial anterior, perónio longo, perónio curto, sóleo, gastrocnémio médio e gastrocnémio lateral. Como exemplo ilustrativo será apresentada a recomendação da localização dos elétrodos relativamente ao músculo gastrocnémio lateral (Figura 2.17).

Fonte: SENIAM[26].

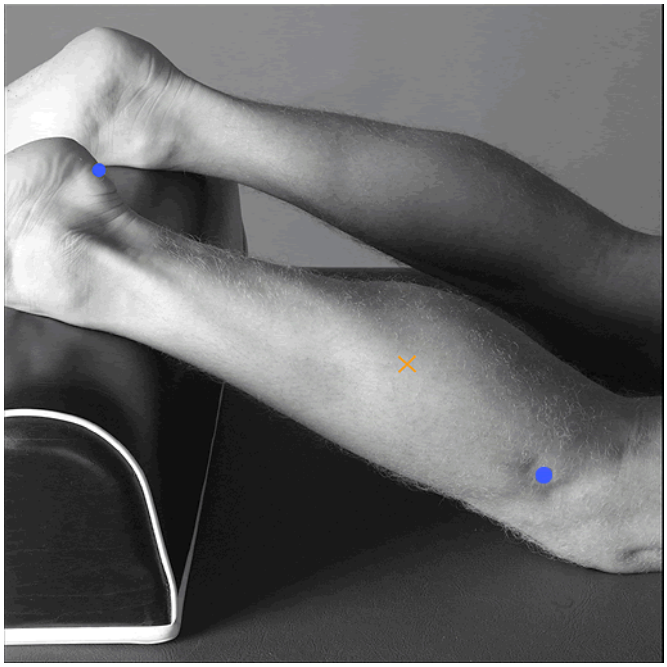


Figura 2.17: *Recomendação da localização dos elétrodos no músculo gastrocnémio lateral.*

A zona de colocação dos elétrodos para medir a atividade elétrica deste músculo exposto, está representada com um ponto azul. Para além da imagem, existe também uma tabela com a informação sobre o músculo, (nome e anatomia) e sobre o procedimento da colocação do sensor. A tabela acerca do músculo da Figura 2.17 é a tabela 2.1.

Tabela 2.1: *Informação do Músculo e Procedimento da Colocação do Sensor.*

Músculo	
Nome	Gastrocnémio
Subcategoria	Lateral
Anatomia do Músculo	
Origem	Côndilo lateral e superfície posterior do fémur, cápsula da articulação do joelho.
Colocação	Parte média da superfície posterior do calcâneo.
Função	Flexão da articulação do tornozelo e auxiliar na flexão da articulação do joelho.
Procedimento da colocação do sensor	
Postura inicial	Deitado sobre a barriga com a face para baixo, o joelho estendido e o pé em direção à extremidade da mesa.
Dimensão do elétrodo	Tamanho máximo na direcção das fibras musculares: 10 mm.
Distância do elétrodo	20 mm
Colocação do elétrodo	
- **localização**	Elétrodos são colocados em 1/3 da linha entre a cabeça do perónio e do calcanhar.
- **orientação**	Em direção à linha entre a cabeça do perónio e do calcanhar.
- **fixação sobre a pele**	Fita (Frente e verso).
- **área de colocação**	Sobre ou a volta do tornozelo.
Teste clínico	Flexão plantar do pé com ênfase em puxar o calcanhar para cima mais do que empurrar o ante pé para baixo. Para a pressão máxima nesta posição, é necessário aplicar pressão contra a parte dianteira do pé, também como contra o calcâneo.
Observações	As recomendações SENIAM também incluem um procedimento de colocação de um sensor separado para o gastrocnémio médio.

2.4.5 Preparação da zona de leitura dos elétrodos

A aplicação de elétrodos EMG de superfície requer uma preparação adequada da pele antes de ser realizado o procedimento de leitura dos elétrodos. De forma a

obter uma boa qualidade de sinal, a impedância da pele deve ser consideravelmente reduzida. Para isso, as células mortas sobre a pele, por exemplo, os pelos, devem ser completamente removidos do local onde os elétrodos irão ser colocados. É aconselhável utilizar um gel, ou álcool, para reduzir a camada seca da pele, de forma a que não haja humidade na pele.

2.4.6 Características do sinal dos elétrodos

Antes de avançarmos para a fase de aquisição de sinal, é muito importante conhecer o sinal de EMG e as várias preocupações e factores que afectam a qualidade do sinal. A amplitude do sinal encontra-se entre 1-10 mV, tornando-se um sinal consideravelmente fraco. O sinal encontra-se na gama de frequências de 0-500 Hz contudo é usual entre os 50-150 Hz. Este sinal é altamente influenciado pelo ruído elétrico, que pode ser causado a partir de várias fontes. O ruído ambiente pode ser causado pela radiação electromagnética, por exemplo, dispositivos de transmissão de rádio, lâmpadas fluorescentes, etc e encontra-se na gama dos 50-60 Hz. Outra forma de ruído que também pode ser considerada é gerada a partir do movimento de objetos. As duas principais fontes deste ruído são a instabilidade da interface de pele com o elétrodo e movimento do cabo do elétrodo, este ruído encontra-se principalmente na faixa dos 0-20 Hz.

2.5 Circuito de aquisição de sinal

O sinal EMG é adquirido através da técnica de amplificação. No inicio, o conceito de amplificador operacional (AO) era de realizar operações analógicas (computadores analógicos). Atualmente, os Amplificador operacional possuem um ganho elevado, impedância de entrada elevada, grande fiabilidade e baixo custo.

O amplificador operacional, Figura 2.18, é um componente que apresenta duas entradas e uma saída. Uma das entradas representada com o sinal $-$ é denominada entrada inversora e a outra entrada apresentada pelo sinal $+$ corresponde à entrada não-inversora, estes dois sinais de entrada são conhecidos por entrada diferencial. Os amplificadores operacionais têm na realidade um ganho de tensão na ordem dos 100 000, tendo sido concebidos para funcionarem com realimentação negativa e portanto para ganhos de tensão menores.

Fonte: Introdução aos amplificadores operacionais[27].

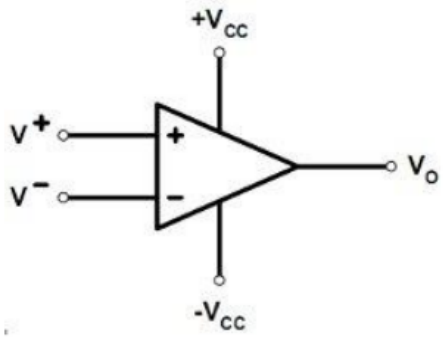


Figura 2.18: *Amplificador operacional.*

Uma das características que em o amplificador operacional difere do amplificador de instrumentação, é que o amplificador de instrumentação deve ser capaz de rejeitar sinais em modo comum. Outra diferença entre os dois, é que os amplificadores operacionais necessitam de uma malha de realimentação seja incorporada para que estes operem de forma controlada, no caso dos amplificadores de instrumentação já possuem estas estruturas intrinsecamente realimentadas.

A amplificação diferencial é obtida com a ajuda de um amplificador de instrumentação de alta impedância de entrada. Um amplificador de instrumentação com três amplificadores é mostrado na Figura 2.19.

Fonte: Introdução aos amplificadores operacionais[27].

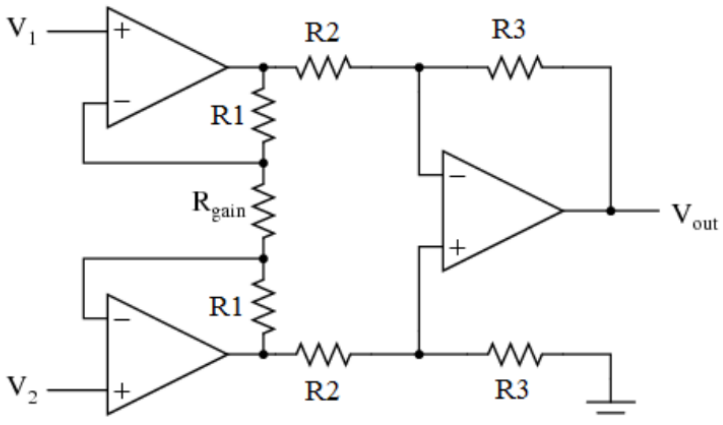


Figura 2.19: *Amplificador de instrumentação.*

O amplificador de instrumentação realiza a amplificação diferencial subtraindo-se as tensões de V1 e V2, desta forma, o ruído que é comum ao V1 e V2 é eliminado. A tendência de um amplificador diferencial de rejeitar sinais comuns a ambas as entradas é determinado pela rejeição em modo comum, do inglês, *Common Mode Rejection Ratio* (CMRR). O CMRR de 90 dB é suficiente para conseguir eliminar sinais comuns de amplificadores de instrumentação, mas com a mais recente tecnologia, embora cara, podemos obter um CMRR de 120 dB. Contudo existem razões para não levar o CMRR até ao limite, uma vez que o ruído eléctrico detectado pelos elétrodos pode não estar em fase. O ganho para o amplificador de instrumentação pode ser configurado usando uma única resistência (Rgain). A equação de ganho e a equação da tensão de saída do amplificador de instrumentação é dada pela equação, (2.1) e (2.2), respetivamente.

$$Gain = (1 + \frac{2R1}{Rgain})\frac{R3}{R2} \tag{2.1}$$

$$Vout = (V2 - V1) \times Gain \tag{2.2}$$

Um pequeno ganho de 5 ou 6 é o recomendado para obter a aquisição de sinal. A eletromiografia de superfície pode ser realizada através de três configurações diferentes: monopolar, bipolar e multipolar.

2.5.1 Configuração monopolar

A configuração monopolar é implementada usando um único elétrodo sobre a pele com um elétrodo de referência, como mostrado na Figura 2.20. A grande vantagem deste método é a sua simplicidade, mas não é o mais recomendado, uma vez que detecta todos os sinais elétricos na vizinhança da superfície onde está a decorrer o procedimento.

Fonte: NR sing Inc[28].

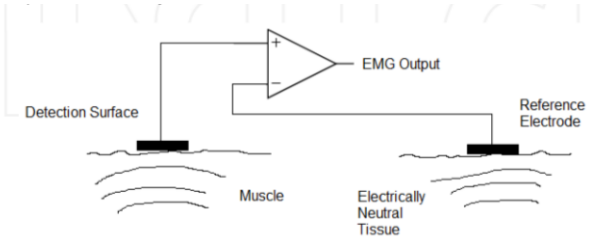


Figura 2.20: *Configuração monopolar.*

2.5.2 Configuração bipolar

A configuração bipolar é usada para a aquisição do sinal de EMG usando dois elétrodos de superfície, mais o elétrodo de referência. Os sinais a partir dos elétrodos estão ligados a um amplificador diferencial e estes elétrodos estão separados, apenas, de cerca de 1-2 cm um do outro. O amplificador diferencial elimina os sinais de ruído comuns a ambas as entradas e, de seguida, amplifica a diferença. As limitações da configuração monopolar são resolvidas com esta configuração, sendo esta a configuração mais utilizada. A configuração do elétrodo bipolar de EMG é mostrado na Figura 2.21.

Fonte: NR sing Inc[28].

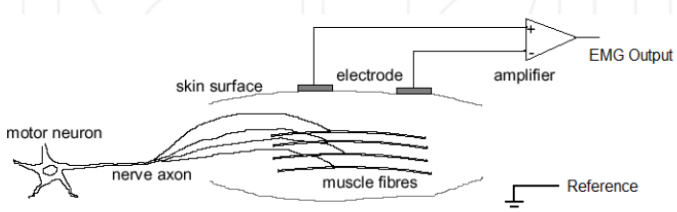


Figura 2.21: *Configuração bipolar.*

2.5.3 Configuração multipolar

Esta configuração utiliza mais do que dois elétrodos de superfície para adquirir o sinal de EMG com o ajuda de um elétrodo de referência. Esta configuração reduz ainda mais a diafonia[11] e o ruído, tornando assim o sinal de EMG obtido com maior qualidade. O sinal de três ou mais elétrodos de superfície, são colocados entre 1-2 cm um do outro e passam através de mais do que dois níveis de amplificação diferencial. Por exemplo, se três elétrodos são utilizados, a técnica de diferencial dupla é aplicada. Esta configuração é utilizada em pesquisas mais abrangentes, como por exemplo, para estudar a orientação das fibras musculares.

2.6 Filtragem do sinal

Esta secção irá discutir as considerações do circuito de aquisição de sinal, de forma a atingir o sinal melhor possível a partir dos músculos do corpo humano, com um detalhe minucioso. O circuito básico para a aquisição de sinal já foi explicado em detalhe na secção anterior (ver secção 2.5). Nesta secção, irão ser discutidos

[11] Perturbação da perceção auditiva, por intervenção de sons parasitas.

os possíveis circuitos que poderão vir a ser implementados, após já termos o sinal amplificado.

2.6.1 Tipos de Filtros

Como referido anteriormente, existem alguns aspetos a ter em atenção relativamente à obtenção adequada do sinal proveniente dos elétrodos. Quando o elétrodo é corretamente colocado e o sinal é extraído, o ruído desempenha um papel importante na qualidade do sinal obtido. Para melhorar o sinal, este tem de ser devidamente filtrado, mesmo após ter sido realizada a amplificação diferencial. As frequências do ruído que contaminam o sinal original podem ser elevadas, também como baixas. O ruído de baixa frequência pode ser causado pelo *offset* do amplificador DC, fixação do sensor na pele e pelas flutuações de temperatura. O ruído de alta frequência pode ser causado pela atividade nervosa[12] e da interferência das transmissões de rádio, computadores, telefones, etc. A fim de remover o ruído introduzido por certas frequências, os filtros de passa-alto, passa-baixo, passa-banda filtro de Notch ou rejeita-banda serão apresentados de seguida[29][30][31].

2.6.2 Filtro Passa-Alto

Um filtro passa-alto é usado para remover o componente de baixa frequência de um determinado sinal. O termo "*cut-off*", denominado de "f_c", é a frequência abaixo da qual todas as frequências são eliminadas. Todas as frequências acima de "f_c"são aceites. Uma resposta do filtro passa-alto é mostrada na Figura 2.22.

Fonte: Electrical Engineering Dictionary[32].

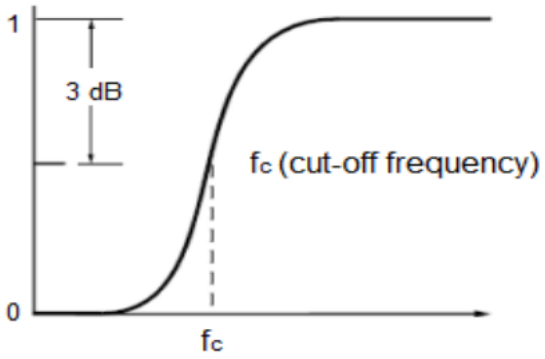


Figura 2.22: *Resposta do filtro Passa-Alto.*

[12]Como atividade nervosa, entenda-se a passagem de corrente que percorre o ser humano.

Um filtro passa-alto pode ser desenvolvido usando uma resistência e um condensador, tornando-o num circuito resistência-condensador (RC). Este circuito é um filtro passa-alto de primeira ordem. É o mais simples filtro de frequências altas possível. O filtro é demonstrado na Figura 2.23.

Fonte: Chegg study[33].

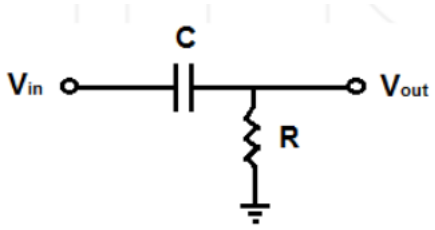


Figura 2.23: *Filtro Passa-Alto de primeira ordem.*

A frequência de corte do filtro passa-alto é dada pela equação (2.3).

$$f_c = \frac{1}{2\pi RC} \tag{2.3}$$

Um filtro passa-alto de segunda ordem também pode ser útil. O circuito utiliza dois filtros de primeira ordem em série e é auxiliado por um amplificador operacional. O circuito é dado pela Figura 2.24.

Fonte: INTECH[?].

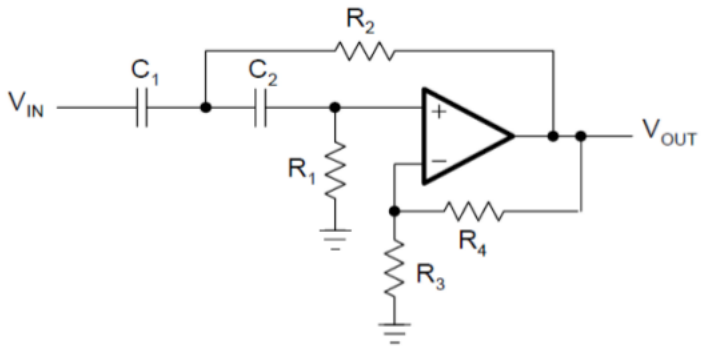


Figura 2.24: *Filtro Passa-Alto de segunda ordem.*

Para este circuito, se $R_1 = R_2; C_1 = C_2$, em seguida, f_c é dado como:

$$fc = \frac{1}{2\pi RC} \tag{2.4}$$

R_3 e R_4 são opcionais e são necessários para ajustes de ganho:

$$A_0 = 1 + \frac{R4}{R3} \tag{2.5}$$

Usando um filtro de segunda ordem é recomendado uma vez que fornecem um *roll-off* de $40dB/dec$, em comparação com os $20dB/dec$ dados pelo filtro de primeira ordem. O uso de componentes activos pode isolar o filtro do resto do circuito.

2.6.3 Filtro passa-baixo

O conceito de filtros passa-baixo é totalmente oposto ao conceito dos filtros de passa-alto. Nestes filtros, as frequências menores do que a frequência de corte são transmitidas e as frequências mais elevadas são removidas. Uma resposta do filtro passa-baixo é mostrada na Figura 2.25.

Fonte: Electrical Engineering Dictionary[34].

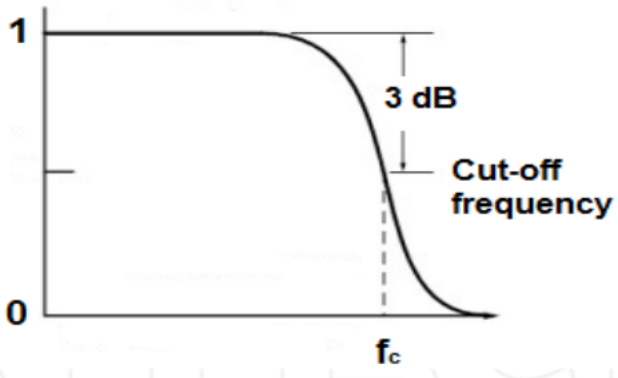


Figura 2.25: *Resposta do filtro passa-baixo.*

O filtro de passa-baixo mais simples que pode ser desenvolvido, inclui uma resistência e um condensador, e é chamado de circuito RC de primeira ordem. O circuito do filtro de passa-baixo de primeira ordem é mostrado na Figura 2.26.

Fonte: Hyperphysics, com edição do autor[35].

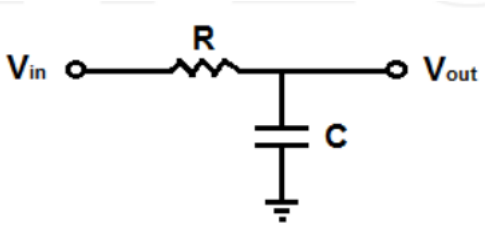


Figura 2.26: *Filtro Passa-baixo de primeira ordem.*

A equação da frequência de corte para o circuito da Figura 2.6 é a mesma que a da equação(2.3). Um filtro passa-baixo de segunda ordem pode ser mais eficaz em comparação com um de primeira ordem. Este filtro pode ser implementado em cascata com dois filtros de primeira ordem ligados a um amplificador operacional. O circuito é apresentado na Figura 2.27.

Fonte: INTECH[22].

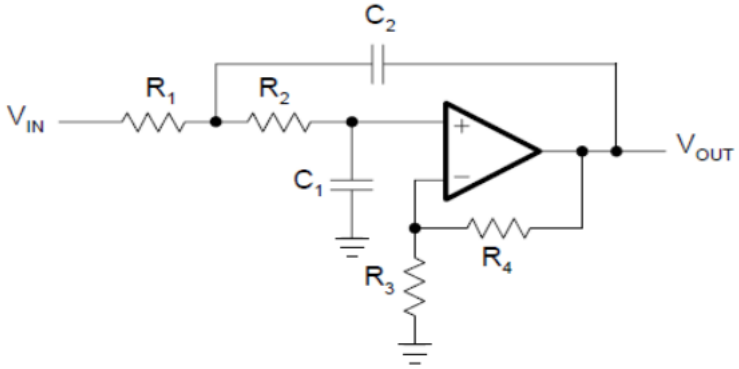


Figura 2.27: *Filtro Passa-baixo de segunda ordem.*

Para $R_1 = R_2$ e $C_1 = C_2$, a frequência de corte do circuito na Figura 2.6 é a mesma que a da equação (2.4). R_3 e R_4 são opcionais, caso sejam necessários ajustes do ganho, como dado na equação (2.5). Um filtro passa-baixo de segunda ordem é novamente recomendado, relativamente ao de primeira ordem pelas mesmas razões mencionadas para um filtro passa-alto de segunda ordem.

2.6.4 Filtro passa-banda

Como mencionado anteriormente, para a transmissão de um sinal de eletromiografia com uma qualidade elevada, as frequências do ruído, quer altas e baixas, devem ser eliminadas. Para isso, apenas um intervalo de frequências específicas deve passar. Isto pode ser possível com a ajuda de um filtro passa-banda. Uma resposta do filtro passa-banda é mostrada na Figura 2.28.

Fonte: Electrical Engineering Dictionary, com edição do autor[36].

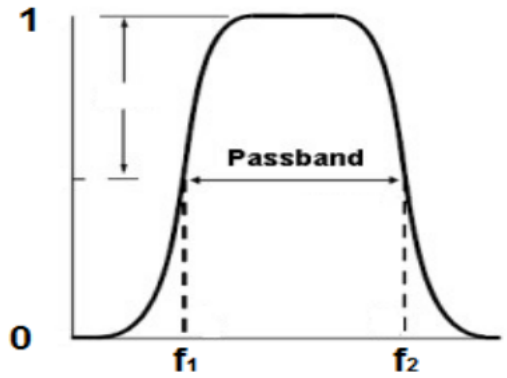


Figura 2.28: *Resposta do filtro passa-banda.*

A região de frequência onde a resposta do sinal é "1"é chamada de "banda passante"e no caso do filtro de passa-banda, é entre f_1 e f_2. Um filtro passa-banda pode ser concebido através de uma ligação de um filtro passa-baixo e um filtro passa-alto em série. Ao seleccionar os valores adequados de resistência (R) e condensador (C), podemos desenvolver um filtro de passa-banda que pode transportar mais eficazmente o sinal de eletromiografia. Recomenda-se que, para a eletromiografia, f_1 deve ser de 65-70 Hz e f_2 deve-se situar entre 150-180 Hz.

2.6.5 Filtro de Notch ou rejeita-banda

Em processamento de sinais, um filtro rejeita-banda é um filtro que permite a passagem da maioria das frequências, porém não deixa passar aquelas frequências que estejam numa determinada faixa definida da quando da utilização do filtro. O principio de funcionamento é o oposto do filtro passa-banda, mencionado na subsecção anterior. Normalmente este filtro é implementado para rejeitar o ruído da rede, ou seja o ruído que possa existir na área onde será utilizado, as frequências que se tentam anular estão entre os 50 Hz, 100 Hz, 150 Hz e 200 Hz.

2.7 Conversão analógica para digital

O processo de digitalização do sinal analógico é levada a cabo com um conversor de analógico para digital. Hoje em dia, o ADC tornou-se num componente comum nos dispositivos eletrónicos modernos e o seu uso tornou-se altamente variado e generalizado. Antes de utilizar o ADC, as especificações, vantagens e limitações têm de ser analisados a fim de selecionar o mais apropriado para a aplicação. Da mesma forma, considerações importantes têm de ser tidas em conta durante a conversão de sinais de EMG para o formato digital. Um ADC comum tem uma gama específica de conversão, existem níveis máximos e mínimos definidos para um ADC sobre os quais este pode operar. Um ADC pode converter o sinal analógico ao longo de um certo número de bits. O número de bits que um ADC pode converter é conhecido como o seu "esquema de quantização". Se um ADC tem uma gama definida e um esquema de quantificação de 'N' bits, em seguida, a resolução do ADC pode ser dada como (equação 2.6):

$$V_{resolution} = \frac{V_{range}}{2^n} \tag{2.6}$$

Para converter um sinal de EMG em formato digital, devem ser conhecidas os três parâmetros: quantização, faixa de conversão e a taxa de amostragem. O número de bits, em que um sinal analógico pode ser convertido para o formato digital por um ADC, é conhecido como quantização. O valor máximo da tensão que um ADC pode converter digitalmente define o intervalo de um ADC. A taxa de amostragem significa o número de amostras que um ADC pode converter em um segundo. Após o sinal de EMG ter sido amplificado até um nível apropriado, a gama de um ADC deve ser seleccionada de modo a que possa compreender um certo nível de tensão. O número de quantização é importante, pois determina a resolução do ADC, quanto maior o número de bits de quantização, menor será a resolução do ADC. A taxa de amostragem do ADC é também bastante importante, esta taxa deve ser mantida a maior possível de forma a que a perda de dados do sinal de EMG seja mínima. Os ADCs estão agora disponíveis em microcontroladores (ver secção 2.8) e têm taxas de amostragem superiores a 1000 kSPS[13] e esquemas de quantização de mais de 24 bits.

[13] *Kilosample(s) per second.*

2.8 Microcontroladores

Um microcontrolador[37] é um microcomputador compacto projetado para controlar o funcionamento de sistemas embebidos que podem ser aplicados em varias áreas, como na robótica, domótica, mecânica, bioengenharia, segurança. Analisando os sistemas existentes no mercado que utilizam microcontroladores, pode notar-se que as suas aplicações são cada vez mais exigentes em termos computacionais, de custo e de consumo energético, o que obriga a que os micro controladores utilizados sejam cada vez mais eficientes. Para além disto tem-se assistido a uma necessidade de que estes dispositivos se liguem uns aos outros quer por Universal Serial Bus (USB), Ethernet ou wireless, razão pela qual os micro controladores deverão incluir periféricos que permitam tais funcionalidades. Os utilizadores têm também obrigado a alterações. As aplicações são cada vez mais complexas, as interfaces desenvolvidas cada vez mais intuitivas e acessíveis, requisitos multimédia também mais exigentes e uma necessidade de convergência de funcionalidades entre plataformas são algumas das exigências actuais. A arquitectura comum de um micro controlador é a seguinte, (Figura 2.29):

Fonte: Microcontroller garden, com edição do autor[38].

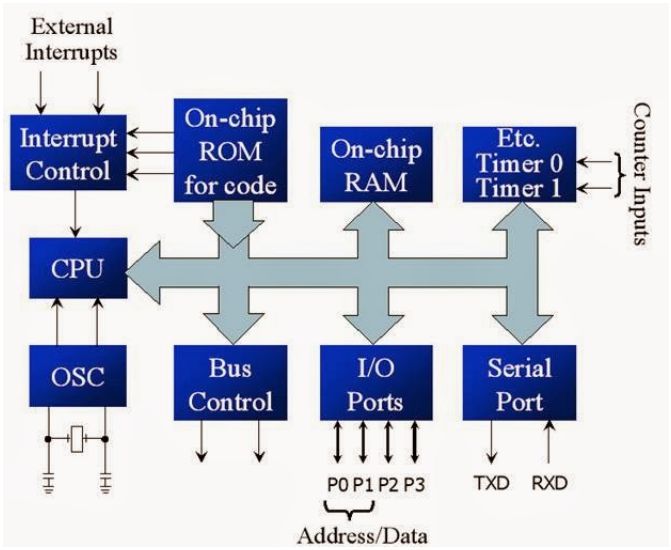


Figura 2.29: *Exemplo da arquitetura de um microcontrolador.*

Relativamente a um microprocessador, que geralmente possui um sistema operativo e uma BIOS, um micro controlador o utilizador é que o programa para o efeito desejado dentro do sistema em que se encontra, podendo eventualmente correr um

sistema operativo chamado de *Real Time Operative System* (RTOS)[39]. Normalmente este sistema operativo só é usado para micro controladores com um poder de processamento maior e com mais *Timers* disponíveis, como no caso dos ARMs. A arquitetura de um micro controlador depende da aplicação para que é construído. Por exemplo, alguns modelos incluem o uso de mais do que uma random acess memory (RAM), read-only memory (ROM) e possuem um maior ou menor número de portas de entrada/saída, *in/out* (I/O).

- Uma *central processing unit* (CPU), que vão desde simples 4 bits até processadores complexos de 64 bits;
- Periféricos, tais como *timer*[14], contadores de eventos e *watchdog*[15];
- RAM para armazenamento de dados. Os dados são armazenados na forma de registos;
- ROM, erasable programmble read-only memory (EPROM), eletrical-erasble programmble read-only memory (EEPROM) ou memória flash permitem o armazenamento de instruções de programa e dados;
- Capacidades de programação;
- Portas de entrada/saída, permitem a sua interligação com dispositivos periféricos para entrada ou saída de dados;
- Um relógio interno ou externo, que varia em frequência, desde 1 MHz até 168 MHz;
- Conversores analógico para digital;
- Portas série;
- Barramento de dados para transportar informações.

Dentro dos microcontroladores existem vários tipos disponíveis no mercado, que variam nos 10 parâmetros mencionados anteriormente. Os microcontroladores mais usados são: a atmega, o PIC e o ARM.

[14]Timer é um relógio que controla a sequência de um evento durante a contagem em intervalos fixos de tempo.

[15]Watchdog é um hardware ou software que gera uma interrupção do *timer* que reinicia o sistema

2.8.1 AVR ATmega

AVR ATmega é um microcontolador designado de, Computador com Conjunto Reduzido de Instruções, do inglês *Reduced Instruction Set Computer* (RISC), com uma arquitetura Havard modificada de 8 e de 32 bits, desenvolvido pela Atmel em 1996. A sua principal caracteristica inicialmente foi a possibilidade de armazenar a programação utilizando uma memoria flash para atingir tal feito.

Ao longo do decorrer do tempo, a Atmel foi lançando novas ATmegas, que levou ao serem criadas familias de microcontroladores, dentro da Atmel existem quatro grupos de classificação: tinyAVR, megaAVR, XMEGA e Atmel At94k[40].

As principais diferenças entre estes grupos de microcontradores da Atmel estão demonstradas na tabela 2.2:

Tabela 2.2: *Microcontradores da Atmel*

tinyAVR	megaAVR	XMEGA	FPSLIC
Memória do programa cerca de 0,5–16 kB	Memória do programa cerca de 4–256 kB	Memória do programa cerca de 16–384 kB	FPGA com cerca de 5000 a 40000 portas lógicas
Cerca de 6–32-pinos	Cerca de 28–100 pinos	Cerca de 44–100 pinos	SRAM para o codigo da AVR, ao contrário das restantes AVRs
Conjunto limitado de periféricos	Conjunto de instruções extenso	Conjunto de instruções mais extenso como o uso de DMA, "Sistema de eventos", e suporte a criptografia	Pode atingir uma frequência de relógio até 50 MHz
	Conjunto extenso de periféricos	Conjunto extenso de periféricos com ADC	

Uma das ATmegas normalmente usadas é a ATmega32, que faz parte do grupo das megaAVR. Existem várias características deste microcontrolador entre as quais:

- Microcontrolador de 8 *bits*;
- 32kbytes de memória *Flash*;
- 2048bytes de memória *RAM*;
- 1024bytes de memória *EEPROM*;
- Pode ser alimentado entre 4,5 V e 5,5 V;

- Corrente máxima fornecida por qualquer pino I/O de 25 mA;
- Corrente máxima fornecida e de entrada em todas as portas I/O de 200 mA;
- Pode operar numa frequência até 16 MHz;
- Oscilador interno de 8 MHz;
- 40 Pinos, dos quais 32 podem ser configurados como I/O, onde 8 destes podem ser conversores A/D de 10 *bits* de resolução com tempo de aquisição programável.

Na Figura 2.30 está o mapeamento dos pinos relativos à ATmega32:

Fonte: Manual da ATmega32[41].

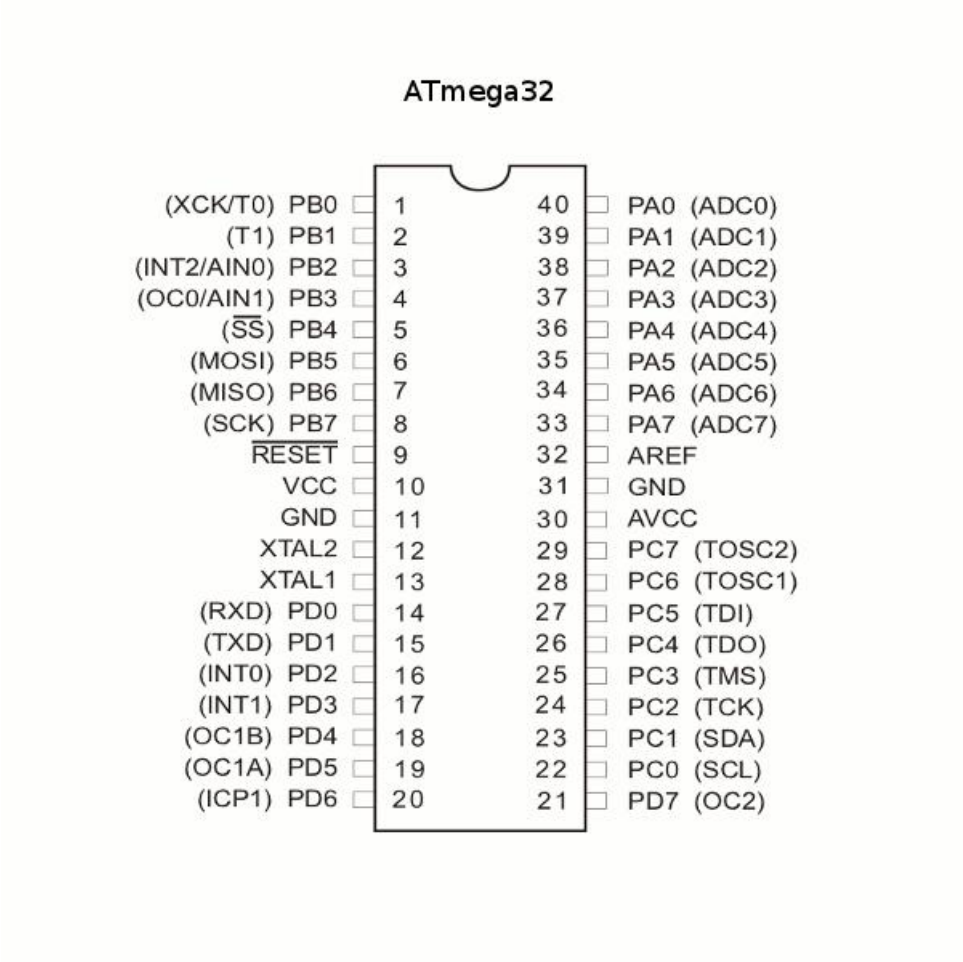


Figura 2.30: *ATmega32.*

2.8.2 PIC

O PIC trata-se de um microcontrolador desenvolvido pela Microchip, é construído com base na arquitectura Harvard com instruções do tipo RISC, tal como na ATmega, é uma linha de arquitectura de processadores que favorece um conjunto simples e pequeno de instruções que levam aproximadamente a mesma quantidade de tempo para serem executadas.[42] O microcontrolador PIC escolhido para ser apresentado nesta subsecção foi o PIC18F4550. Na Figura 2.31 está o mapeamento dos pinos relativos ao PIC18F4550:

Fonte: Manual da PIC18F4550[43].

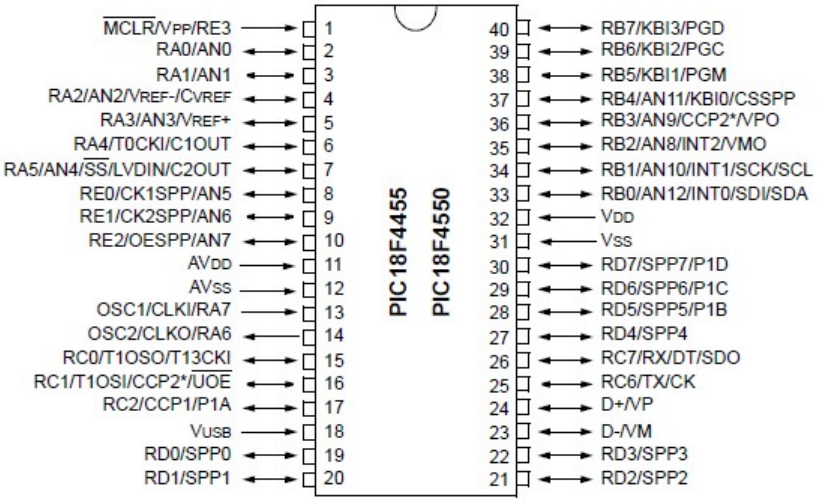


Figura 2.31: *PIC18F4550.*

Existem várias características deste microcontrolador entre as quais:

- Microcontrolador de 8 *bits*;
- 32kbytes de memória *Flash*;
- 2048bytes de memória *RAM*;
- 256bytes de memória *EEPROM*;
- Pode ser alimentado entre 4,5 V e 5,5 V;
- Corrente máxima fornecida por qualquer pino I/O de 25 mA;
- Corrente máxima fornecida e de entrada em todas as portas I/O de 200 mA;

- Pode operar numa frequência até 48 MHz (12 MIPS - milhões de instruções por segundo;
- Oscilador interno de 8 MHz;
- 40 Pinos, dos quais 35 podem ser configurados como I/O, onde 13 destes podem ser conversores A/D de 10 *bits* de resolução com tempo de aquisição programável.

2.8.3 ARM

Os microcontroladores ARM (*Acorn RISC Machine*), fazem uso da arquitetura ARM, possuem um processador de 32/64 bits e normalmente são usados com maior foco em sistemas embebidos de grande escala. Os processadores ARM são conhecidos pela sua versatilidade, pois possuem poucas instruções para programação. Atualmente são encontrados em *smartphones*, *tablets*, calculadoras, módulos de desenvolvimento como *raspberries* e *odroids* e aplicações industriais. Atualmente existem 9 famílias de micro controladores, ARM7 Thumb, ARM9 Thumb, ARM9E, ARM10E, ARM11, ARM15, SecurCore, OptimoDE Data Engine, e Cortex Family.

O processador ARM Cortex-M3 (primeiro ARM de 2006), faz parte da família Cortex e foi lançado com o intuito de alcançar o mercado dos processadores de 32-bit. É um processador com excelente desempenho e com algumas características até então apenas disponíveis em processadores de um gama alta.

O micro controlador escolhido para ser usado como referência foi o STM32Primer, o seu *pinout* está representado na Figura 2.32. Este modelo de ARM CORTEX-M3, apresenta as seguintes características:

- Núcleo RISC 32-bit ARM CORTEX-M3;
- Oscilador interno 8 MHz / Oscilador externo 4-16 MHz;
- Máxima frequência de funcionamento – 72 MHz (utilizando PLL);
- Memória Flash – 128 KBytes (16 K x 64-bits);
- Memória SRAM – 20 KBytes;
- Número de GPIOS – 51 (PA0 .. 15, PB0 .. 15, PC0 .. 15, PD0 .. 2);
- Temporizadores (16 bit) - TIM1, TIM2, TIM3 e TIM4;
- ADC - 2 x ADC 12 bits (até 16 canais);

- Comunicações: 2 x SPI; 2 x I2C; 3 x USART; 1 x CAN; 1 x USB;
- Modo de baixo consumo energético - SLEEP, STOP, STANDBY;
- Tensão de Alimentação - 2 V a 3,6 V.

Fonte: Slides 1, sistemas embebidos ISEP[44].

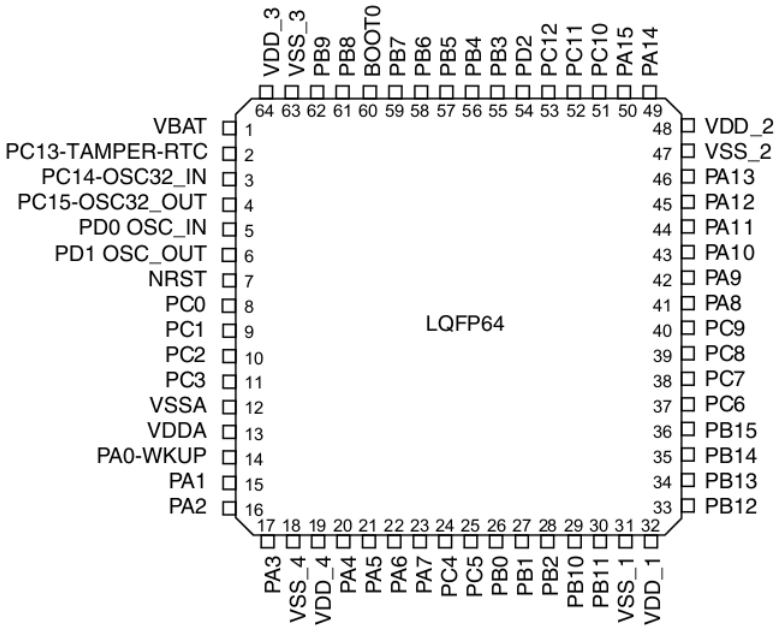


Figura 2.32: *Pinout do STM32F103RBT6 (LQFP64).*

2.9 Transmissão de dados

A USART (Universal Synchronous Asynchronous Receiver Transmitter) é um microchip que facilita a comunicação através de uma porta série de um computador usando o protocolo RS-232C[45]. Tal como a UART (Universal Asynchronous Receiver / Transmitter), a USART possibilita ao computador a interface necessária para a comunicação com modems, dispositivos com *Bluetooth* e outros dispositivos. No entanto, ao contrário de um UART, a USART oferece a opção de comunicação de forma síncrona. Na comunicação entre dois processos, o modo síncrono requer que cada um dos processos envie uma resposta de cada vez, sem ser necessária criar uma nova comunicação. O modo assíncrono significa que um processo funciona de forma

independente de outros processos. Por exemplo, um telefonema é visto como uma forma de comunicação sincrona, um dos lados responde enquanto o outro aguarda e quando este acaba de receber a informação passa a ser ele a enviar, sem ser preciso começar um novo telefonema. Enquanto para o modo assíncrono, um exemplo do nosso quotidiano é o envio de emails, um dos lados envia a informação e não é necessário que o destinatário responda de imediato, podendo esta informação ser guardada para mais tarde ser tratada e respondida.

As diferenças práticas entre o modo síncrono (que só é possível com uma USART) e modo assíncrono (que é possível com qualquer um os modos mencionados, UART ou USART) podem ser descritas da seguinte forma:

- Modo síncrono requer dados e uma frequência de relógio. Modo assíncrono exige apenas dados;
- No modo síncrono, os dados são transmitidos a uma taxa fixa. No modo assíncrono, os dados não têm de ser transmitidos a uma taxa fixa;
- Os dados síncronos são normalmente transmitidos sob a forma de blocos, enquanto os dados assíncronos são normalmente transmitidos em um byte de cada vez;
- O modo síncrono permite uma taxa de transferência de dados superior de que o modo assíncrono onde, todos os outros fatores são mantidos constantes.

2.9.1 Periféricos concebidos para a transmissão de dados

No mercado existem diversos periféricos com tecnologias que permitem estabelecer comunicações entre dispositivos com uma qualidade bastante aceitável. Nesta subsecção, foram alvo de estudo três tecnologias que poderão vir a ser implementadas no projeto, podendo assim optar pela que se torne mais vantajosa.

Infravermelhos

Os raios infravermelhos podem ser utilizados para transmitir sinais digitais entre computadores. Para tal, torna-se necessário que estes se encontrem relativamente próximos uns dos outros, o que leva a que, apenas possam ser utilizados a umas distâncias relativamente curtas e para além deste contra tempo, também é necessário que não existam obstruções físicas no espaço onde os sinais devem circular. As comunicações com infravermelhos podem atingir velocidades da ordem dos 10 Mbps, contudo são mais dispendiosas e mais susceptíveis as erros do que as próximas tecnologias que irão ser apresentadas (Bluetooth 2.9.1, Wi-fi 2.9.1).

Bluetooth

Bluetooth é uma tecnologia desenvolvida para estabelecer uma comunicação com um raio reduzido. Esta tecnologia visa substituir os fios necessários para comunicação entre dispositivos eletrónicos, manter níveis elevados de segurança, ter baixo consumo de energia e baixo custo. Além disso, possui comunicação serviços síncrona, como transmissão de voz, como assíncrona, como transferências de arquivos. A mais básica forma da ligação do Bluetooth consiste na forma de *master-slave*. O dispositivo Bluetooth (*master*) está conectado e comunica com um número de dispositivos na rede (*slaves*). A qualquer momento, o dispositivo *master* consegue transmitir informação para outro dispositivo. Esta tecnologia opera na faixa Industrial, Scientific, Medical (ISM) a volta dos 2,45 GHz, que pode variar dependendo do local em que nos encontramos no mundo, sendo esta variação cerca de 0,05 GHz. Para existir comunicação entre dois dispositivos Bluetooth é necessário que ambos se encontrem dentro de limite de proximidade. O alcance do Bluetooth foi dividido em três classes:

- Potência máxima de 100 mW, alcance de até 100 metros;
- Potência máxima de 2,5 mW, alcance de até 10 metros;
- Potência máxima de 1 mW, alcance de até 1 metro.

A velocidade de transmissão de dados com o Bluetooth é relativamente baixa, na versão 2.0, a taxa pode alcançar, no máximo 3 Mb/s (megabit por segundo). Embora estas taxas sejam baixas, são suficientes para uma conexão satisfatória entre a maioria dos dispositivos. Contudo, está é uma tecnologia em constante evolução, como prova, a versão 3.0 já é capaz de atingir taxas de 24 Mb/s.

Wi-fi

As redes sem fio IEEE 802.11[46], também conhecidas por *wireless*, foram uma das grandes novidades tecnológicas dos últimos anos. Atualmente, esta é a tecnologia mais usada para redes locais sem fios. Como prova deste sucesso, pode-se citar o rápido crescimento do número de *Hot Spots* e o facto dos computadores portáteis já saírem de fabrica equipados com dispositivos IEEE 802.11 integrados.

Neste momento existem 3 versões do IEEE 802.11, a versão IEEE 802.11b a mais utilizada, possui um velocidade de transmissão de dados de cerca de 11 Mps, comunica com uma frequência de 2,4 GHz e possui um alcance entre os 30-45 metros. A IEEE 802.11a, é uma nova versão, que possui uma velocidade de transmissão superior a IEEE 802.11b, com 54 Mps, é relativamente mais cara, comunica na banda

dos 5 GHz e tem um alcance que vai dos 7 metros até aos 22 metros, aproximadamente. A versão IEEE 802.11g é a versão que irá substituir a IEEE 802.11b, tem um velocidade de transmissão de 54 Mbps, comunica na frequência dos 2,4 GHz e tem um raio de alcance que varia entre os 30-45 metros, a grande vantagem para a IEEE 802.11a é o seu custo, esta versão é bastante mais barata o que a tornar mais apetecível ao utilizador e também o alcance de comunicação que permite cobrir uma área mais vasta.

2.10 Produtos no mercado para realizar EMG

Existem vários tipos de produtos no mercado, para a realização a eletromigrafia, com diversos tamanhos e características. Nesta secção serão apresentados três tipos de sistemas que já existem no mercado para a realização da eletromiografia.

2.10.1 MyoTrac 3G

O Myotrac 3G é uma unidade de eletromiografia de superfície, que possui 2 canais para realizar a leitura da informação proveniente dos sensores. Esta leitura é altamente precisa o que permite aos especialistas discriminar grupos principais de músculos pélvicos e avaliar com precisão esses grupos, a sua função e o uso incorreto desses músculos. Este produto faz uso de um liquid-crystal display (LCD), incorporado, para visualizar os registos obtidos pela leitura dos impulsos e permite guardar até 8 sessões da avaliação do paciente. É extremamente portátil com uma autonomia de até 14 horas. Os dados das sessões, análise estatística e certas observações sobre os pacientes podem ser guardadas virtualmente ou, então, também existe a possibilidade de serem exportadas para um formato físico, através da impressão. Na Figura 2.33 é visível o produto descrito.

Fonte: Home health provider[47].

Figura 2.33: *MyoTrac 3G*

2.10.2 Aparelho Electroterapia/Biofeedback e EMG Mod. 2U2P

O aparelho eletroterapia/*biofeedback* dispõe de electroterapia de baixa energia de impulso, sessões de *biofeedback* com medições eletromiográficas de superfície (EMGS) e testes eletroneurográficos (ENG) sobre ramificações dos nervos sensório-motores onde existem evidências de patologias compressivas. É constituído por 2 canais de saída universais para electroterapia, 2 canais de saída para terapia perineal[16] e percutânea[17] e 2 canais de entrada para EMG/ENG e *biofeedback*. Possui 33 programas terapêuticos básicos (que podem ser mudados ou servir para criação de novos) organizados por 6 grupos. Possui ainda uma interface de controlo intuitivo e simples, baseado em menus interactivos activados através de um ecrã táctil, podendo os resultados serem apresentados no ecrã ou impressora através da impressora sem fios[48]. Na Figura 2.34 está apresentado o aparelho electroterapia/*biofeedback*.

Fonte: Medigalia[49].

Figura 2.34: *Aparelho Electroterapia/Biofeedback e EMG Mod. 2U2P.*

2.10.3 Biopac MP35

O sistema Biopac é um conjunto integrado de software e hardware para aquisição e análise de dados, em atividades relacionadas com as ciências da vida, desenvolvido em vista a ser aplicado em estabelecimentos de ensino e no campo da investigação. Faz uso da captação de sinais fisiológicos como EMG, eletroencefalográfico, eletrocardiográfico, etc. Na sua base, está uma caixa que possui até 4 canais de comunicação com o exterior para a aquisição de dados. Para visualizar os dados recebidos dos sensores, existem um *software* desenvolvido pela mesma empresa que criou o produto, que permite visualizar a informação recebida, dando a oportunidade de não só visualizar o sinal, mas também analisá-lo, eliminar o ruído proveniente de outras fontes elétricas (através de um processo de calibração, que é sempre efetuado a

[16]Referente ao períneo.
[17]Incisão, atravessa a barreira cutânea.

quando de uma nova utilização), amplificar o sinal e guardá-lo[50]. Na Figura 2.35 é visível a caixa mencionada anteriormente.

Fonte: Elaborada pelo autor.

Figura 2.35: *Biopac MP35.*

3

Trabalho com o Biopac MP35

O facto de ter sido dada a oportunidade de trabalhar com o sistema Biopac (ver secção 2.10.3), foi bastante enriquecedor em vários níveis, desde ter ganho a sensibilidade, na prática, de como são conduzidos os testes para a realização da eletromiografia, também como, após o final da experiência com o Biopac, ter uns resultados que serviram como referência para os valores que virão a ser obtidos pelo projeto desenvolvido. Nesta secção serão descritos os dois testes que foram realizados com o *kit Biomedical Engineering*, um primeiro teste onde foi registada a atividade muscular produzida apenas pelo encerrar do punho, no segundo teste o objetivo foi idêntico, mas desta vez a ação pedida envolve um dinamómetro para o individuo apertar com uma mão, com o intuito de haver um aumento do registo da atividade muscular, uma vez que existe um objeto para aplicar a força exercida pelo encerrar do punho. Ambos estes testes, foram conduzidos através de um guião disponibilizado pela mesma empresa que fabricou este produto.

3.1 Experiência 1

No primeiro teste, o objetivo foi de registar a atividade muscular pela simples ação de fechar o punho. Para isso foram usados os seguintes equipamentos: Biopac MP35, os cabos de ligação aos elétrodos (SS2L), três elétrodos de gel (ver secção 2.4.2), transformador do Biopack MP35, um cabo USB e um computador a correr o sistema operativo Windows XP. O primeiro passo foi de ligar o Biopac MP35,

conectar o cabo USB entre o Biopac MP35 e o computador seguido da colocação dos elétrodos no ante-braço (os elétrodos devem ser colocados no braço dominante do utilizador, para que a amplitude da atividade muscular seja maior) com a disposição apresentada na Figura 3.1.

Fonte: Elaborada pelo autor.

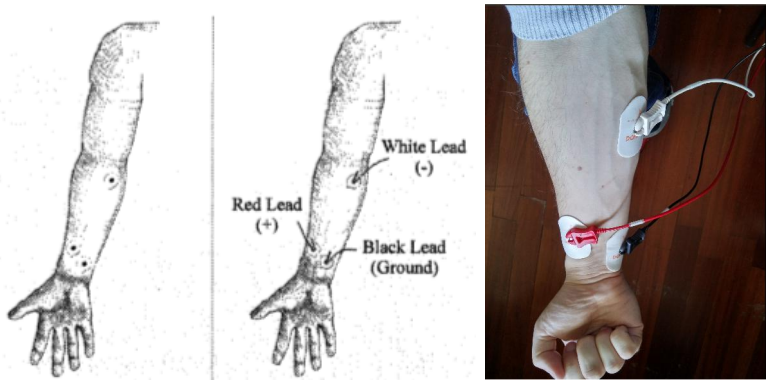


Figura 3.1: *Colocação correta dos elétrodos.*

De lembrar que a colocação dos elétrodos, não é arbitraria mas segue as regras definidas pelo SENIAM (ver secção 2.4.4). O cabo que conecta os elétrodos ao Biopac MP35 é constituído por 3 fios de cores diferentes (branco, vermelho e preto) para garantir que cada fio está ligado ao eletrodo correspondente, conforme o indicado na Figura 3.1. Após a conexão do cabo de forma correta aos 3 elétrodos, este foi ligado ao Biopac MP35 pela porta série numero 3. Com todo o *hardware* envolvido neste experiência ligado e com todas as ligações efetuadas, passou-se para a etapa de registar os valores obtidos pela leitura dos elétrodos. A transmissão de dados é feita através de uma ligação USB que liga o Biopac MP35 ao computador e os dados são visualizados em tempo real no *software*. Para visualizar os dados recebidos no computador, foi usado o *software* Student Lab System, onde foi selecionado o projeto "L01-EMG-1", projeto que já vem com o programa e foi desenvolvido diretamente para o primeiro teste da eletromiografia. Com o programa aberto e o Biopac MP35 reconhecido, foi então feita a calibração do material, para estabelecer os parametros internos do hardware em uso (como o ganho, *offset* e a escala de medida) valores necessários para que as medições que virão a ser efetuadas sejam as mais corretas possíveis. Para calibrar o sistema é pedido que após 2 segundos do inicio do registo, o utilizador feche o punho com toda a força que conseguir, para que haja um aumento da atividade registada. Este processo demora 10 segundos e ao fim desse tempo é

mostrado um gráfico com os valores obtidos. Para a calibração ser considerada bem feita, é preciso que durante os 2 segundos iniciais, o valor do registo seja igual a 0, uma vez que o utilizador não exerceu força, após este 2 segundos, altura em que é pedida que haja então o encerrar do punho, deve-se verificar no gráfico um aumento da atividade registada. Na Figura 3.2, é apresentado o gráfico obtido pelo processo de calibração.

Fonte: print screen do software biopac, elaborado pelo autor.

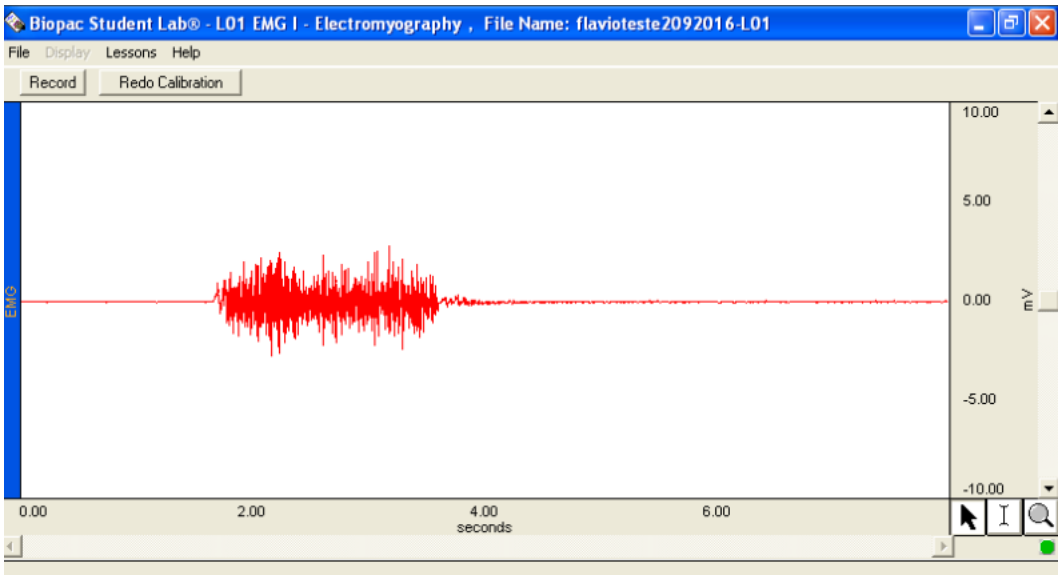


Figura 3.2: *Calibração do sistema.*

Como é possível ver na Figura 3.2, o valor inicial é de 0 mV e após o fechar do punho surgiu um aumento na atividade registada, tal como era previsto para uma calibração correta. Com a calibração efetuada, passou-se então para a etapa seguinte. Nesta etapa o procedimento pedido foi de fechar o punho de 2 em 2 segundos e em cada um destes compassos a força exercida era aumentada em partes iguais, até que na 4º vez e ultima a força fosse máxima. A Figura 3.3 está divida em duas partes em que se pode ver dois tipos diferentes de leitura, no gráfico de cima (cor vermelha) corresponde ao sinal que o sensor envia sem nenhum tratamento, em que no eixo do x está representado em tempo em segundos e a grandeza indicada pelo eixo do y é a tensão em mV, enquanto o gráfico de baixo (cor verde), diz respeito ao mesmo sinal mas já filtrado e tratado pelo *software*, em que o eixo do x representa a unidade temporal em segundos e no eixo do y os valores da tensão em mV.

Fonte: print screen do software biopac, elaborado pelo autor.

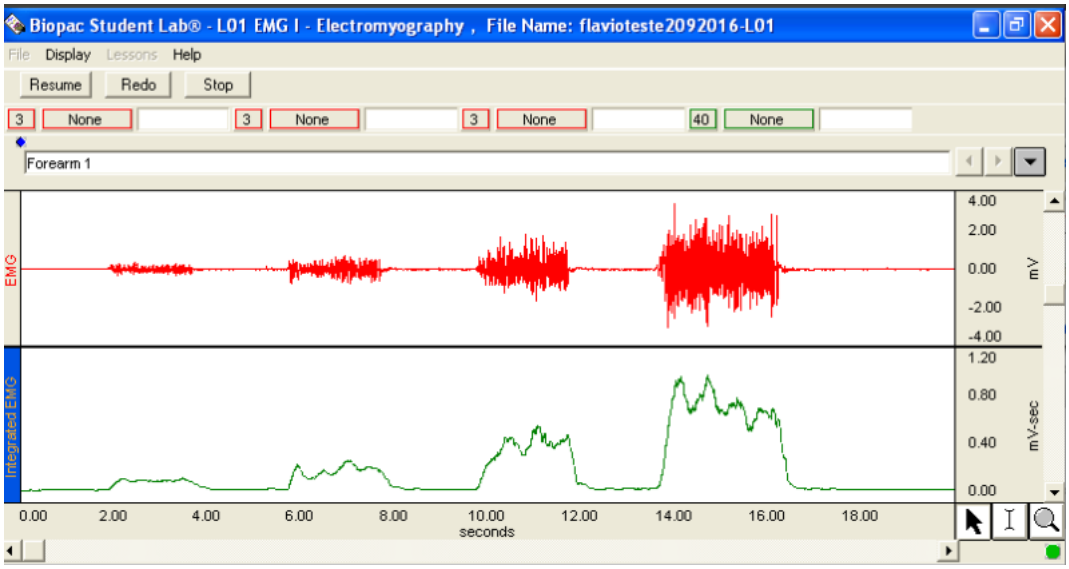


Figura 3.3: *Resultados do 1º teste.*

3.2 Experiência 2

Relativamente ao segundo teste, o objetivo é o mesmo que no primeiro teste, ou seja registar a atividade muscular, mas como será usado um dinamómetro também será registada a força exercida sobre o mesmo. A grande diferença entre este teste comparativamente ao teste anterior, é a utilização do dinamómetro. O uso do dinamómetro nesta segunda experiência permitiu que os valores obtidos tivessem uma amplitude maior, uma vez que o utilizador não exerceu apenas a força de encerrar o pulso contra a mão, mas num objeto. Uma vez que as ligações necessárias para a realização da experiência já estavam feitas bastou apenas seguir as instruções de como obter os dados. O procedimento foi idêntico ao do primeiro teste, o dinamómetro foi apertado 4 vezes, com intervalos de 2 segundos entre elas e a força aplicada foi aumentando até que na 4ª vez a força aplicada fosse máxima.

A Figura 3.4 está dividida em três partes em que estão apresentados três tipos diferentes de gráficos, todos os gráficos têm no eixo do x representada a unidade temporal em segundos, o primeiro gráfico (em cima, com a cor azul) é a força exercida pelo utilizador correspondente a cada uma das compressões aplicadas no dinamómetro em que no eixo do y está apresentada a força em Kg, a segunda leitura (gráfico do meio, com a cor vermelha) é o que o sensor envia, sem ser filtrado ou

tratado pelo *software*, no eixo do y está o valor da tensão em mV, por último, o terceiro registo (gráfico de baixo, com a cor verde) é o mesmo sinal lido pelos elétrodos, tal como na segunda leitura, mas já filtrado e tratado pelo *software* Student Lab System e a grandeza que o eixo do y indica é o valor da tensão em mV.

Fonte: print screen do software biopac, elaborado pelo autor.

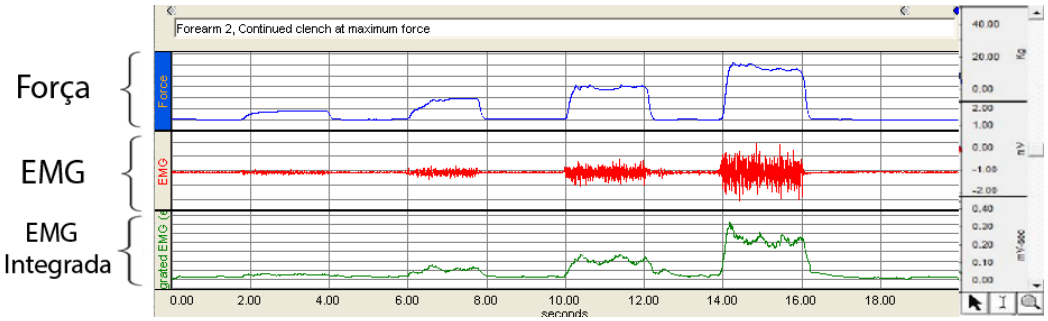


Figura 3.4: *Resultados do 2º teste.*

Depois dos testes realizados com o sistema biopac e com uma ideia mais prática do que será expectável do que acontece quando existe uma leitura dos impulsos elétricos, foi possível ganhar uma sensibilidade para o tipo de sinal que deverá ser obtido com a implementação do circuito de aquisição.

4

Arquitetura geral do sistema

De acordo com os objetivos apresentados e com o estudo efetuado no estado da arte, foi possível especificar as características do sistema que se pretende desenvolver. O projeto a desenvolver deve ser capaz de ler o sinal EMG e enviar esse sinal para uma interface gráfica, de forma a que o médico consiga visualizar o sinal obtido da experiência realizada. O sistema deve operar em dois modos distintos, um deles em tempo real, em que o sinal lido é de seguida amostrado na interface gráfica na forma de um gráfico. O segundo modo de operação tem mais em vista o objetivo proposto inicialmente, de recolher todos os dados durante um certo intervalo de tempo (definido pelo momento em que a tampa da caixa foi aberta até que encontrou um batente indicando que a operação chegou ao fim), onde o sinal obtido durante a experiência deve ser armazenado na memória do microcontrolador e só depois de terminada a fase de leitura do sinal EMG, é que este deve ser apresentado na interface gráfica, novamente em forma de gráfico.

Os médicos fisioterapeutas definiram um tempo de amostragem mínimo, que é caraterístico para o teste da eletromiografia, de cerca de 1 ms, ou seja, o sistema deve obter 1000 amostras do sinal durante um segundo. A função da interface gráfica não deve passar apenas por apresentar o sinal EMG, deve também possuir a capacidade de informar ao microcontrolador qual o modo que o sistema deve operar, tempo real ou guardar os dados, deve ser capaz de enviar a ordem de início de leitura e termino da mesma e por último, guardar um registo da experiência, quer em alguma forma de texto com a informação da tensão do sinal associada a determinado tempo, ou

então guardar o gráfico do sinal EMG em forma de imagem.

Para além do projeto conseguir ler o sinal EMG corretamente e apresentá-lo graficamente, o projeto deve possuir uma solução que permita ao produto evoluir, que seja possível adicionar mais componentes ao projeto, nomeadamente alguns sensores e manter a mesma estrutura do *hardware* original sem ser necessário reconstruir completamente o projeto para se adaptar às novas exigências. Os responsáveis do projeto podem optar assim por um sistema que cumpre com os requisitos iniciais, ou escolherem o sistema que, embora mais caro, cumpra com os requisitos iniciais e possua a capacidade de evolução.

Resumindo os requisitos para este sistema são os seguintes:

- Obtenção de um sinal EMG onde é possível extrair informações acerca dos movimentos dos músculos;
- Possibilidade do sistema operar em dois modos de aquisição de dados distintos, um modo onde será possível visualizar o sinal em tempo real e um segundo modo onde os dados do sinal são guardados e posteriormente será feita a apresentação dos dados graficamente;
- Taxa de amostragem do sinal, de pelo menos 1 ms;
- Implementação do sistema, com um baixo custo e de forma a que o produto final não possua grandes dimensões para não dificultar a qualidade dos testes que serão efetuados;
- Criação de uma interface gráfica onde será possível apresentar graficamente o sinal EMG;
- Sistema com a capacidade de evoluir;

Depois de apresentados os requisitos deste projeto e de se ter indicado as suas características, foi necessário definir qual a arquitetura do sistema que se deveria construir. O diagrama da arquitetura do sistema é apresentado na Figura 4.1.

Fonte: elaborado pelo autor.

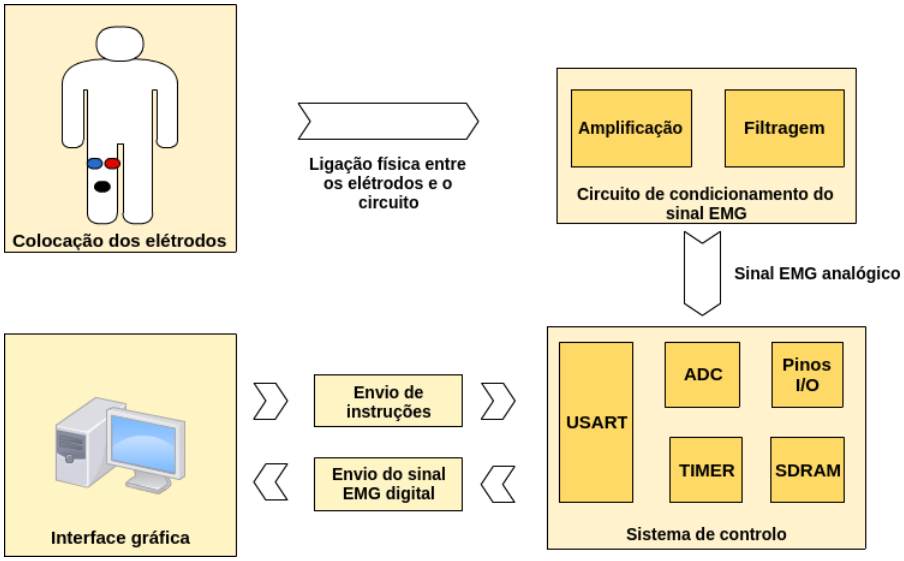


Figura 4.1: *Arquitetura do sistema.*

Tal como pode ser visto pela Figura 4.1, primeiramente, devem ser colocados elétrodos no indivíduo na zona dos músculos onde se pretende efetuar os testes, onde são conectados até a placa de condicionamento do sinal através de três fios, um para cada elétrodo. De acordo com a pesquisa bibliográfica efetuada no estado da arte, o circuito de aquisição do sinal possui duas componentes principais, uma delas é a amplificação do sinal para aumentar a amplitude do sinal lido, através de um amplificador de instrumentação que providencia um ganho, ao valor do sinal, que é regulável e introduz pouco ruído ao sinal. Como mencionado no capítulo do estado da arte, o valor máximo para este tipo de sinal é de cerca de 6 mV, por isso é importante que a amplitude do sinal seja aumentada para que os impulsos dados pelos movimentos dos músculos seja visualizado com maior facilidade. A segunda etapa é a filtragem do ruído inerente a este género de sinais, utilizando filtros de passa-alto, passa-baixo em cascata para restringir o espectro de frequências que interessam para este tipo de sinal, que de acordo com a pesquisa efetuada, este valor de frequências encontra-se entre os 20 Hz e os 250 Hz, para eliminar o ruído da rede que se situa nos 50 Hz foi implementado um filtro rejeita-banda.

Depois de o sinal ser tratado pelo circuito de aquisição, este sinal deve entrar num conversor analógico para digital para poder ser interpretado pelo microcontrolador. O sistema de controlo deve possuir pelo menos, um módulo ADC, um módulo para a transmissão de dados, dois módulos de *timer*, que seja relativamente barato e de

pequenas dimensões. O microcontrolador deve possuir um módulo ADC capaz de obter taxas de conversão inferiores a 1 ms, para conseguir trabalhar no intervalo de amostragem mínimo definido pelos fisioterapeutas, sem que a qualidade do valor obtido seja afetada. Dentro do bloco do sistema de controlo, deve ocorrer a conversão do sinal de analógico para digital e o valor obtido pela conversão deve ser enviado pelo módulo de transmissão de dados. A taxa de transmissão de dados escolhida não deve afetar a qualidade dos valores que envia nem o tempo de amostragem. O sistema de controlo deve estar configurado e ligado a um dispositivo de comunicação de forma a que consiga enviar o sinal para a interface gráfica e receber simultaneamente informações que possam surgir por parte da mesma.

Dentro da interface gráfica será possível visual o gráfico do sinal EMG de uma forma clara e com o registo do tempo associado a cada medida do sinal EMG para facilitar a posterior análise do médico. Para além de receber informações do sistema de controlo, esta aplicação gráfica deve ser capaz de enviar comandos para o microcontrolador, tais como, escolher o modo de operação, em tempo real ou guardar os dados, juntamente com a opção de início e fim da leitura do sinal e da capacidade de guardar os registos das experiências efetuadas. De lembrar que para existir uma comunicação entre o sistema de controlo e a interface gráfica, o dispositivo de comunicação deve ser do mesmo género.

Para tornar o sistema mais robusto e com a capacidade de evoluir, serão implementados dois tipos de microcontroladores, um de 8 bit que cumpre com os requisitos iniciais do projeto, ou então passar pela opção do sistema com um microcontrolador de 32 *bit* que permite que sejam adicionadas mais funções ao projeto base. A opção de terminar a leitura de dados, no modo em que o sistema deve guardar o sinal e só depois enviá-lo, é dada pela interface gráfica, pois a caixa que os engenheiros de mecânica ficaram de fazer não ficou concluída a tempo e conforme explicado no capítulo 1.1, a instrução de terminar a leitura seria dada apenas quando a tampa encontra-se um dos batentes. Esta foi a forma encontrada de simular o teste de recolha do sinal EMG no modo guardar dados, sem a utilização da caixa.

Relativamente ao tempo de amostragem, é expectável que o sistema consiga converter o sinal de analógico para digital e envia-lo dentro do intervalo de tempo de 1 ms. O tempo de conversão para os módulos ADC dos microcontroladores usados, situa-se na ordem das dezenas ou poucas centenas de μs, enquanto a velocidade de transmissão de dados está limitada pelo dispositivo de comunicação que será implementado, cerca de 115200 *bps*. O valor do sinal será enviado em formato American Standard Code for Information Interchange (ASCII), onde cada caratere corresponde a um byte. A opção de enviar o valor neste formato foi tendo em vista

que, posteriormente, a manipulação deste valor por parte da interface gráfica será mais simples. Sabendo que, no caso do microcontrolador de 32 *bit*, a resolução do módulo ADC é de 12 *bit* e no microcontrolador de 8 *bit* é de 10 *bit*, o nosso valor da conversão pode ir até 4095 ou 1023 respetivamente, o que leva a que tenham de ser enviados, no caso do valor da conversão ser máximo, 4 bytes mais um byte "$\backslash n$"que é colocado sempre no final de cada valor indicando o final do mesmo, como por exemplo "$4095 \backslash n$". Numa transmissão de dados existe um *start bit* e um *stop bit* mais a trama de 8 bit, o que corresponde a um total de 10 bit, sabendo que a nossa taxa de transmissão é de 115200 *bps*, o tempo de envio de cada caractere é, aproximadamente, 86 μs. Como queremos enviar cinco caracteres o tempo total para enviar a mensagem completa é cerca de 0,43 ms. Mesmo com o tempo de conversão máximo, aproximadamente, 260 μs e adicionando o tempo de transmissão máximo 0,43 ms, (5 caracteres), o tempo total para estas tarefas anda à volta dos 0,7 ms, ficando com alguma margem de manobra para o tempo de amostragem necessário, de 1 ms, o que torna teoricamente possível a implementação deste projeto ao nível dos tempos de amostragem.

5 Implementação

Neste capítulo serão apresentadas as várias fases de implementação do projeto. Em primeiro lugar, será explicado como foi realizada a aquisição e filtragem do sinal EMG, seguido da apresentação do hardware usado neste projeto, depois será introduzida a estrutura do código e o código implementado para cada um dos microcontroladores, posteriormente é feita uma apresentação e explicação da interface gráfica desenvolvida.

5.1 Aquisição e filtragem do sinal EMG

A fase relativa à aquisição e tratamento do sinal EMG, será descrita nesta secção. O sinal EMG é captado através de três elétrodos colocados com uma disposição e localização especifica, de acordo com a zona onde se pretende atuar (secção 2.4.4). O que obtemos com este sinal é a diferença de potencial proveniente das células musculares, dadas pelo elétrodo positivo e pelo eléctrodo negativo. Após ser obtido o sinal, é necessário passar ao tratamento do mesmo, este processo envolve um amplificador de instrumentação para aumentar o ganho do sinal original, por filtros Rejeita-Banda, Passa-Baixo e por Passa-Alto para eliminar o ruído inerente ao sinal e por último um amplificador somador não-inversor para aumentar o *offset* de forma a que o sinal lido nunca possua componente negativa uma vez que o conversor analógico para digital não lê valores negativos.

Na Figura 5.1 é apresentado um diagrama de blocos com as etapas desta fase.

Fonte: elaborado pelo autor.

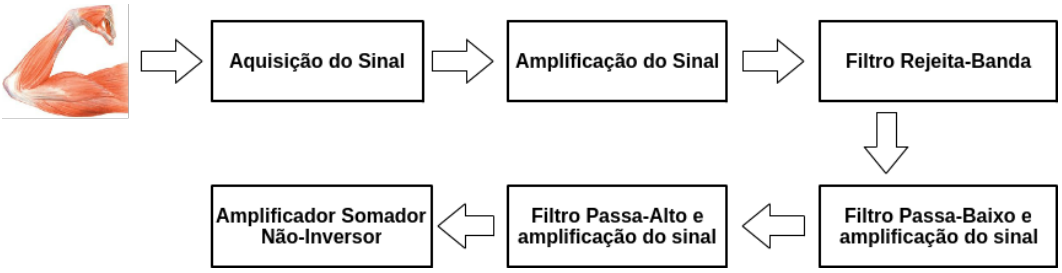


Figura 5.1: *Diagrama de Blocos da aquisição e tratamento do sinal EMG.*

5.1.1 Características do conversor AD do AVR e do ARM

Antes de ser construído o circuito para aquisição e tratamento do sinal EMG é necessário conhecer o conversor que virá a ser utilizado, visto que o circuito a desenvolver depende das características do conversor. O maior requisito para o conversor AD deste projeto, é de conseguir atingir uma taxa de amostragem superior ou igual a 1000 amostras por segundo, sendo este o valor mínimo exigido pelos médicos fisioterapeutas para o teste de eletromiografia.

O AVR escolhido, ATmega 32, já possui um conversor bastante competente embutido, com um valor máximo para a taxa de amostragem de cerca de 75000 amostras por segundo de acordo com o manual do microcontrolador. Contudo, como queremos usar uma resolução de 10 *bit*, o número de amostras por segundo baixa, pois de acordo com o fabricante para que o erro de conversão seja menor para uma resolução de 10 *bit* é necessário operar num intervalo de tempo de conversão diferente. Enquanto que para uma resolução de 8 *bit* o tempo de conversão pode variar entre $13\mu s$ e $260\mu s$ com uma resolução maior, de 10 *bit*, o fabricante não garante a mesma fiabilidade da conversão, por isso definiram uma nova janela temporal, onde garantem que a conversão com 10 *bit* é mais credível. Este novo intervalo de tempo vai de $65\mu s$ até $260\mu s$, o que leva o número máximo de amostras seja de, aproximadamente, 15000 amostras por segundo. Para além da característica do número máximo de amostras, este módulo tem disponível 8 entradas que permitem converter até 8 sinais diferentes, possui uma resolução de 10 bit[1], a tensão de referência é ajustável e varia entre 0 e 5,8 V. Com estas características, o conversor do AVR é suficiente para as exigências deste projeto.

Para o microcontrolador ARM, este possui 3 conversores AD, que cumprem

[1] A resolução de um ADC significa que o sinal lido será divido em tantos níveis conforme a resolução do conversor, no caso do AVR, o sinal lido é dividido em 1024 níveis.

com o requisito exigido neste projeto. Dentro das suas principais características é relevante mencionar: a sua taxa de amostragem situada nas 200 000 amostras por segundo, até 24 canais de conversão, uma resolução de 12 bits que permite dividir o sinal convertido em 4096 níveis e a sua tensão de referência varia entre 0 V e 3,6 V, contudo, como neste projeto foi utilizada uma placa de desenvolvimento este valor de referência é fixo nos 3 V.

Como a única diferença, entre os dois conversores apresentados anteriormente, que possui influência direta no circuito de aquisição e tratamento do sinal é o valor da tensão de referência, esse valor do AVR foi regulado para igualar o valor da tensão de referência do ARM uma vez que este é fixo. Desta forma o circuito de aquisição e tratamento do sinal desenvolvido é igual para os dois microcontroladores usados.

5.1.2 Amplificação do sinal EMG

Depois de estudados e analisados vários circuitos para a obtenção do sinal de EMG, foi possível montar um circuito capaz de ler o sinal EMG, tendo como base os esquemas que durante a pesquisa bibliográfica reuniram os melhores resultados.

Na Figura 5.2 é apresentado o sinal EMG proveniente dos elétrodos, sem qualquer tipo de amplificação nem tratamento, este tipo de sinal é conhecido como sinal *raw*.

Fonte: print screen do osciloscópio, elaborado pelo autor.

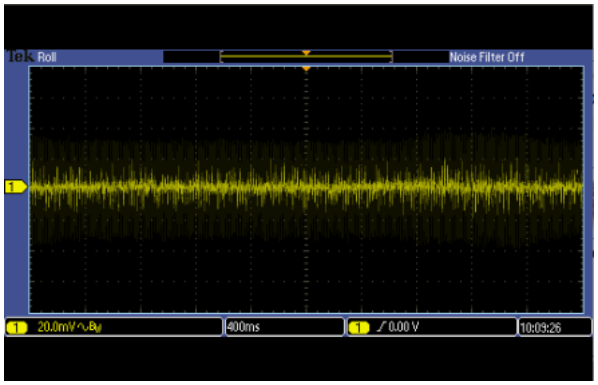


Figura 5.2: *Sinal original proveniente dos elétrodos.*

Como é visível na Figura 5.2, este sinal tem bastante ruído e uma amplitude bastante baixa, a escala do osciloscópio estava na mais baixa possível, 20 mV e mesmo assim não é visível o momento de contração do músculo que ocorreu a meio da amostragem deste sinal. Uma vez que o valor máximo de tensão que o ser humano liberta durante a contração de um músculo é de aproximadamente 6 mV,

é necessário aumentar essa amplitude para ser possível visualizar o momento de ativação do músculo. Para conseguir esta amplificação do sinal, os amplificadores de instrumentação são os mais indicados para este tipo de sinal, uma vez que, para sinais de baixa amplitude (uma característica dos sinais EMG), estes são os mais sensíveis, têm um ganho ajustável e possuem uma elevada impedância de entrada. Para conhecer mais sobre estes amplificadores, este são explicados com mais detalhe na secção 2.5.

Neste projeto o amplificador de instrumentação utilizado foi o INA128P, muito utilizado se não em todos os circuitos pesquisados, na sua maioria. Escolheu-se este amplificador, pois apresenta um baixo nível de ruído, o que é otimo já que o sinal a amplificar já possui ruído suficiente. Outro aspeto importante é o facto de necessitar de uma baixa potência de alimentação. Este amplificador é constituído por três amplificadores e por sete resistências, para controlar o ganho é preciso adicionar uma resistência externa ao amplificador (R_g), de forma a obter o ganho desejado (G). Na Figura 5.3 é exibido o esquema do INA128P:

Fonte: Texas Instruments[51].

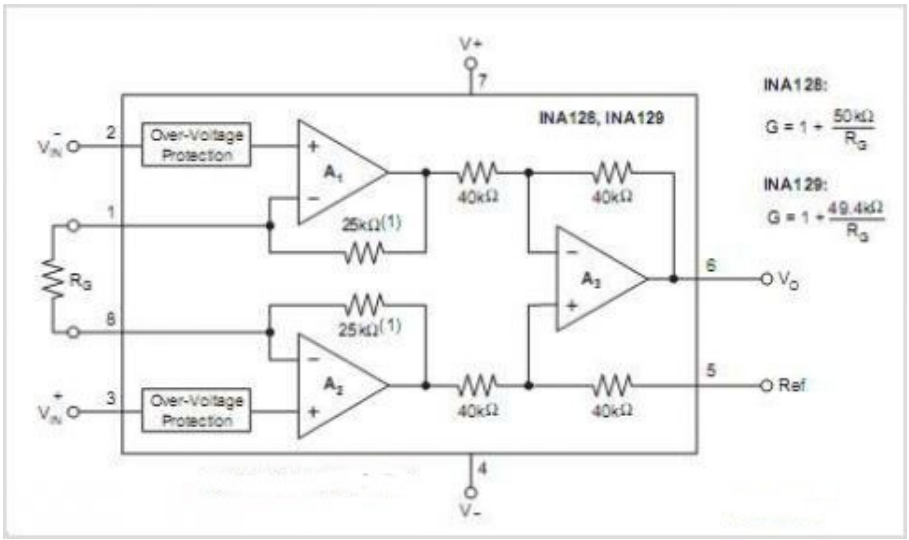


Figura 5.3: *Esquema do INA128P.*

O ganho total do circuito foi repartido em três partes, na primeira parte o ganho é dado pelo amplificador de instrumentação, na segunda parte o ganho é alcançado pelo filtro Passa-Baixo e por último, o ganho imposto no circuito é conseguido através do filtro Passa-Alto. Contudo, a amplificação maior fica com o INA128P,

pois este possui um nível de ruído inferior quando comparado com os amplificadores usados nos filtros. Este método de repartição do ganho permite que o ruído não seja completamente amplificado antes de ser removido pelos filtros, obtendo assim um sinal mais limpo e com uma maior amplitude à medida que lhe é retirado o ruído.

Antes de ser escolhida a amplificação mais adequada, é preciso ter em atenção as características do conversor de analógico para digital. Tendo em conta os aspetos mencionados na secção 5.1.1, para conhecermos a amplitude máxima que pode ser empregue neste circuito recorremos a equação 5.1, em que tensão de entrada máxima é de 3 V e a amplitude máxima de um sinal EMG é de 6 mV:

$$G = \frac{Tensao\ de\ entrada\ maxima}{Amplitude\ maxima\ do\ EMG} \tag{5.1}$$

Através da equação 5.1 foi possível encontrar qual o valor máximo da amplificação que pode ser aplicado ao circuito, cerca de 500 vezes. Contudo, como não queremos componente negativa no sinal, temos que garantir que o valor do sinal quando o músculo está inativo/repouso, se situe no valor intermédio do intervalo de tensão de referência, nos 1,5 V, (*offset*), em vez dos 0 V. Se fosse aplicado um ganho de 500, surgiriam partes do sinal que não eram lidas, pois ultrapassam o limite do valor de referência do conversor. Como demonstra a equação 5.2, o valor do sinal seria de 4,5 V:

$$Sinal = (G\ \times\ Amp\ max\ EMG) + offset \tag{5.2}$$

Para manter o valor do sinal dentro do intervalo da tensão de referência, entre 0 e 3 V, o ganho tem de ser menor, para encontrar o valor do ganho coloca-se a equação 5.2 em ordem ao ganho (equação 5.3):

$$G = \frac{(sinal - offset)}{Amp\ max\ EMG} \tag{5.3}$$

Com o ganho de 250, dado pela equação 5.3 já obtemos um valor máximo do sinal que se encontra dentro da janela de tensões que o conversor analógico para digital opera. No entanto, para evitar trabalhar na região limite [0 3] V, o valor do ganho teve de ser diminuído, ficando assim com alguma margem de segurança. A região de valores que se decidiu trabalhar foram entre 0,3 V e 2,7 V, o que corresponde a um ganho de 200, dado pela equação 5.3.

Como foi mencionado anteriormente, o ganho será repartido ficando o ganho maior, cerca de 50, a cargo do amplificador de instrumentação, o ganho escolhido para o filtro Passa-Baixo foi de 2, totalizando o ganho total do circuito para 100 e como o ganho final que queremos obter é de 200, é necessário que o ganho fornecido

pelo filtro Passa-Alto seja de 2.

Na equação 5.4 é demonstrado o cálculo do ganho para o amplificador de instrumentação INA 128P, a resistência R_g é a resistência externa e G representa o ganho:

$$G = 1 + \frac{50000}{R_g} \tag{5.4}$$

Para conseguirmos um ganho de 50, é preciso calcular qual o valor da resistência que devemos aplicar ao amplificador de instrumentação. Para encontrar o valor da resistência foi usada a equação 5.4 em função a R_g (5.5):

$$R_g = \frac{50000}{G - 1} \tag{5.5}$$

No mercado não existe uma resistência com o valor obtido na equação 5.5 ($\simeq$ 1020Ω), por isso, a solução passou por colocar três resistências (uma de 1000Ω e duas com 10Ω) em série para obter o valor desejado de 1020Ω. É de lembrar que as resistências possuem um valor de incerteza que varia entre os 1% e 5% do seu valor, o que leva a que o valor das resistências usadas não seja exatamente igual ao valor teórico.

Como é possível ver na Figura 5.3, o amplificador de instrumentação necessita de alimentação negativa (pino número 4). De referir que todo o circuito é alimentado com 5 V, uma vez que este valor de tensão é adequado para todos os componentes envolvidos. Para conseguir a tensão com valor negativo, seria preciso usar mais uma fonte de alimentação, o que tornaria o produto mais volumoso, menos portátil e mais dispendioso.

Para contornar esta questão, foram analisadas outras alternativas, sendo a que se revelou mais vantajosa e mais eficaz foi a opção pelo componente ICL7660. Este componente é a solução adequada para o problema, pois é capaz de converter tensões com valores positivos em negativos, colocando à saída o simétrico do valor de tensão da entrada. O esquema do ICL7660 é apresentado na Figura 5.4.

Fonte: intersil[52].

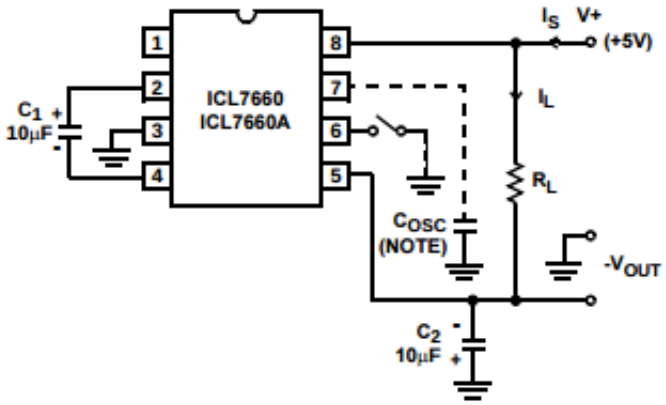


Figura 5.4: *Esquema do ICL7660.*

Com a configuração do amplificador de instrumentação explicada, na próxima subseção serão apresentados os filtros aplicados ao circuito para eliminar o ruído inerente ao sinal.

5.1.3 Filtragem do ruído do sinal EMG

Neste circuito foram utilizados três filtros em cascata para eliminar o máximo ruído possível. Para mais informações sobre o funcionamento e utilidade sobre os filtros utilizados, estes foram explicados na secção 2.6.

O primeiro filtro implementado foi o filtro Rejeita-Banda para filtrar o ruído da rede de 50 Hz, uma vez que este produto será usado num laboratório, contendo inúmeros dispositivos eletrónicos que têm influenciam direta e indireta no nosso sistema, como por exemplo, através da radiação eletromagnética que estes emitem e outras fontes de ruído. Na Figura 5.5 é representado o esquema do filtro Rejeita Banda.

Fonte: Desenvolvimento e validação de um sistema de recolha de sinal eletromiográfico baseado em placa de aquisição[53].

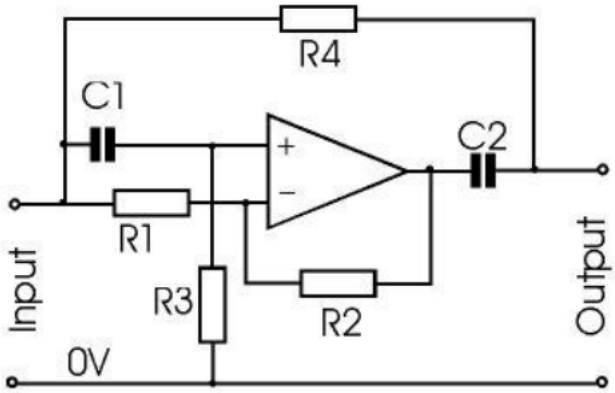


Figura 5.5: *Esquema exemplo do filtro Rejeita-Banda.*

Para um filtro Rejeita-Banda ser considerado um filtro com uma resposta adequada, este tem de possuir uma resposta com uma largura de banda o mais próxima possível da frequência a eliminar. No entanto, já foi mencionado anteriormente que os filtros não possuem respostas ideias, contudo um filtro que atenue uma pequena largura de banda que inclua a frequência de corte, já é considerado um filtro apropriado. A equação caracteristica deste filtro é apresentada de seguida (equação 5.6):

$$fc = \frac{1}{2 * \pi \times R \times C} \tag{5.6}$$

Conhecendo o valor da frequência de corte (fc) e para tornar as contas mais simples, é assumido que $R1 = R2 = 10k\Omega$, $R3 = R4$ e $C1 = C2 = 47nF$. Para ser possível implementar este filtro só falta conhecer o valor de $R3$ e de $R4$. Através da equação 5.7 esse valor é calculado, fazendo a equação 5.6 em função de R ($R3$, $R4$ são representadas por R e $C1$, $C2$ por C):

$$R = \frac{1}{2 * \pi \times fc \times C} \tag{5.7}$$

Através da equação 5.7 foi conhecido o valor da resistência, cerca de $68k\Omega$. O segundo filtro implementado foi o filtro Passa-Baixo, que possui uma maior atenuação às frequências mais altas, com uma frequência de corte configurada para os 250 Hz, eliminado assim frequências superiores a 250 Hz, que não são relevantes para a leitura do sinal EMG. Na Figura 5.6 é apresentado o esquema do filtro Passa-Baixo de segunda ordem.

Fonte: INTECH[?].

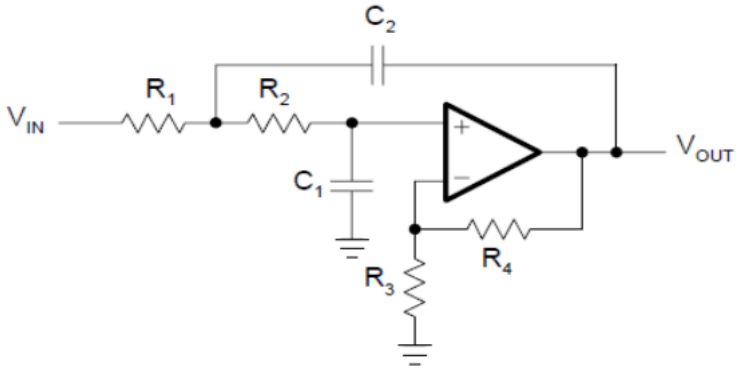


Figura 5.6: *Esquema exemplo do filtro Passa-Baixo.*

Para tornar a implementação do filtro mais prático em termos analíticos, os valores das resistências R1 e R2 foram iguais $R1 = R2$, acontecendo o mesmo para o valor dos condensadores $C1 = C2$. Sabendo qual a frequência de corte (fc), 250 Hz, e uma vez que o valor do condensador foi estabelecido à priori, cerca de $470nF$, o valor da resistência a aplicar no circuito é a nossa incógnita, para calcular o valor desta foi usada a equação 5.7, onde R representa $R1$, $R2$ e C representa $C1$, $C2$), substituindo estes valores obtemos o valor das resistências, de aproximadamente 1354Ω como não existe no mercado resistências com este valor, a solução adoptada passou por colocar cinco resistências em série para assim alcançar um valor perto do desejado. As cinco resistências usadas foram, uma de 1000Ω, de 220Ω, 100Ω, 22Ω e 10Ω conseguindo assim um valor mais perto do pretendido, cerca de 1352Ω.

Para conseguir o ganho definido para esta etapa, (de 2), as resistências $R3$ e $R4$ possuem o mesmo valor para assim obter o ganho desejado, como é dado pela equação 5.8, o valor escolhido para estas resistências foi de 1000Ω:

$$Ganho = \frac{R4}{R3} + 1 \tag{5.8}$$

O terceiro filtro aplicado foi o filtro Passa-Alto. Este filtro possui uma frequência de corte para os 20 Hz, já que abaixo desta frequência existem duas fontes importantes de ruído que condicionam a qualidade do sinal lido, tal como a ligação entre o elétrodo e a pele do utilizador e o movimento do cabo que faz a conexão entre o elétrodo e o circuito. Na Figura 5.7 é representado o esquema exemplo de um filtro Passa-Alto de segunda ordem.

Fonte: INTECH[?].

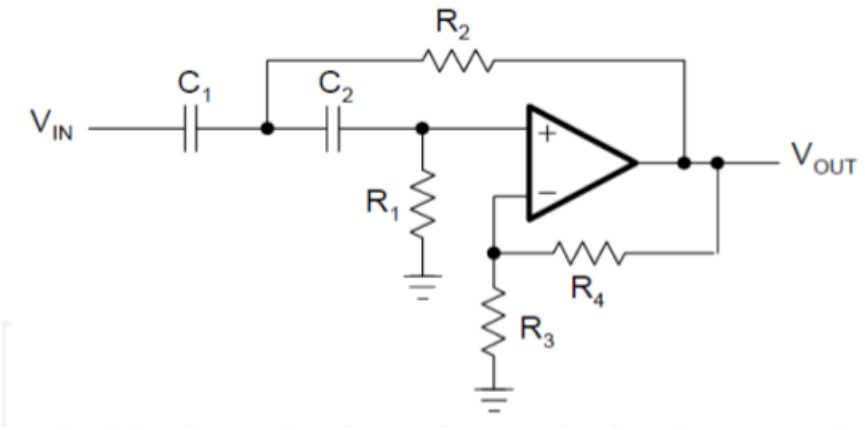


Figura 5.7: *Esquema exemplo do filtro Passa-Alto.*

Tal como aconteceu para os últimos filtros, para este também foi considerado que a resistência $R1 = R2$ e $R3 = R4$ e os condensadores possuem o mesmo valor, $C1 = C2$ pelo mesmo motivo mencionado anteriormente. Sabendo qual o valor da frequência de corte (fc), 20 Hz, e o valor para os condensadores definido nos $100nF$, falta só conhecer o valor de $R1$ e $R2$. Aplicando a equação 5.7, é conhecido o valor para $R1$ e $R2$ ($R1$, $R2$ são representados por R e $C1$, $C2$ por C), cerca de 79577Ω.

Tal como nos filtros anteriores, o valor da resistência é um valor que não existe no mercado, por isso foi necessário aplicar a mesma solução para obter um valor aproximado. Para isso foram colocadas cinco resistências em série com os seguintes valores, $47k\Omega$, $22k\Omega$, $10k\Omega$, 470Ω e de 100Ω conseguindo um valor de aproximadamente $79,57k\Omega$.

Para obter o ganho estabelecido para a última fase, (de 2), as resistências $R3$ e $R4$ possuem o mesmo valor,o valor escolhido para estas resistências foi de 1000Ω, para assim obter o ganho desejado, equação 5.8.
Para aplicar a tensão de *offset* ao circuito de forma a prevenir que o sinal lido possua valores de tensão negativos, foi aplicado um amplificador somador não-inversor. De lembrar que este género de circuito não é usado como amplificador, mas sim como um somador de tensões, ou seja, a amplitude do nosso sinal não vai aumentar, o que vai ser incrementado é o valor inicial com que o sinal começa a ser lido. Na Figura 5.8 está representado o diagrama de um amplificador somador não-inversor.

Fonte: Mastering eletronics design[54].

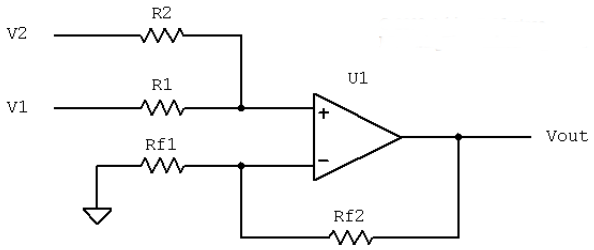


Figura 5.8: *Esquema de um amplificador somador não-inversor.*

Uma vez que o valor de tensão de referência ficou definido nos $3V$, a tensão de *offset* foi calculada para que o valor do sinal de EMG lido quando o músculo se encontra em repouso seja de $1,5V$, colocando assim o sinal a operar no meio do valor de referência. Este valor de *offset* já foi mencionado anteriormente, nas equações 5.2, 5.3 para quando se estava a definir o ganho para o sistema. Usando a nomenclatura dos componentes da Figura 5.8, o nosso $V1$ é o valor do sinal EMG, que em repouso se situa nos $4mV$ [2], (o valor ideal seria de $0mV$, mas uma vez que foram usados sucessivos aumentos de ganho para o sinal, os componentes foram adicionado erro até que este acabou por se desviar ligeiramente dos $0mV$, contudo não é problemático já que podemos corrigir com os restantes componentes deste circuito para obter à saída os $1,5V$.). Como a incógnita que queremos descobrir é o valor a aplicar em $V2$, o valor escolhido para as resistências foi idêntico, ou seja $Rf1 = Rf2 = R1 = R2$ com o valor de 1000Ω e *vout* é igual a 1,5 V. Através da equação 5.9, é possível encontrar qual o valor de V2:

$$V2 = -\frac{(Rf2 + Rf1) \times R2 \times V1 - Rf1 \times vout \times R2 - Rf1 \times vout \times R1}{(Rf2 + Rf1) \times R2} \quad (5.9)$$

Com a equação 5.9 foi encontrado o valor de V2, aproximadamente 1,496 V. Para conseguir implementar este valor no circuito foi usado um potenciómetro de $1k\Omega$ para obter o valor pretendido, regulando-o de forma a que na sua saída estivesse o valor de 1,496 V.

Com a introdução do amplificador somador não inversor, o circuito para a aquisição e filtragem do sinal EMG ficou completo. Na Figura 5.9 é apresentado o esquema do circuito desenvolvido para a obtenção do sinal EMG.

[2]Este valor foi medido com um multímetro com a escala de medida de 2 V

Fonte: elaborado pelo autor.

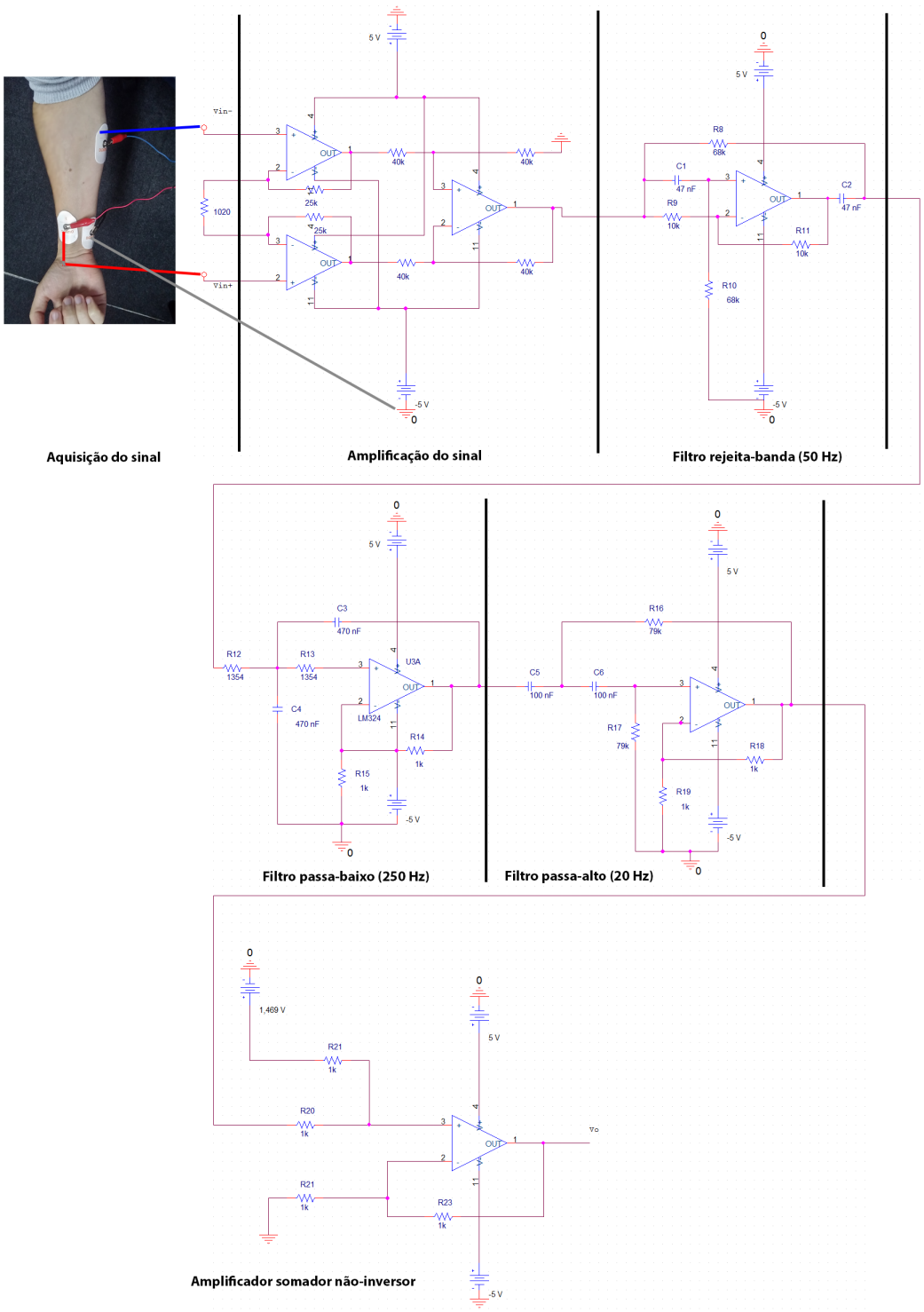


Figura 5.9: *Esquema do circuito de aquisição e filtragem do sinal EMG.*

5.2 Hardware

Nesta secção serão abordados os restantes componentes utilizados neste projeto, mais concretamente o módulo Bluetooth, a alimentação do sistema, os elétrodos e os microcontroladores.

5.2.1 Módulo Bluetooth

Para estabelecer a comunicação entre a *breadboard* e a interface gráfica, foi utilizado um módulo Bluetooth HC-06. Este módulo foi escolhido como o dispositivo para a transmissão de dados, uma vez que é relativamente barato, de relativa fácil implementação e consegue taxas de envio que não comprometem, nem a qualidade do sinal enviado nem o tempo de amostragem. Este módulo permite comunicar através de porta série, sendo necessário inicializá-lo com o mesmo *baud rate* do componente que queremos conectar, neste caso de 115200 *bps*. Foi selecionado o *baud rate* de 115200 *bps* pois este é o valor máximo que o módulo HC-06 permite. Uma vez que este módulo vem com um *baud rate* de 9600 *bps* de fábrica, foi necessário alterar esse valor. Para modificar o *baud rate* foi usado o terminal "serial port terminal", onde após estabelecida uma ligação entre o módulo Bluetooth e o terminal, foi enviado o comando $AT + BAUD8$ em que o módulo retorna uma mensagem de "ok"caso a alteração do *baud rate* tenha sido feita com sucesso. Para além da alteração do *baud rate* também é possível modificar o nome do módulo Bluetooth, escolher uma *password* e restaurar os valores de fábrica.[55]

Fonte: elaborado pelo autor.

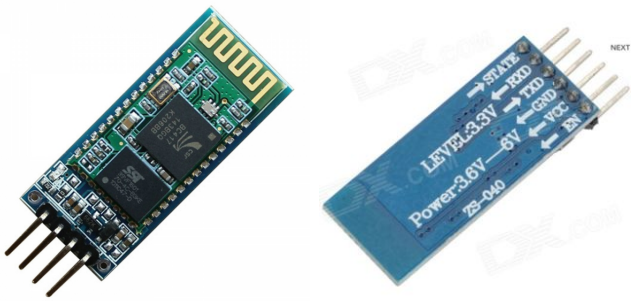


Figura 5.10: *Módulo bluetooth, HC-06.*

O módulo possui 4 pinos, ver Figura 5.10, VCC onde são ligados 5V para ser alimentado, GND em que é ligado ao GND do circuito, TXD, pino onde é transmitida a informação e RXD, pino onde é efetuada a receção de dados. Os pinos TXD e

RXD são ligados aos pinos RXD e TXD respetivamente do microcontrolador que se esteja a utilizar. Esta ligação é cruzada, para que o local emissor não seja conectado ao local emissor também, é necessário que de um local onde seja efetuada a emissão de dados e no outro seja capaz de receber essa mesma informação.

5.2.2 Alimentação do sistema

Para alimentar todos os componentes da placa de aquisição, foram usados dois tipos diferentes de alimentação. Para o circuito com a *ATmega32*, foi utilizada uma bateria de 12 V com 1800 mAh, com um adaptador de forma a que alimentação fosse distribuída através de dois cabos, um *VCC* e outro *GND*. Relativamente ao circuito com o ARM, uma vez que este está implementado numa placa de desenvolvimento a sua alimentação é apenas feita através de uma porta mini *USB*, por isso foi usado um *power bank* com 5 V e 5600 mAh, com uma saída *USB* onde foi ligada à placa através de um cabo que possui numa das extremidades uma porta *USB* e na outra uma porta mini-*USB*. A alimentação da placa seriam então feita com dois cabos, um entre o pino de 5 V da placa de desenvolvimento e a placa de aquisição de sinal e outro entre o pino GND da placa de desenvolvimento e a placa de aquisição de sinal.

Todos os componentes utilizados no circuito de aquisição de sinal funcionam com uma tensão de 5 V, o que levou a que fosse necessário implementar no circuito com a *ATmega32* o regulador de tensão L7805, para reduzir a tensão de alimentação proveniente da bateria, de 12 V, para os 5 V. No caso do circuito com o ARM, uma vez que a tensão de alimentação já se situa nos 5 V, não foi preciso utilizar mais nenhum componente adicional. Na Figura 5.11 é apresentado o L7805 e o seu esquema.

Fonte: elaborado pelo autor.

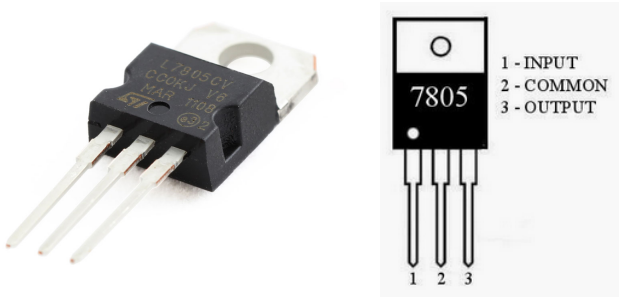


Figura 5.11: *L7805 e o seu esquema.*

5.2.3 Elétrodos de superfície

Os elétrodos usados foram os elétrodos de superfície, a principal característica deste tipo de elétrodo é que possuem um gel de forma a que o sinal lido seja menos ruidoso e são indolores para o paciente. Estes elétrodos são ligados à placa de aquisição através de três crocodilos, um para cada elétrodo, de lembrar que cada experiência exige a utilização de três elétrodos. Na Figura 5.12 estão apresentados os elétrodos e os crocodilos usados neste projeto.

Fonte: elaborado pelo autor.

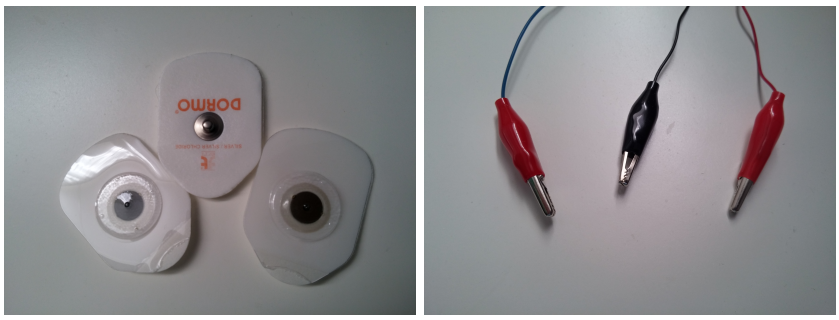


Figura 5.12: *Esquerda, elétrodos de superfície usados. Direita, crocodilos.*

5.2.4 ATmega32

O microcontrolador escolhido dentro dos AVR foi a ATmega32. Este microcontrolador cumpre os requisitos necessários para desenvolver o projeto, possui três *timers* onde serão necessários dois, um módulo de conversão de analógico para digital integrado, um módulo de comunicação síncrona para estabelecer uma comunicação com outros dispositivos, uma frequência máxima de relógio de 16 Mhz e 40 pinos. Para além das garantias que as suas características proporcionam, este é um microcontrolador barato. Para mais informações sobre este tipo de microcontrolador e a sua família consultar secção 2.8.1. Na Figura 5.13 é apresentado o circuito elétrico da ATmega, onde é possível conhecer todas as ligações. Nos pinos $XTAL1$ e $XTAL2$ foi ligado o cristal de 16 MHz, no pino $AREF$ é ligada a tensão de referencia para o módulo ADC, cerca de 3 V, os pinos $AVCC$ e GND correspondem à alimentação do módulo ADC, 5 V, os pinos VCC e GND são relativos à alimentação do microcontrolador, com 5 V, no pino $ADC0$ é ligado o sinal proveniente da placa de aquisição do sinal EMG e por último, o pino TXD é ligado ao pino RX do módulo HC-06 enquanto que o pino RXD é conectado ao pino TX do HC-06. O módulo HC-06 é alimentado com 5 V.

Fonte: elaborado pelo autor.

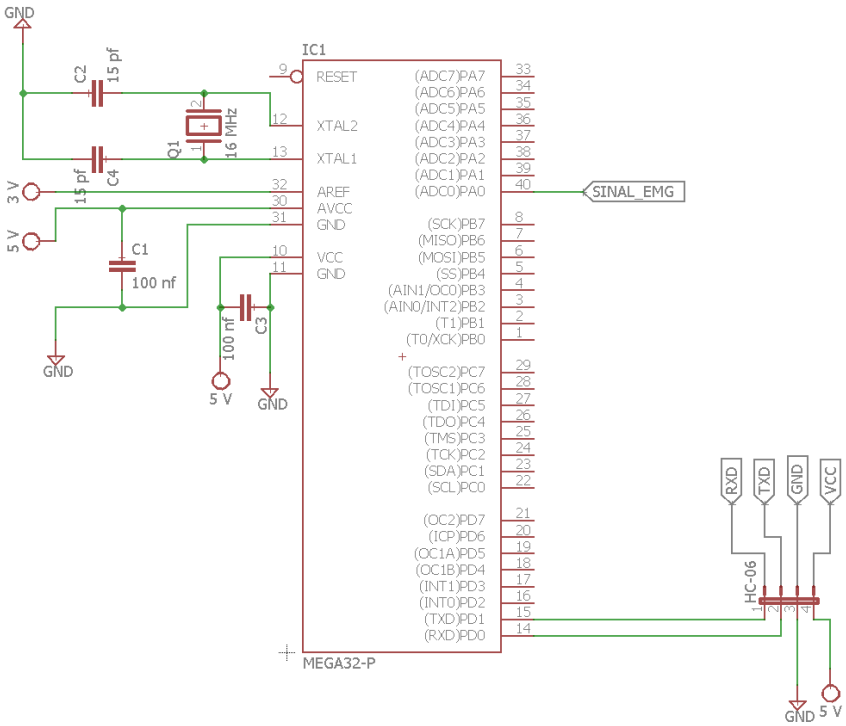


Figura 5.13: *ATmega32.*

Para programar o ATmega32 foi usada a programadora USBASP V2.0. Esta programadora possui 10 pinos, embora sejam necessários apenas 6, um para a alimentação, outro para a massa, os restantes são o MISO, MOSI, SCK e RESET. Todos estes 4 pinos têm de ser conectados aos respetivos pinos do microcontrolador em uso. Na Figura 5.14 está representada a programadora utilizada (imagem à esquerda) e o seu esquema (imagem à direita).

Fonte: elaborado pelo autor.

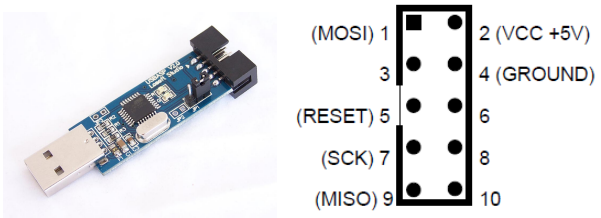


Figura 5.14: *Esquerda, USBASP V2.0. Direita, esquema USBASP V2.0*

5.2.5 STM32F429

O microcontrolador ARM usado neste projeto foi o ARM Cortex M4 de 32 *bits*. Este microcontrolador cumpre e excede todos os requisitos necessários para desenvolver este projeto, possui dezassete *timers*, três módulos ADC de doze *bits*, quatro módulos de USART, uma frequência de relógio máxima de 180 MHz e 144 pinos. Na Figura 5.15 são apresentadas todas as ligações efetuadas na placa de desenvolvimento usada.

Fonte: elaborado pelo autor.

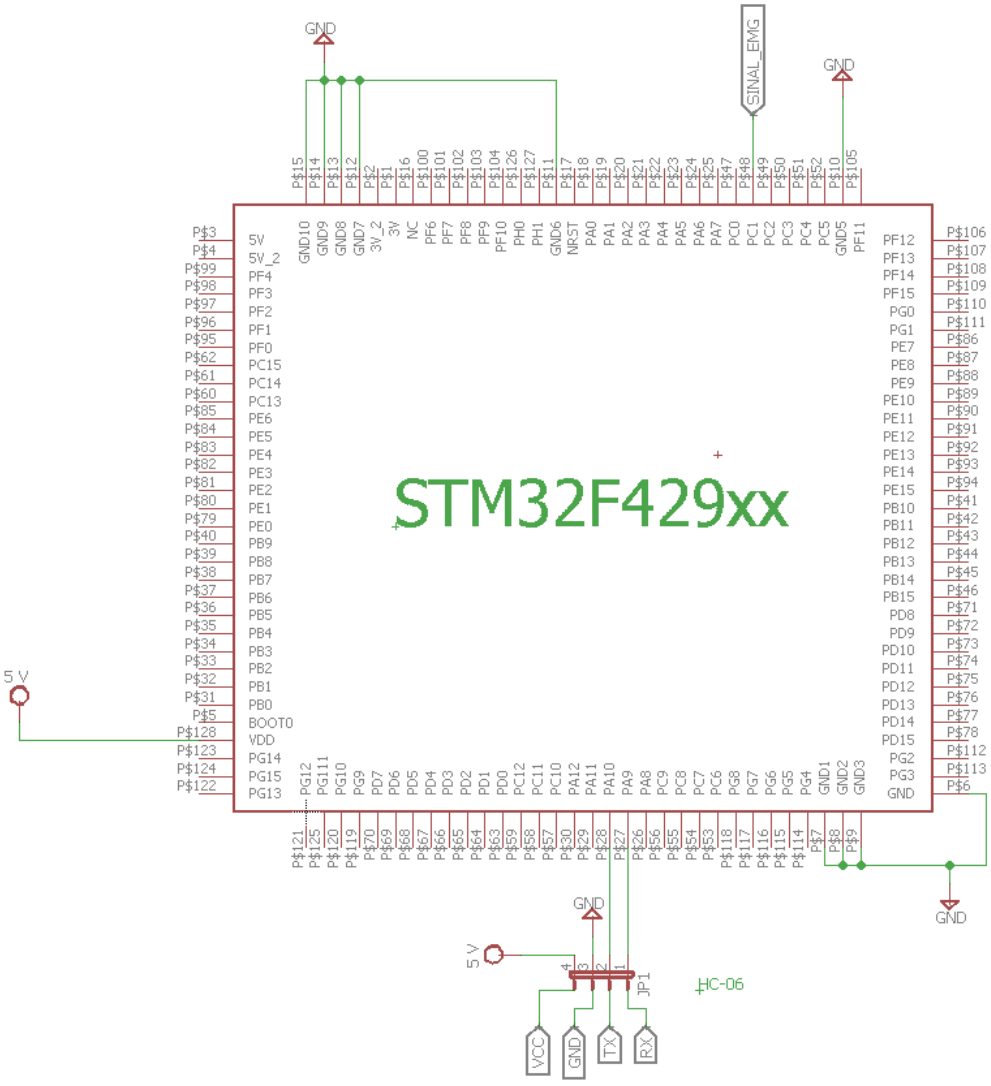


Figura 5.15: *Placa de desenvolvimento ARM.*

Este ARM está implementado numa placa de desenvolvimento, que torna o trabalho com o ARM mais eficaz, uma vez que os pinos estão disponíveis de uma forma mais simples e acedível. A aplicação deste microcontrolador é mais para uma prespetiva de futuro, caso seja necessário dotar este sistema com mais capacidades, como novos componentes, novos sensores, estabelecer comunicações com vários dispositivos, de uma forma geral, dotar este projeto com a capacidade de evolução. Relativamente ao preço é mais caro do que a ATmega32, mas a diferença de preço é aceitável pois oferece mais soluções. Para mais informações sobre os ARM, estes são explicados no capitulo 2.8.3.

Os pinos VDD e GND correspondem à alimentação da placa, com cerca de 5 V, o pino PC1, é relativo a aquisição por parte do módulo ADC do sinal EMG proveniente da placa de aquisição do sinal EMG, o pino $PA10$ é ligado ao pino TX do módulo HC-06 e o pino $PA9$ é conectado ao pino RX do HC-06. O HC-06 é alimentado com uma tensão de 5V.

Para passar o programa para o ARM é necessário conectá-lo a uma entrada USB do computador, através de um cabo mini-USB para USB (Figura 5.16).

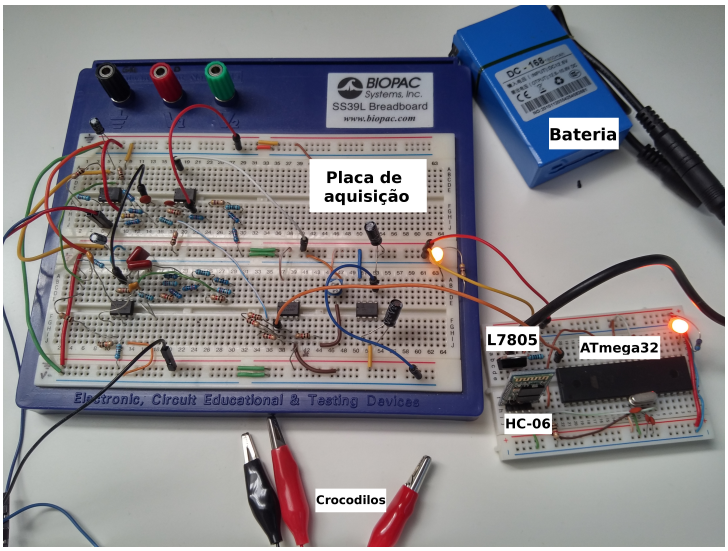
Fonte: elaborado pelo autor.

Figura 5.16: *Cabo mini-USB para USB.*

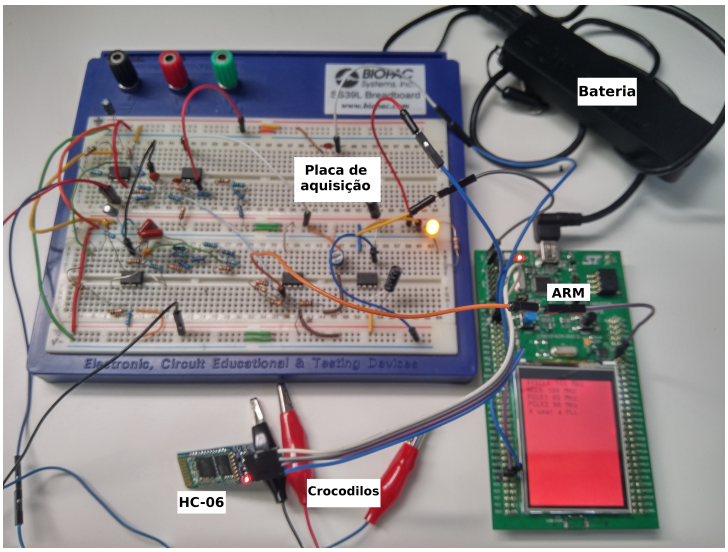
5.2.6 Montagem final

O circuito da placa de aquisição para o sistema com a ATmega32 é representado na Figura 5.17 a), a montagem para o sistema com o ARM é apresentado na Figura 5.17 b).

Fonte: elaborado pelo autor.



a)



b)

Figura 5.17: *a), Placa de aquisição para o sistema com a ATmega32. b), Placa de aquisição para o sistema com o ARM.*

5.3 Estrutura do código

A estrutura do código para cada um dos microcontroladores é idêntica, uma vez que estes passam pelas mesmas etapas para amostrar o sinal proveniente do circuito de aquisição.

Como o microcontrolador faz uso da receção de certos caracteres para por em marcha certas acções, como por exemplo, escolher o modo de operação ou definir o momento de início ou fim da leitura do sinal, foram colocadas algumas restrições no envio dos caracteres por parte da interface gráfica, para que não hajam situações em que a ordem para o fim da leitura seja dada sem que a ordem de início da leitura tenha sido recebida.

Estas restrições são mais aprofundadas na secção 5.4, contudo para o leitor conseguir entender o código implementado relativamente aos microcontroladores é feita de seguida uma breve explicação.

Inicialmente as únicas letras que se podem enviar são as referentes ao modo de operação. Após o modo de operação ficar definido, o envio dos caracteres para escolher o modo de operação ficam indisponíveis passando a ficar ativo o envio da letra referente à ordem de início da leitura. Quando o início da leitura é recebido, a opção de enviar o caractere que corresponde a esta ação fica desativo e por lógica, o próximo passo é de ativar o envio do fim da leitura. Com o fim de leitura recebido por parte do microcontrolador, o envio deste caractere fica indisponível e volta a ficar ativo o envio das letras referentes à escolha do modo de operação e o ciclo repete-se após o modo voltar a ficar definido.

A opção de colocar estas restrições no lado do emissor, passou pelo facto de ser mais simples desativar e ativar o envio de caracteres por parte da interface gráfica do que aumentar a complexidade do código dos microcontroladores.

O fluxograma relativamente ao código principal criado para os microcontroladores é apresentado na Figura 5.18, onde é demonstrado o funcionamento do programa na função sua principal. Começando por inicializar as variáveis e fazer a configuração de cada um dos módulos ADC, USART e dos *timers* de acordo com o pretendido. É possível ainda visualizar as várias condições que existem, como por exemplo a verificação do modo de operação.

Fonte: elaborado pelo autor.

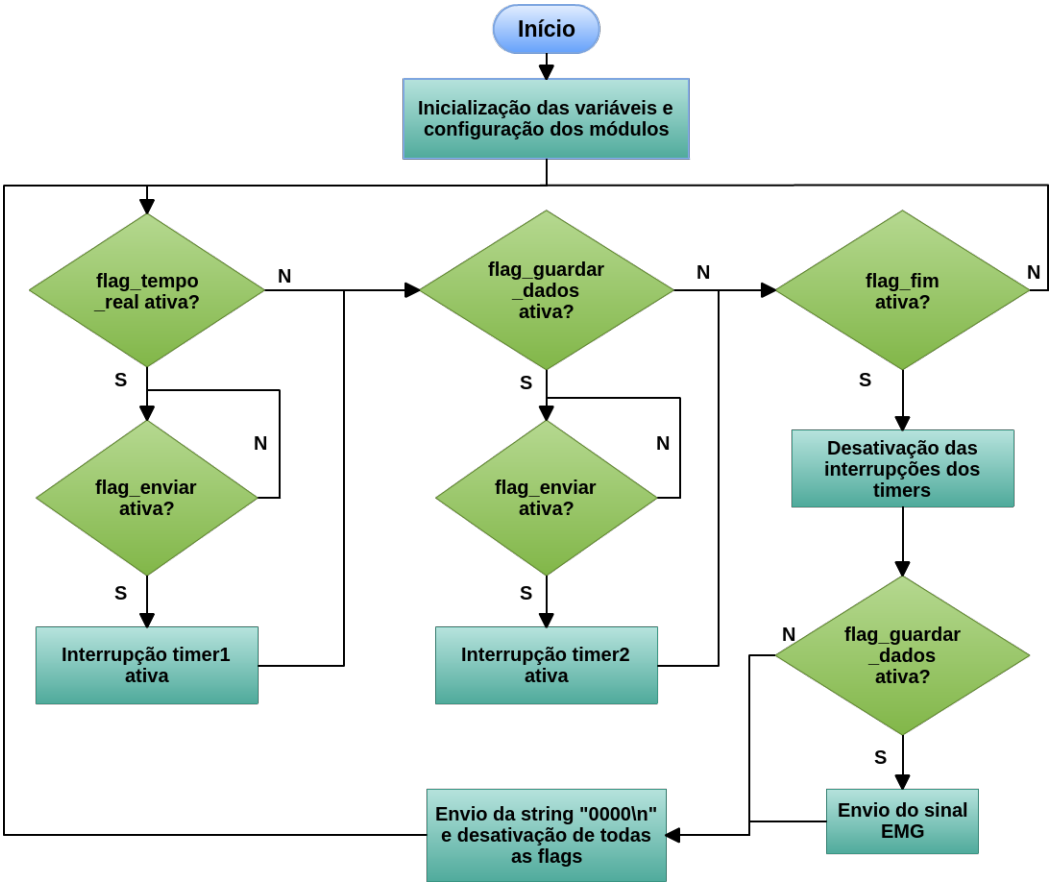


Figura 5.18: *Fluxograma dos microcontroladores.*

Na Figura 5.19 está representada a rotina de interrupção receção da USART, em que cada caractere corresponde a um evento que vai desencadear uma respetiva ação do código principal.

Fonte: elaborado pelo autor.

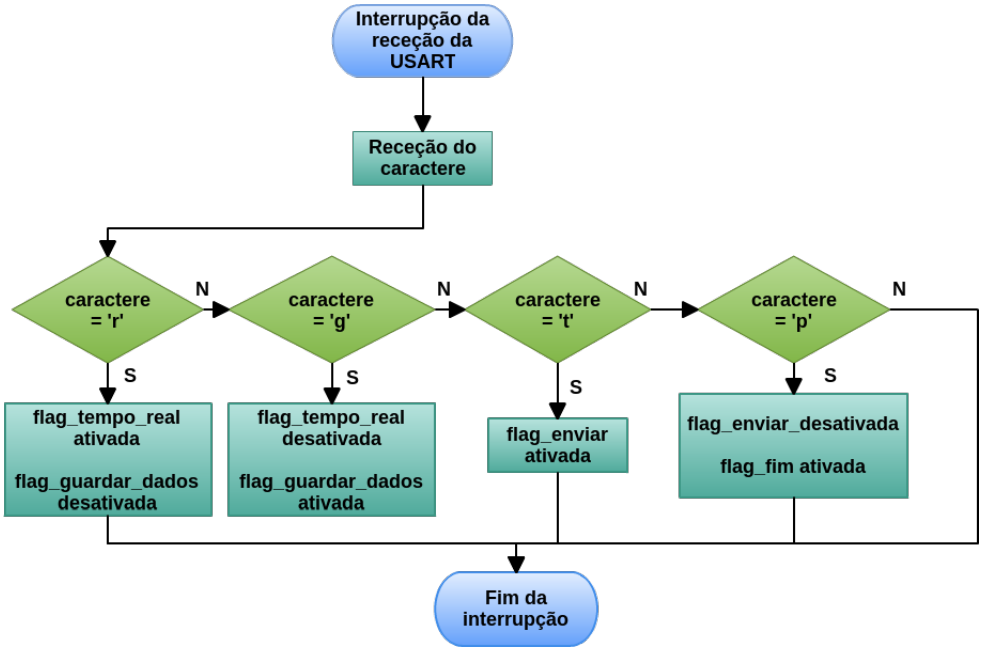


Figura 5.19: *Fluxograma da rotina de interrupção da USART.*

Na Figura 5.20 são apresentadas as rotinas de interrupção de cada um dos *timers*, que ocorrem sempre dentro de um determinado intervalo de tempo, onde é feita a conversão do sinal analógico para digital e no caso do *timer* associado ao modo "tempo real"também é realizado o envio deste valor.

Fonte: elaborado pelo autor.

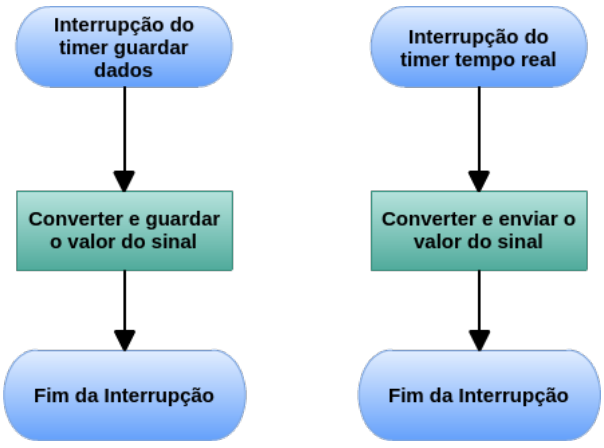


Figura 5.20: *Fluxograma da rotina de interrupção dos timers.*

Explicando a Figura 5.18, em primeiro lugar temos a inicialização das variáveis, seguida da configuração dos módulos *Timer*, *ADC* e *USART*. O módulo *Timer* serve para definir o tempo de amostragem, o módulo ADC é usado para converter o sinal analógico em digital e o módulo USART é utilizado para comunicar com outros dispositivos, enviando e recebendo informações. Depois de terem sido realizadas todas as configurações, o sistema fica à espera que receba um caractere, que lhe informe qual o modo que deve operar, a definição da escolha do modo situa-se dentro da rotina da interrupção da USART (Figura 5.19), que ocorre sempre que o sistema recebe um caractere por parte da interface gráfica. Foi definido que este programa podia ter dois modos de operação, um deles funciona da seguinte forma, à medida que obtém o valor da conversão A/D, esse valor deve ser enviado para um dispositivo alvo, este modo é denominado de modo "tempo real".

O segundo modo, recolhe todos os dados durante um certo intervalo de tempo e depois de terminada a experiência, envia todos os dados recolhidos até aquele instante, este segundo modo é intitulado de "guardar dados". Para este sistema reconhecer qual o modo em que deve operar, o sistema deve receber um caractere r indicando que o modo escolhido foi o modo "tempo real"ou a letra g que representa o modo "guardar dados". Caso não receba nenhum caractere ou o caractere recebido seja diferente destes dois (r e g) o sistema fica à espera até que receba novamente um caractere e que este seja um dos definidos para escolher um dos modos de operação. Após a receção do caractere r, o modo ficou escolhido e o sistema fica à espera que seja enviado o sinal de *trigger*[3], através da receção da letra t, para assim dar inicio ao processo de amostragem do sinal EMG. Depois de recebida a letra t o programa entra dentro da rotina de interrupção do respetivo *timer* onde irá realizar as instruções apresentadas na Figura 5.20, relativamente ao *timer* "tempo real", dentro de um intervalo de tempo conhecido como o tempo de amostragem. Este processo só termina quando o sistema receber o caractere p, que retorna à etapa em que fica à espera de um caractere para escolher o modo de operação.

Caso o modo de operação escolhido foi o modo "guardar dados"(recebeu a letra g), o sistema fica novamente à espera da receção do caractere t. Quando recebe esse caractere, o sistema dá início ao processo de "guardar dados"dentro da rotina de interrupção do respetivo *timer*, com um certo tempo de amostragem, que consiste na conversão do valor analógico do sinal EMG vindo da placa de aquisição para um valor digital, onde esse valor é guardado na memória interna do microcontrolador. Quando o sistema receber a letra p, o processo de converter e de guardar são interrompidos e

[3]A palavra que mais se assemelha em português, neste contexto, é disparo, que significa que foi enviada uma ordem para inicializar uma operação.

todos os valores guardados são enviados para o dispositivo alvo e o sistema retorna para o fase em que fica à espera da escolha do modo de operação.

5.3.1 Código - AVR

Todo o código desenvolvido para o AVR foi realizado no *eclipse 4*, que é um software de desenvolvimento, pode ser utilizado em *Windows*, *linux* e *macOS*, é livre e suporta vários tipos de linguagem através de *plugins*[4] entre elas, a linguagem C que foi a linguagem escolhida.

Antes de ser explicado o código, de referir que foi usado um relógio externo para ser possível atingir uma frequência maior, uma vez que a frequência máxima do relógio interno da ATmega32 está limitada aos 8 MHz. Com o relógio externo foi possível duplicar essa valor, operando-se com uma frequência de 16 MHz (frequência máxima para o microcontrolador escolhido, quando se usa um relógio externo). Para indicar ao microcontrolador que o relógio a utilizar seria um relógio externo, foi necessário alterar os *fuse bits*[5]. No *eclipse*, existe uma opção que permite alterar os *fuse bits* (ver Figura 5.21), bastando introduzir os valores que queremos. Neste caso, de acordo com o manual da ATmega32, para ser escolhido o modo de funcionamento que use um relógio externo os *fuse bits* deveriam de ser os seguintes, *low fuse*:0xFF, *high fuse*:0x99.

Fonte: print screen do software eclipse, elaborado pelo autor.

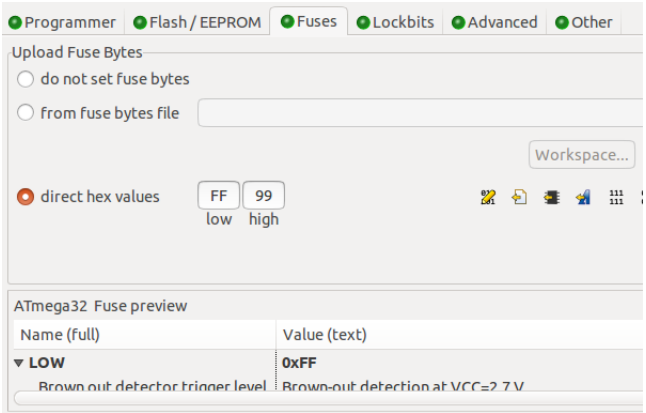


Figura 5.21: *Eclipse, janela de alteração dos fuse bits.*

[4]É um programa de computador, que permite adicionar novas funções/caractereísticas a outros programas.

[5]São *bits* que alteram o modo de funcionamento do microcontrolador, no manual de cada microcontrolador existe uma tabela com as várias alterações possíveis.

Depois do microcontrolador estar configurado, restou apenas colocar um cristal ligado aos pinos corretos da ATmega32, com dois condensadores que ligam cada pino do cristal à massa, o valor dos condensadores foi de 15 pf. Na Figura 5.22 é visível o cristal utilizado (imagem à esquerda) e o esquema de montagem do cristal com o microcontrolador (imagem à direita).

Fonte: elaborado pelo autor.

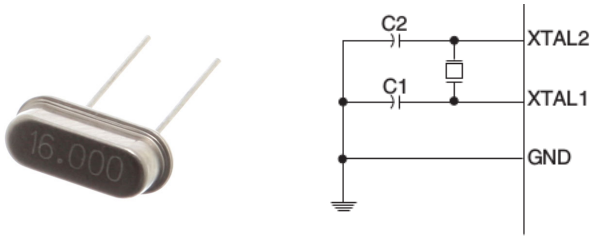


Figura 5.22: *Esquerda, relógio externo de 16 MHz. Direita, esquema de ligação, entre o AVR e o relógio externo.*

Tal como referido na subseção 5.3, o primeiro passo foi de definir as variáveis que serão usadas, todas elas são declaradas globalmente. Como a maioria destas variáveis vão ser utilizadas dentro de rotinas de interrupções, estas devem ser classificadas como "*volatile*"para poderem ser acedidas por funções que não são interrupções. Depois de inicializadas as variáveis, passou-se para a configuração dos módulos, o primeiro a ser configurado foi o *timer*, como serão utilizados dois modos de operação diferente, "tempo real"e "guardar dados", foram implementados dois *timers*. O primeiro *Timer* implementado, foi o *Timer2* de 8 bits, que será usado pelo modo "guardar dados", com um tempo de amostragem de 1 ms[6]. Através da equação 5.10, é possível definir o tempo de amostragem que queremos usar:

$$f\_tim = \frac{f\_cpu}{(N \times (OCR + 1))} \tag{5.10}$$

Como o tempo de amostragem é de 1 ms a frequência correspondente é de 1000 Hz, logo f_tim é de 1000 Hz, também é sabido que f_cpu é de 16 MHz. Para ficar apenas uma variável desconhecida, o valor estabelecido para OCR foi o seu valor máximo, 255 (*timer* de 8 bits). O valor para N, que representa o *prescaler*, é conhecido através da equação 5.11, que teve origem depois de ter sido colocada a equação 5.10 em função de N:

[6] 1 ms, é o tempo de amostragem recomendado pelos fisioterapeutas para o teste de eletromiografia

$$N = \frac{f\_cpu}{(f\_tim \times (OCR + 1))} \tag{5.11}$$

O resultado dado pela equação 5.11 foi de aproximadamente 62,5, contudo como não existe nenhum *prescaler* com o valor de 63 no manual do AVR, o valor escolhido foi de 64, que é o valor mais próximo. Devido a este constrangimento, é necessário calcular o valor exato para OCR (equação 5.12, que originou da colocação da equação 5.10 em função a OCR).

$$OCR = -\frac{f\_tim \times N - f\_cpu}{(f\_tim \times N)} \tag{5.12}$$

O novo valor para o OCR é de 249. Este *Timer* opera no modo de *clear Timer on Compare* (CTC) que faz com que a variável contador seja limpa sempre que o valor do contador seja igual ao *OCR*. Sempre que esta igualdade acontece é ativada uma interrupção do *timer2*. Com o valor do *prescaler*, do *OCR* e definido o modo que o timer deve operar, já é possível configurar o *timer2* corretamente.

Para implementar o segundo *timer* para o modo "tempo real", foi tido em consideração que o programa para que o microcontrolador estava a enviar continuamente os dados, não suportava uma taxa de amostragem de 1 ms. O tempo que demorava a receber e colocar o valor do sinal num gráfico era superior a 1 ms e o conceito de tempo real deixava de ser válido, pois o valor do sinal que estava a ser amostrado graficamente não correspondia ao valor atual da atividade muscular. Embora o programa receba todos os dados que lhe fossem enviados, estes eram guardados sequencialmente por ordem de chegada num *buffer*, de tamanho ajustável onde eram colocados no gráfico pela mesma ordem. Contudo, como o programa demorava mais tempo a desenhar o gráfico para o novo valor do que a receber novos valores, à medida que o tempo passava o *buffer* continuava a crescer e o atraso para a representação gráfica do último valor recebido ia crescendo também, existindo testes em que o músculo já tinha sido ativo e apenas passados alguns segundos é que essa indicação era representada graficamente.

Por isso, a taxa de amostragem teve de ser mais baixa do que o inicialmente previsto, ficando esta com uma taxa de 32 ms. O valor de 32 ms foi conseguido através de experiências sucessivas de envio de dados a taxas de amostragem diferentes, até que a aplicação gráfica apenas recebesse um valor no seu *buffer* e que esse valor fosse imediatamente tratado, não recebendo mais nenhum valor até terminar todo o processo de apresentação do sinal graficamente.

Esta taxa permite ao programa conseguir receber os dados por parte do microcontrolador, realizar as operações matemáticas necessárias para o tratamento dos

dados, que serão abordadas na secção 5.4, e colocar o novo valor num gráfico sem que receba mais nenhuma informação do microcontrolador. Para configurar o novo *timer*, *timer0* de 8 bits, foi aplicada a equação 5.11, onde f_cpu é a mesma, a frequência do *timer* é de 31,25 Hz e foi dado novamente o valor máximo para OCR, 255. Uma vez que o valor máximo para o *prescaler* é de 128 e o novo valor do *prescaler* foi de 2000, não é possível utilizar este *timer* com a taxa de amostragem desejada. A solução passou por utilizar o *timer* de 16 *bits*, *timer1*. Com 16 *bits*, o valor máximo para OCR é de 65535, logo através da equação 5.11, é sabido o valor do *prescaler*, cerca de 7,81.

Com o novo valor do *prescaler* já é possível encontrar no manual do AVR um valor que mais se aproxima de 7,81, ficando com *prescaler* de 8 para este *timer*. Calculando agora o valor real de OCR, com o *prescaler* a 8 através da equação 5.12, obtem-se um OCR de 63999.

Tal como o *timer2*, o *timer1* funciona no modo CTC. As interrupções de ambos os *timers*, são inicialmente desativadas, sendo ativadas apenas quando é recebido o sinal de *trigger*.

Para realizar a conversão do sinal analógico para digital, foi configurado o módulo ADC do AVR. Este módulo foi configurado para usar uma tensão de referência externa, que serão os 3 V, opera com uma resolução de 10 bit e realizara leitura e respetiva conversão do sinal no pino ADC0 do AVR. Estando o módulo praticamente configurado, falta apenas escolher qual a frequência a que o ADC deve funcionar. De acordo com o manual da ATmega32, o intervalo de tempo em que o ADC opera corretamente com uma resolução de 10 bit, situa-se entre os $65\mu s$ e os $260\mu s$. Como queremos que a conversão seja o mais rápido possível, uma vez que estamos a trabalhar na ordem dos ms, foi tentado encontrar uma frequência para o ADC que mais se aproximasse dos $65\mu s$. Para encontrar a frequência que o ADC deve funcionar, é necessário recorrer a equação 5.13:

$$f\_adc = \frac{f\_cpu}{Prescaler} \tag{5.13}$$

Onde f_adc corresponde à frequência do ADC, f_cpu é a frequência do relógio externo, 16 MHz e o *prescaler* varia entre os valores de 2, 4, 8, 16, 32, 64 e 128. Como queremos que o valor da taxa de conversão se enquadre dentro do intervalo de tempo definido pelo fabricante, o único *prescaler* que se pode utilizar é o 128, todos os restantes valores do *prescaler* colocavam a f_adc fora da janela temporal para a conversão, situada entre os $65\mu s$ e os $260\mu s$. Através da equação 5.14 obtemos o valor de f_adc que se situa nos 125 kHz.

$$Prescaler = \frac{f\_cpu}{f\_adc} \tag{5.14}$$

Com o valor de f_adc de 125 kHz, que corresponde a $8\mu s$ e sabendo que para haver uma conversão completa são necessários 13 ciclos, o que leva a que o tempo total da conversão analógica para digital seja de $104\mu s$. Com o valor da frequência do ADC conhecido, o módulo ficou totalmente configurado de acordo com as nossas exigências.

Por último, falta explicar a configuração da USART, o módulo que permite ao microcontrolador comunicar com outros dispositivos. A USART ficou configurada da seguinte forma: modo síncrono, sem paridade, 1 *stop bit* e uma trama com 8 *bits*. Para escolher o *baud rate* deste microcontrolador, teve-se em conta que o *baud rate* máximo do componente usado para comunicar, o módulo *bluetooth* HC-06, é de 115200 *bps*. Dado que para existir uma comunicação, quer o AVR e HC-06 devem possuir o mesmo *baud rate*. Para configurar a USART com um *baud rate* de 115200 *bps*, é preciso recorrer a equação 5.15

$$UBRR = \frac{f\_cpu}{(baudrate - 1) * 16} \tag{5.15}$$

Em que $UBRR$ é o valor que informa ao AVR qual o *baud rate* que deve usar, f_cpu é a frequência do relógio externo, 16 MHz e *baud rate* é o valor de *baud rate* que queremos trabalhar, 115200 *bps*. Conhecendo todos as variáveis a exceção do $UBRR$, obtemos o seu valor através da equação 5.15, que é de aproximadamente, 8.

Com um *baud rate* de 115200 *bps*, significa que são necessários cerca de $8,6\mu s$ para enviar um bit. Como uma mensagem é constituída com um *start bit*, uma trama de 8 *bit* e um *stop bit*, perfazendo um total de 10 bit por mensagem, sabemos que para enviar uma mensagem são necessários cerca de $86\mu s$. O valor do sinal é passado para formato ASCII, através de uma função denominada de "*sprintf*"que converte um valor inteiro numa *string*[7] uma vez que é mais simples depois trabalhar este valor pela interface gráfica. Sabendo que cada caractere, no formato ASCII, corresponde a 8 bit, são enviados no caso extremo, cinco caracteres, quatro destes são relativos ao valor da conversão, (uma vez que estamos a trabalhar com uma resolução de 12 bit e 10 bit, o que leva a que o valor máximo da conversão possua quatro algarismos, 4093 e 1023 respetivamente) e mais um caractere que indica o fim da mensagem, o "$\backslash n$". Com 5 caracteres, o tempo total da mensagem fica cinco vezes maior, ou seja, aproximadamente 0,434 ms.

[7]Na programação de computadores, uma *string*, é uma sequência de caracteres, que pode ser usada para representar palavras, frases, ou então apenas letras.

Por fim, o *bit* correspondente as interrupções do registo SREG foi ativado para permitir que hajam interrupções dos módulos mencionados em cima. Com a explicação da configuração dos módulos, a primeira fase do código fica concluída.

Feita a explicação de como foram configurados os módulos, será demonstrada agora o funcionamento do restante código. Na função principal entramos num ciclo infinito que fica à espera que o sistema receba um dos caracteres que define o modo de operação que o utilizador pretende. No seguinte excerto de código é apresentado o ciclo mencionado. Na linha 121 é verificado se a variável relativa ao modo "tempo real"foi ativa e na linha 131 é verificado se a variável do modo "guardar dados"foi ativada. Daqui em diante não serão feitas mais referências as linhas de código, pois este já se encontra exposto nos fluxogramas das figuras 5.18, 5.19 e 5.20.

```
118     while(1)
119     {
120
121         if(flag_tempo_real==1)
122         {
123
124             if(flag_enviar==1)
125             {
126                 TIMSK=0b00010000;
127             }
128
129         }
130
131         if(flag_guardar_dados==1)
132         {
133
134             if(flag_enviar==1)
135             {
136                 TIMSK=0b10000000;
137             }
138
139         }
```

Quando é recebido um caractere, o sistema entra na interrupção da receção da USART. Para saber se a informação recebida é um dos caracteres que definem o modo de operação, a mensagem obtida é comparada com uma das letras *r* ou *g*. Caso coincida com uma das letras, é então ativada a variável correspondente ao modo de operação representado pela letra recebida. A variável relativamente ao modo "guardar dados"é denominada de "$flag\_guardar\_dados$", enquanto para o modo "tempo real"é chamada de "$flag\_tempo\_real$". Para ativar estas variáveis são lhes atribuído o caractere *1* e para serem desativas é lhes atribuído o *0*.

Depois de recebida e ativada a variável, voltamos à função principal onde esta fica à espera do sinal do *trigger*. Quando o sinal do *tigger* é recebido, o processo é o mesmo, o sistema vai a interrupção da USART e verifica se a informação que recebeu foi a letra *t*. Caso seja a letra *t*, é ativada a variável que dá inicio à leitura.

Como mencionado anteriormente, a conversão de AD é efetuada na rotina de interrupção do *timer* respetivo ao modo selecionado. Esta interrupção ocorre sempre que o valor do contador do *timer* é igual ao valor do OCR, o que faz com que no *timer2* a interrupção aconteça a cada 1 ms e no *timer1* seja a cada 32 ms. Na interrupção de cada *timer* ocorre sempre a conversão de AD, no entanto dependendo do modo de utilização, este pode ser o único acontecimento ou podem existir outros. No caso do modo "guardar dados"depois de obtido o sinal da conversão, o próximo passo é guardar esse valor num vetor.

Contudo, no modo "tempo real", para além da conversão, esse valor é transformado numa *string* para poder ser enviado pela USART. Este processo continua até que haja à receção da letra p, que leva a que o sistema entre na rotina de interrupção da USART e desative as variáveis mencionadas anteriormente e ative a variável, chamada de "$flag\_fim$"que indica que o processo foi parado.

De volta a função principal, caso estejamos a lidar com o modo "tempo real"é enviada a string "$0000 \backslash n$"para a interface gráfica para informar que o sistema parou com sucesso. Caso o modo da operação foi o modo "guardar dados", todos os dados armazenados no vetor, serão agora convertidos numa *srting* e enviados pela USART, quando a transmissão acabar é enviada também a *string* "$0000 \backslash n$". Com todas as variáveis relativas aos modos de funcionamento limpas, o sistema fica novamente à espera que receba a letra que indique o novo modo que o utilizador pretende usar.

5.3.2 Código - ARM

Tal como no AVR, no ARM também foi utilizado o *ecplise 4* para desenvolver o código. A linguagem escolhida foi em C.

O ARM utilizado neste projeto, funciona a uma frequência de 180 MHz, contudo, esta frequência varia de acordo com o conjunto de periféricos que queremos usar, uma vez que estão ligados a diferentes tipos de barramentos. Existem dois tipos de barramentos disponíveis, um de alta velocidade, APB2, que permite frequências máximas na ordem dos 90 MHz e um barramento de baixa velocidade, APB1, com uma frequência máxima de 45 MHz.

Uma das grandes diferenças, em termos de código, entre o ARM e o AVR, é que no ARM não trabalhamos com registos, mas sim com estruturas já pré-definidas, onde basta introduzir no campo apropriado o valor que queremos.

Tal como no AVR, no ARM também foram utilizados dois *timers*, ambos de 16 bits, um para cada modo de operação. Para criar o *timer* referente ao modo "guardar dados", com uma taxa de amostragem de 1 ms, foi preciso primeiro identificar o barramento a que o *timer4* pertence. De acordo com o manual do ARM, este módulo

encontra-se no barramento de baixa velocidade, no APB1. O seu funcionamento é idêntico ao *timer* do AVR, realiza uma contagem crescente e sempre que o valor do contador igualar o valor definido por nós é gerada uma interrupção. Para configurar este *timer* para produzir uma interrupção a cada 1ms (1000 hz), é necessário usar a equação 5.16:

$$f\_tim = \frac{clk\_tim}{(prescaler + 1) * period} \tag{5.16}$$

Onde f_tim é igual a 1000 Hz, que representa a frequência que queremos que o *timer* gere uma interrupção, $clk\_tim = 90MHz$ é a frequência do relógio do *timer*, como mencionado anteriormente a frequência para este *timer* deveria de ser de 45 MHz uma vez que está conetado ao barramento de baixa velocidade, mas como o ARM está a usar a PLL[8] para suportar uma frequência de 180 MHz, o valor de clk_tim é multiplicado por 2. Para existir apenas uma variável, o valor para o *period* foi definido como 1000. Substituindo todos os valores na equação 5.16 obtemos o valor para o *prescaler* (equação 5.17, que teve origem com a colocação da equação 5.16 em função do *prescaler*):

$$prescaler = -\frac{f\_tim \times period - clk\_tim}{(f\_tim \times period)} \tag{5.17}$$

Através da equação 5.17 foi então conhecido o valor para o *prescaler*, cerca de 89, para a configuração do *timer* ficar completa. Para implementar o segundo *timer* relativo ao modo "tempo real"todo o procedimento foi igual, à exceção da configuração para a taxa de amostragem, que neste modo é de 32 ms. Para conhecer o valor do *prescaler* a usar, foi usada novamente a equação 5.17, onde desta vez o valor para frequência do *timer* é de 31,25 Hz, substituindo os valores, obtemos o valor para o novo *prescaler* de aproximadamente 2879.

Com ambos os *timers* configurados, passou-se à implementação do módulo ADC. O ADC usado foi o ADC1, ligado ao barramento APB2, com uma resolução de 12 bits. No caso do ARM, não existe um intervalo de tempo pré-definido para que a conversão de um sinal analógico para digital deva demorar. Esse intervalo de tempo depende da frequência relógio do módulo ADC que se esteja a usar, no número de ciclos e da resolução. De acordo com o manual deste ARM, o valor da frequência do ADC que resulta em menos erros, quer de conversão ou de *offset* são os 30 MHz. Dada a equação 5.18, é possível encontrar o valor para o *prescaler*, com f_ADC de 30 MHz e clk_ADC1 igual a 90 MHz:

[8] A PLL é um mecanismo que pode ser usado para multiplicar o relógio interno ou externo, de forma a gerar frequências mais elevadas para o sistema.

$$f\_ADC = \frac{clk\_ADC1}{prescaler} \tag{5.18}$$

O valor dado pela equação 5.18 foi cerca de 3, contudo como os valores do *prescaler* estão pré-definidos em 2, 4, 6 e 8 não foi possível obter o valor de 30 MHz, por isso optou-se por utilizar o valor mais próximo, *prescaler* de 4. Substituindo o valor de *prescaler* com 4, na equação 5.18, obtemos um valor de f_ADC de 22,5 MHz.

Como mencionado anteriormente, o tempo de conversão depende da frequência do módulo ADC, do número de ciclos e da resolução. Conhecendo o valor da frequência e da resolução, faltou encontrar o valor adequado para o número de ciclos. Neste ARM, existem vários valores para o número de ciclos, sendo o mais baixo de 3, passando por 15, 28, 56, 84 ,112, 144 e 480. Para evitar de trabalhar na região limite, optou-se por escolher 15 número de ciclos para obtermos uma conversão completa. Com a equação 5.19, é apresentado o tempo que demora até haja uma conversão completa, cerca de $8\mu s$.

$$t\_conv = \frac{1}{f\_ADC} \times n\_ciclos \times resolucao \tag{5.19}$$

Foi encontrado assim o tempo que demora para que haja uma conversão completa do sinal analógico para digital. Com todo o módulo ADC configurado, falta apenas explicar a implementação da *USART*. A configuração da *USART* é semelhante a do AVR, com 1 *stop bit*, sem paridade, com um *baud rate* de 115200. Para indicar qual o *baud rate* a usar, no ARM, basta preencher o campo de "*baud rate*"na estrutura da *USART* com o valor de 115200 *bps*, não sendo preciso aplicar nenhuma fórmula. Por último foram configurados os módulos do *timer* e da *USART* para gerarem interrupções, através da estrutura do ARM "*NVIC*"que permite indicar qual a prioridade que queremos que cada uma das interrupções possua. A interrupção da receção da *USART* ficou definida como mais prioritária do que as interrupções dos *timers* que possuem o mesmo nível de prioridade entre eles. O maior nível de prioridade da interrupção da USART deve-se ao facto de quando o sistema receber algum caractere, este faça com que a interrupção seja atendida imediatamente não esperando até que o *timer* que esteja ativo termine o seu processo, contribuindo para um maior rigor dos tempos de inicio e fim da leitura do sinal, um dos objetivos principais do desenvolvimento deste projeto.

Com a configuração do módulo das interrupções, a etapa de configuração dos módulos é dada como concluída. Relativamente à descrição do restante código, este é idêntico ao descrito na subseção do AVR, como pode ser visto na seção dos anexos.

5.4 Interface gráfica

A criação de uma interface gráfica foi um dos requisitos adicionais pedido pelos médicos fisioterapeutas. A interface gráfica deveria de ser capaz de mostrar, pelo menos um gráfico representativo do sinal EMG, de dar a ordem de início para leitura do sinal e guardar um registo dos valores obtidos.

O primeiro *software* que se usou para criar a interface gráfica foi o MaTrix Laboratory R2015a (MatLab), contudo, apesar de o sinal ser lido e mostrado num gráfico com sucesso, à medida que o programa recebia mais dados, este tornava-se cada mais mais lento e a operação em modo "tempo real"não era válida, pois o sinal mostrado no gráfico já não correspondia à realidade, devido ao atraso na amostragem do sinal. Para além, do problema enumerado anteriormente, o MatLab não é um *software* livre, o que levava a que, caso a interface fosse toda implementada neste software, o laboratório teria de obter obrigatoriamente uma licença para poder usar esta interface, tornando o projeto com um custo elevado apenas devido à compra da licença. Com estes dois contratempos, foi decidido avançar para outro *software* que não apresentasse os mesmos problemas e que não adicionasse mais constrangimentos. Após a pesquisa e o estudo efetuado, o melhor *software* encontrado foi o QT Creator 4. Este é um programa leve, possui a capacidade de comunicação através de uma porta série, permite trabalhar matematicamente os valores recebidos, suporta várias linguagens como por exemplo: C++, QML e JavaScript, possui uma interface gráfica com uma implementação simples e amigável para o utilizador com varias opções e funções, é livre (uma grande vantagem quando comparado com o MatLab) e funciona tanto em *linux* como em *windows*.

A interface gráfica foi criada com recurso a linguagem C++. Antes de ser implementada uma comunicação pela porta série no QT Creator, foi necessário emparelhar o módulo Bluetooth com o computador.

Para emparelhar o módulo Bluetooth, HC-06, com o sistema operativo em uso é preciso navegar até ao software de "assistente de configuração de módulos Bluetooth", clicar sobre o nome do módulo que queremos conectar e inserir o *pin* que vem associado a este tipo de módulos, por defeito este *pin* é o 1234. Para estabelecer a conexão, no *ubuntu* 14.04 entre o módulo *bluetooth* do computador e do HC-06, foi preciso recorrer ao terminal onde foram inseridos os seguintes comandos: "*sudo hcitool scan*"para listar todos os módulos *Bluetooth* visíveis (Figura 5.23):

Fonte: print screen do software consola, elaborado pelo autor.

```
flavio@FlavioPortatil:~/Área de Trabalho$ sudo hcitool scan
[sudo] password for flavio:
Scanning ...
        20:16:05:26:97:48       HC-06
flavio@FlavioPortatil:~/Área de Trabalho$
```

Figura 5.23: *Lista dos módulos Bluetooth.*

Como visível na Figura 5.23, o último comando retornou o nome do módulo *Bluetooth* usado neste projeto, o HC-06, juntamente com o seu endereço (20 : 16 : 05 : 26 : 97 : 48). Com o comando "*sudo rfcomm bind /dev/rfcomm1 20:16:05:26:97:48*"é criada uma nova porta designada de "*rfcomm1*"onde é atribuído o endereço do HC-06 obtido com o comando anterior. Relativamente ao *Windows*, depois de introduzido o *pin*, o *Windows* faz o emparelhamento com o módulo e cria automaticamente uma porta série. Para saber qual a porta série a que o módulo ficou associado é preciso ir até ao "configurador de dispositivos"e clicar sobre a categoria "Portas (COM e LPT)"onde é apresentada a porta a que este ficou associado. Na Figura 5.24 é apresentado o nome da porta a que o módulo HC-06 ficou associado, COM4.

Fonte: print screen do software configurador de dispositivos, elaborado pelo autor.

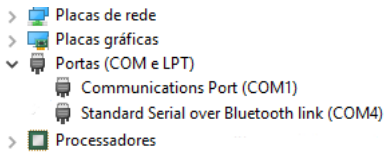


Figura 5.24: *Configurador de dispositivos.*

De referir que o processo de conexão ao módulo HC-06 no *Ubuntu* é preciso efetuar sempre que seja iniciada uma nova seção, enquanto no *Windows* a porta série fica sempre configurada. Após explicado o processo de emparelhamento e criação de uma porta série, já é possível estabelecer uma comunicação entre o QT Creator e o módulo Bluetooth. Na Figura 5.25 é apresentado o fluxograma da interface gráfica.

Fonte: elaborado pelo autor.

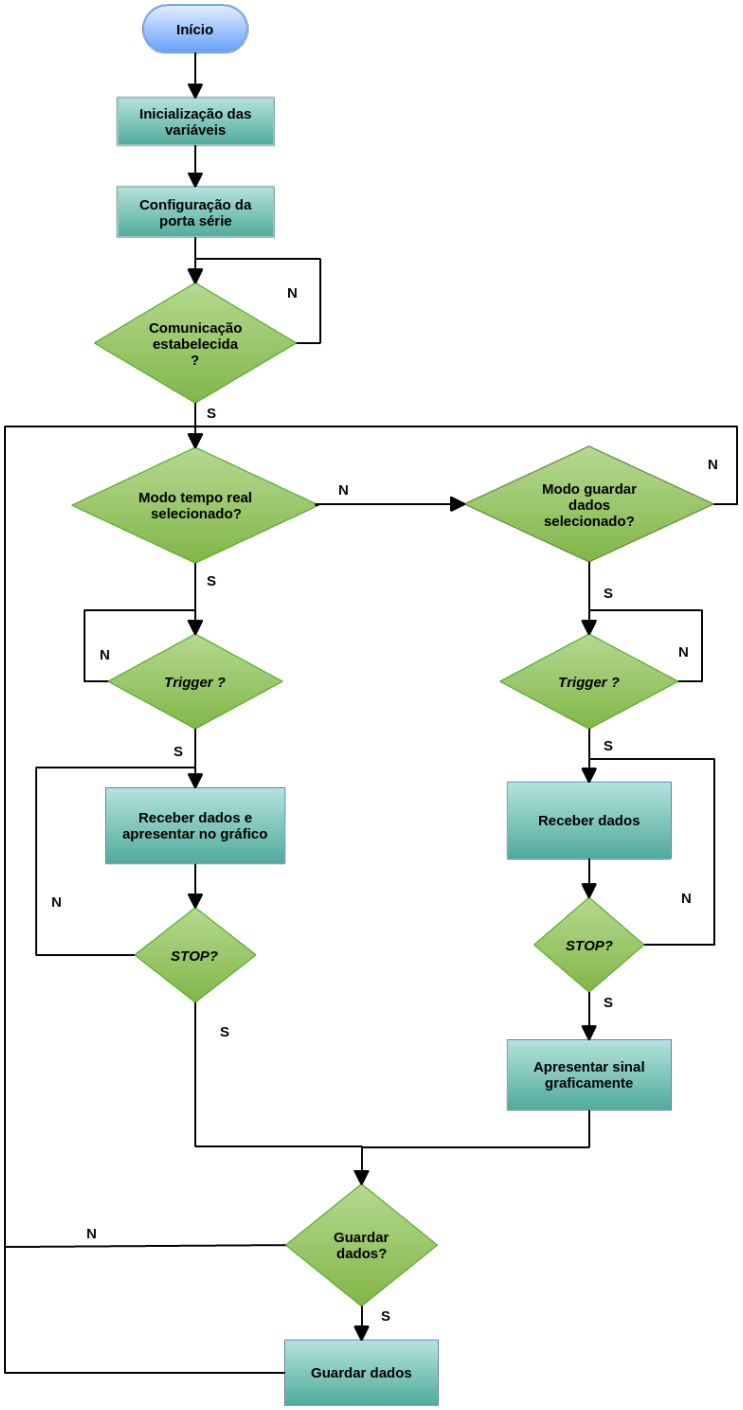


Figura 5.25: *Fluxograma da interface gráfica.*

Começando a explicar o fluxograma da interface gráfica, primeiramente foi criada

a porta série do lado do QT Creator. Para isso, foi necessário indicar qual a porta série do HC-06, $rfcomm1$ no *Ubuntu*, para o *Windows* fica $com4$. Foi configurada uma comunicação igual à do AVR e do ARM, com um *baud rate* de 115200 *bps*, uma trama de 8 *bits*, sem paridade, um *stop bit*. Sempre que são recebidos dados pela porta série, o programa chama a função "*readData*".

Com a configuração da porta série concluída, fica a cargo do utilizador escolher quando é que a ligação deve ser estabelecida ou interrompida. Para oferecer esta possibilidade foram criados dois botões com o propósito de conectar e desconectar. A interface gráfica deste software permite escolher que objetos (como por exemplo botões, tabelas, listas), que queremos implementar no ambiente gráfico, bastando arrastá-los para a interface gráfica. Neste caso foram escolhidos dois botões, que foram nomeados de "ligar"e "desligar", onde as suas ações são ativadas quando o utilizador clica apenas uma vez sobre eles. Dentro do botão "ligar"a comunicação com a porta série é aberta e fica a espera que o programa estabeleça corretamente a ligação, caso a comunicação não seja estabelecida com sucesso, o programa retorna uma mensagem de erro e é pedido para que o utilizador tente novamente. Quando a comunicação é estabelecida, este botão torna-se desativado, ativando o botão "desligar"que permite encerrar a comunicação. A opção de ativar e desativar os botões, foi uma forma de proteger o programa para que o utilizador não aceda a funções que ainda não foram ativadas/configuradas. Na Figura 5.26 são apresentados os dois botões mencionados, de notar que o botão "desligar"aparece ativado juntamente com o botão "ligar"apenas para efeito ilustrativo, uma vez que esta condição nunca ocorre quando se está a efetuar um teste.

Fonte: print screen do programa desenvolvido, elaborado pelo autor.

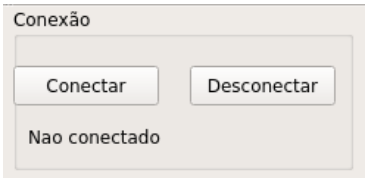


Figura 5.26: *Botões ligar e desligar.*

Depois de estabelecida uma comunicação com a placa de aquisição, é preciso indicar ao microcontrolador qual o modo que deve operar, se em "tempo real"ou no modo de "guardar dados". Para o utilizador realizar essa escolha, foram implementados na interface gráfica dois botões de rádio e um botão "normal". Em

cada um dos botões rádio, existe a opção de escolha para cada um dos modos, onde o estado de uma variável, ativada ou desativada, indica ao sistema qual a escolha do utilizador, a variável ativa representa que o modo escolhido foi o "tempo real", enquanto a variável destivada indica que o modo escolhido foi o "guardar dados". O terceiro botão, com o nome de "confirmar", envia o caractere para o microncontrolador conforme o estado atual da variável, com a letra *r* para o modo "tempo real"e a letra *g* para o modo "guardar dados". De seguida é apresentada a Figura 5.27 que apresenta os botões que permitem selecionar o modo de operação e o botão de confirmação do modo escolhido. De referir que devido as características dos botões radiais, este não podem estar ativos em simultâneo, o que os torna ideias para a escolha do modo a operar, visto que apenas um dos modo é que pode estar ativo.

Fonte: print screen do programa desenvolvido, elaborado pelo autor.

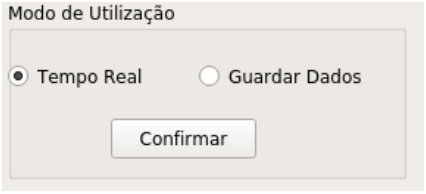


Figura 5.27: *Botões para a escolha do modo a operar.*

Com o modo a operar selecionado, falta dar a ordem de início de leitura do sinal. Este ordem é dada através de um clique num novo botão "inicio"que, quando ativado envia a letra *t*, dando assim o sinal de *trigger* para que o microcontrolador comece a enviar o valor do sinal para o programa. Para que o utilizador não esteja a alterar o modo de operação enquanto decorre uma leitura de sinal, os botões de rádio, o botão "confirmar"e o botão "inicio"são desativados voltando a ficar ativados apenas com o atual modo de leitura seja terminado.

Com o sinal de *trigger* recebido pelo microcontrolador este começa então a enviar os dados, o que ativa a função de receção de dados, do programa, "*readData*".

Dentro da função "*readData*", o valor enviado pelo microcontrolador é recebido e guardado num *buffer*[9], dependendo do modo que se escolheu para a operação, este valor pode ser imediatamente apresentado num gráfico e o *buffer* é limpo, modo "tempo real", ou continuar a ser guardado no *buffer* até que um sinal que indique o fim da emissão de dados ocorra, fazendo com que todos os valores recebidos até aquele instante sejam passados para um vetor onde serão posteriormente apresentados num gráfico. Relativamente à escala de tempo, representada no eixo dos x

[9]Um *buffer*, é uma região de memória física utilizada para armazenar temporariamente os dados.

nos gráficos, o valor da variável tempo é incrementado sempre que haja a receção de um valor, por exemplo, como é sabido a taxa de amostragem do modo "guardar dados"é de 1 ms, logo esta variável será aumentada de 1 em 1 sempre que se receba um valor durante este modo, o mesmo acontece para o modo "tempo real", mas com um incremento de 32, já que a taxa de amostragem é de 32 ms.
Depois de recebido o valor do sinal EMG este é convertido para que seja apresentado em V. Para isso, foi necessário medir com um multímetro qual o valor real de tensão para o pino de tensão de referência para o ADC de ambos os microcontroladores. Depois de efetuada a medição, o valor obtido situa-se nos 2,88 V o que significa que, para o caso do ARM, como a resolução é de 12 *bits*, o valor de 4095 corresponde a 2,88 V. Sabendo esta condição, é possível converter todos os valores que são recebidos em bits e apresentá-los em V, através da equação 5.20. O mesmo processo acontece para o AVR, mas o valor máximo de 2,88 V corresponde a 1024 bits.

$$conv\_do\_sinal\_recebido = \frac{2,88 \times valor\_recebido}{4095} \tag{5.20}$$

Após a conversão do valor do sinal tenha sido efetuada, o sinal é apresentado em forma de gráfico. Com um gráfico já apresentável, foi criada uma nova janela onde é apresentado um segundo gráfico em que foram implementados um conjunto de técnicas para melhorar a forma como o sinal é representado. Primeiramente foi encontrado o valor do sinal quando o músculo esta em repouso. Esse valor foi definido como 0, o que o primeiro método, chamado de retificação [56], passa todos os valores que estejam abaixo de 0 para valores positivos. Por exemplo, se o valor em repouso for de 500 e existirem dois valores inferiores a 500, um de 495 e outro de 490, este dois valores passariam a 505 e 510 respetivamente. Este método faz com que todos os valores sejam apresentados apenas em um dos hemisférios do gráfico. A segunda técnica aplicada, foi de calcular o valor médio dos valores, dentro de uma janela com um certo número de amostras, eliminando assim os valores de pico fazendo com que o desenho do sinal EMG se torne mais suave. Este segundo gráfico é apresentado em simultâneo ao gráfico original, por debaixo deste, para o utilizador poder assim visualizar os dois gráficos e reparar nas melhorias implementadas.

Para indicar ao microcontrolador o fim da leitura e envio dos dados do sinal EMG foi criado um botão, nomeado de "fim", que quando clicado envia a letra p para o microcontrolador. Para o programa receber a confirmação que o microcontrolador recebeu a ordem para terminar de enviar dados, o microcontrolador envia a *string* "0000 $\backslash$ n". Na Figura 5.28 estão apresentados os botões de "inicio"e "fim".

Fonte: print screen do programa desenvolvido, elaborado pelo autor.

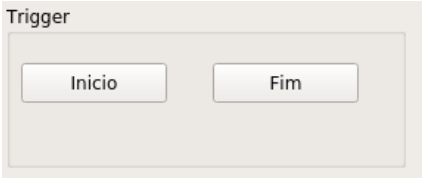


Figura 5.28: *Botões para o início e fim da leitura do sinal.*

Depois de indicado o fim da leitura do sinal, foi implementada a opção de guardar todos os registos dos valores obtidos até aquele instante, como também os gráficos. Para isso foi criado um novo botão, "guardar registo atual"(ver Figura 5.30), que quando clicado abre uma janela que pergunta ao utilizador qual o nome que deseja associar ao registo que pretende guardar. Esta janela possui uma caixa de texto onde deve ser introduzir o nome do registo, dois botões, um botão chamado de "Ok"que confirma o nome inserido na caixa de texto e um botão nomeado de "Cancelar"para cancelar a ação de guardar os registos. Na Figura 5.29 é apresentada a janela da opção de guardar os registos do teste realizado.

Fonte: print screen do programa desenvolvido, elaborado pelo autor.

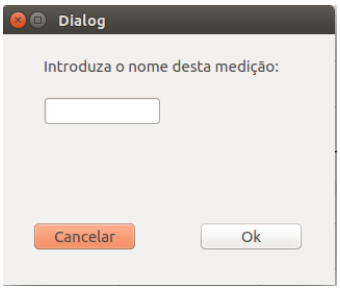


Figura 5.29: *Nova janela para registo dos valores obtidos.*

Fonte: print screen do programa desenvolvido, elaborado pelo autor.

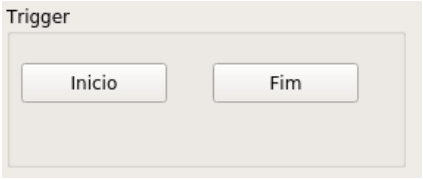


Figura 5.30: *Botão para o guardar o registo da experiência atual.*

Com o nome introduzido pelo utilizador, é criada uma nova pasta no diretório do programa, (este diretório pode ser alterado no código, mas por defeito fica associado a pasta onde se encontra o executável da interface gráfica) com o nome atribuído pelo utilizador, onde são guardados os dois gráficos com a extensão *png*, mais um ficheiro em formato de texto com todos os valores do sinal recebido associados a determinado instante de tempo. Para diferenciar as duas figuras guardadas, a imagem do gráfico associada ao sinal original sem qualquer tratamento, fica com o nome pedido ao utilizador mais o sufixo de "$\_sinal\_da\_placa$", enquanto que para o gráfico do sinal já tratado, o nome do imagem fica, nome introduzido pelo utilizador mais o sufixo "$\_sinal\_final$". O ficheiro de texto com os registos do tempo e do valor de tensão do sinal é chamado de "log".

Por último, o programa limpa todas as variáveis e retorna a etapa onde pede ao utilizador para escolher entre os dois modos de operação. Todo o código desenvolvido para a criação da interface gráfica está disponível na seção do anexo C. O aspeto final da interface gráfica é mostrado na Figura 5.31.

Fonte: print screen do programa desenvolvido, elaborado pelo autor.

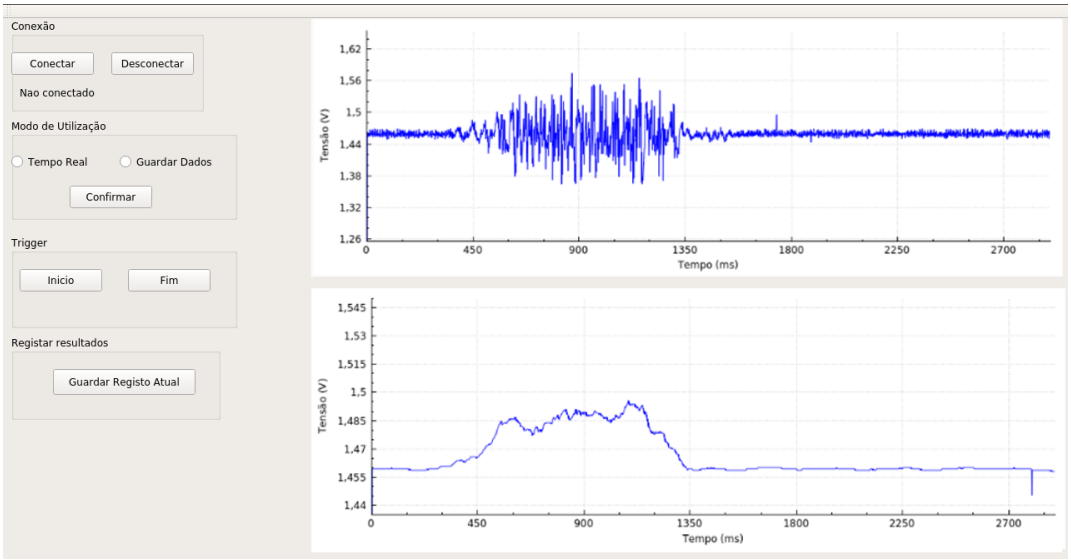


Figura 5.31: *Aspeto da interface gráfica desenvolvida.*

Como é visível na Figura 5.31, os botões mencionados anteriormente estão dispostos no lado esquerdo, agrupados dentro de uma janela, cada uma representada com um nome que carateriza a função de cada um dos botões que essa janela agrupa, com um aspeto simplista para dar um efeito mais "limpo"ao utilizador. No meio da

interface gráfica existem dois gráficos, em que o gráfico de cima corresponde ao sinal EMG original transmitido pelo microcontrolador e o segundo gráfico é relativo ao sinal EMG trabalhado, com um conjunto de técnicas matemáticas que o tornam mais apresentável para o utilizador, sem perder as informações que o tornam essencial para uma avaliação médica.

6

Resultados

Neste capítulo serão apresentados os resultados obtidos com a implementação dos sistemas criados para efetuar a leitura de um sinal EMG. Para além da apresentação dos resultados, será feita também uma análise ao sinal obtido, comparando-o com o sinal do sistema biopac, assim como, descobrir qual o microcontrolador mais indicado para um projeto desta dimensão, indicando as vantagens e desvantagens de cada um. Será apresentado também o mecanismo que permite ao utilizador guardar o registo dos testes de EMG efetuados. Daqui para a frente quando for mencionado o sistema AVR, é relativo à montagem da placa de aquisição em conjunto com o microcontrolador ATmega32, quando for referido o sistema ARM, é referente à montagem da placa de aquisição com o microcontrolador ARM.

Para perceber se o sinal EMG obtido é valido, este foi comparado com o sinal do sistema biopac, onde a experiência foi recriada nas mesmas condições para todos os sistemas, de forma a que as únicas divergências que possam surgir sejam apenas no tipo do sinal lido. Antes da colocação dos elétrodos, a zona onde foi lido o sinal foi limpa e foram removidos os pelos para não interferirem com a qualidade da leitura. A colocação dos eletrodos foi realizada de acordo com o protocolo SENIAM [25], (ver secção 2.4.4), no braço dominante do utente, braço direito. A experiência consistiu em registar a atividade muscular dos músculos do ante-braço, causada pelo aumento gradual da força aplicada com o encerramento do pulso, em que este estava dois segundos em repouso e os dois segundos seguintes encerrado, o processo foi repetido num total de 4 vezes, somando um tempo total de aproximadamente 18 segundos.

Na Figura 6.1 é apresentada a disposição dos elétrodos durante a experiência.

Fonte: elaborado pelo autor.

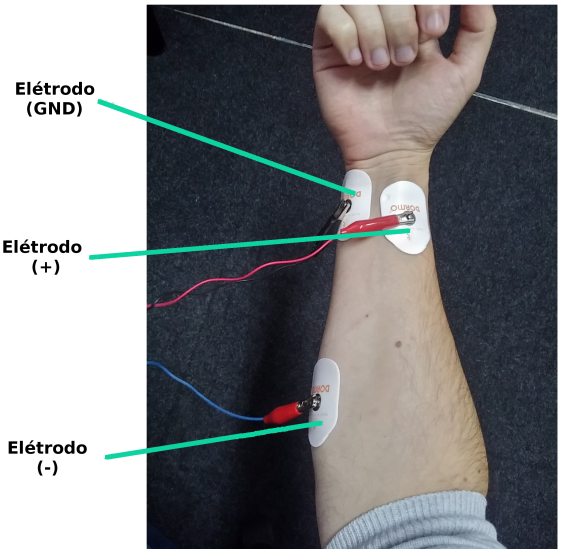
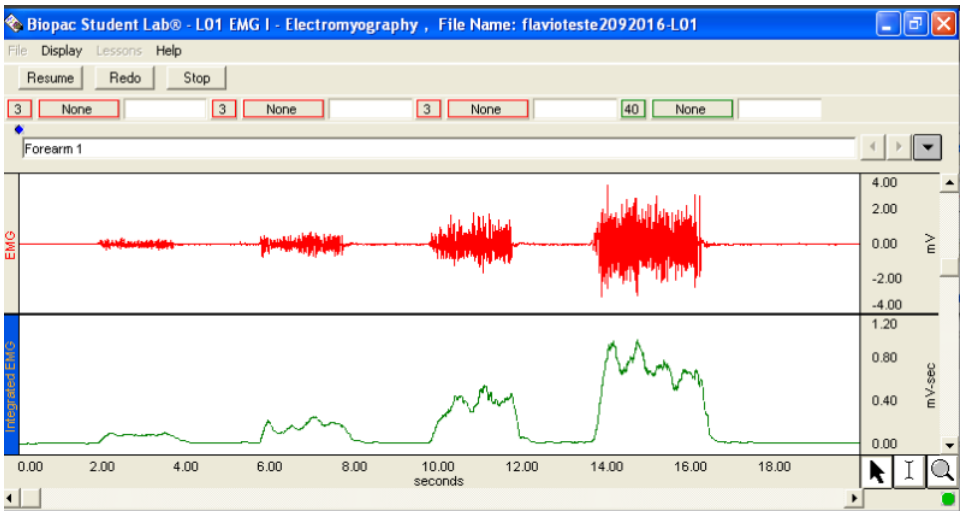


Figura 6.1: *Disposição dos elétrodos da experiência.*

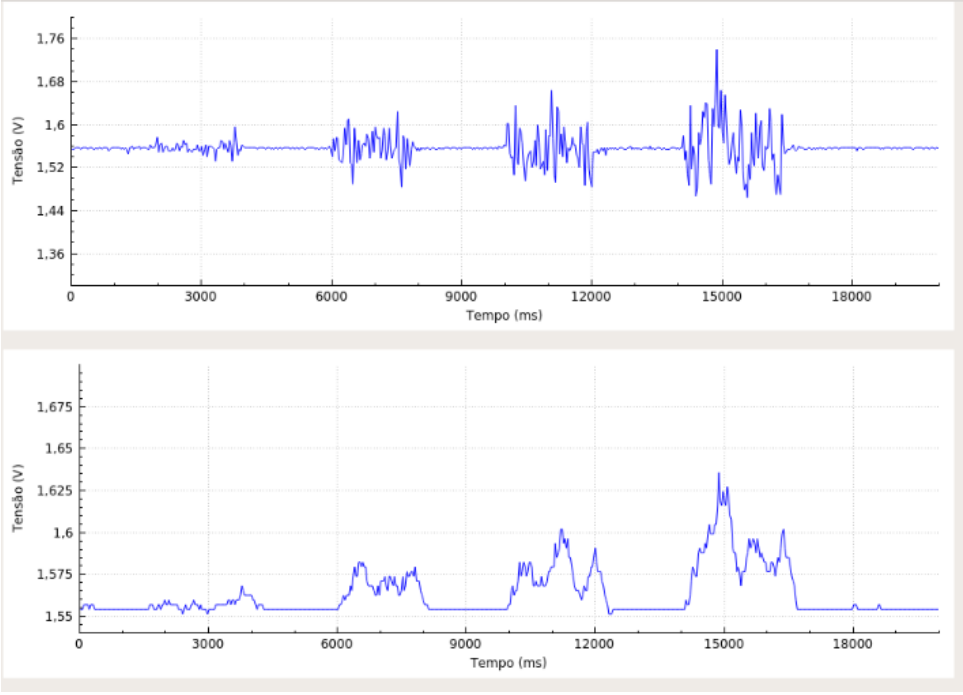
A taxa de amostragem com o sistema biopac não é conhecida e neste modo não é possível alterar, contudo depois de vários testes efetuados com este sistema com diferentes taxas de amostragens, foi possível concluir que o sinal esteja a ser amostrado a uma taxa maior ou igual a 1 ms, enquanto que para o sistema desenvolvido a taxa de amostragem para este modo de operação, "tempo real", situa-se nos 32 ms. Em ambos os sistemas o eixo do x representa o tempo, no biopac em segundos e no sistema AVR em milissegundos, com um tempo total de cerca de 20 segundos. No eixo do y, o biopac expressa a tensão do sinal em mV, com um intervalo de valores de -4 mV a 4 mV para o sinal EMG original e para o sinal já tratado, com um intervalo de valores de 0 mV a 1,2 mV. Para o sistema AVR, o eixo do y é representado por V, com uma janela de valores de 1,40 V a 1,74 V e para o sinal tratado o intervalo de valores é de 1,545 V até 1,65 V. Quer para o sistema AVR como para o sistema ARM, o eixo do x está dimensionado para apresentar todos os valores dentro dos primeiros 20 segundos, quando este valor é ultrapassado, deixa de ser apresentado o último registo do sinal associado ao valor de tempo mais antigo para que surja o mais recente de forma a que se mantenha sempre com uma janela de 20 segundos, independentemente do tempo total da experiência. Relativamente ao eixo do y, estes valores são ajustáveis, e foram definidos após vários testes com diferentes utilizadores para perceber qual era o valor máximo e minimo de cada

uma das janelas que deveria ser usado. O sinal obtido com o sistema biopac e com o sistema de aquisição com a ATmega32 é apresentado na Figura 6.2.

Fonte: a) print screen do software biopac e b) do programa desenvolvido, elaborado pelo autor.



a)



b)

Figura 6.2: *a), sinal do biopac. b), sinal do sistema AVR no modo "tempo real".*

O resultado obtido com o sistema desenvolvido, relativamente ao AVR, é bastante satisfatório, uma vez que o sinal obtido é semelhante ao registado pelo biopac e é possível identificar os momentos em que o músculo foi ativado cumprindo assim com o objetivo proposto com a criação deste projeto. Contudo, como a taxa de amostragem é menor, 32 ms, o gráfico não possui a mesma densidade de pontos comparativamente ao sinal do biopac e os valores de pico surgem com maior brutalidade, já que existem menos amostras entre os 32 ms para suavizar esse aumento de valor. Para perceber como seria o gráfico do biopac com uma taxa de amostragem perto dos 32 ms, foi usado o programa em modo livre, sem as configurações já criadas para o sistema estar pronto para ler determinado tipo de sinal. Foi então escolhido um tempo de amostragem que se aproximasse mais dos 32 ms, infelizmente, como não é possível introduzir o valor que queremos, foi escolhido um valor, dentro da gama que eles dispõem, sendo 20 amostras por segundo o valor mais próximo do desejado. Na Figura 6.3 é visível o gráfico obtido com sistema biopac com um tempo de amostragem de 50 ms, onde a grandeza representada pelo eixo do x é em s, com um tempo total de 8 ms e no eixo do y é apresentado o valor do sinal em mV, com uma janela de -11,60 mV até -2,90 mV.

Fonte: print screen do software biopac, elaborado pelo autor.

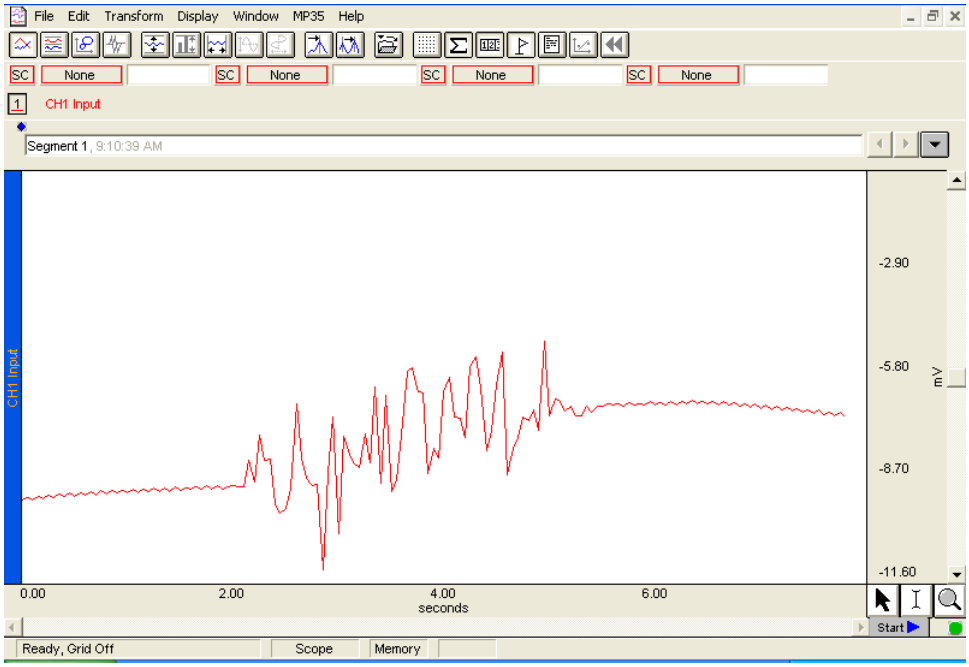


Figura 6.3: *Sinal do sistema biopac, com 20 amostras por segundo.*

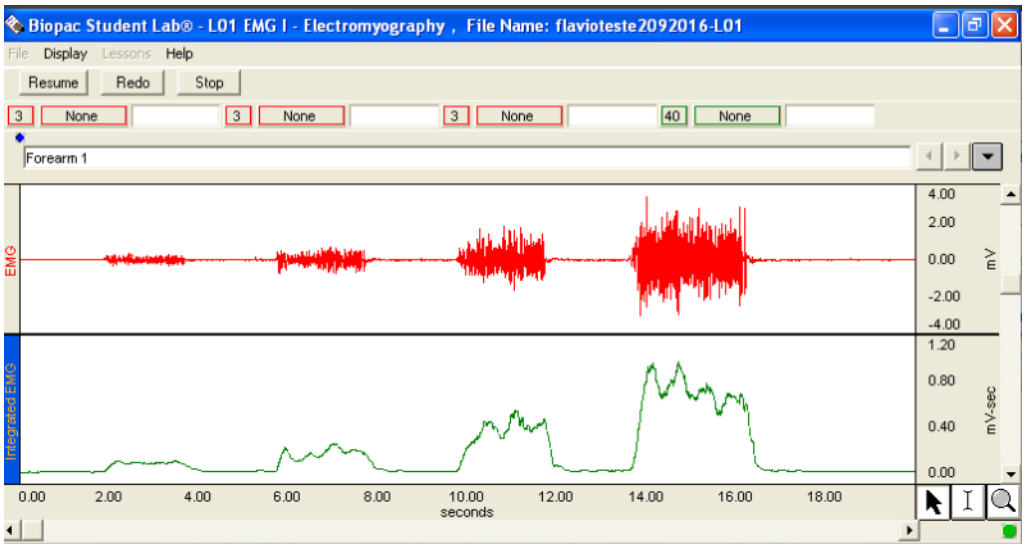
Como se pode concluir pela Figura 6.3, quando o sinal possui menos amostras por segundo é natural que os valores de pico fiquem mais isolados e que o sinal não seja tão denso como quando se esta a trabalhar com uma frequência de amostragem maior. Concluindo, o sinal obtido com o sistema AVR é valido e pronto a ser implementado num projeto de eletromiografia.

Relativamente ao sistema com o ARM, a comparação do sinal obtido em modo "tempo real", com o sinal de biopac é apresentado na Figura 6.4. Para o sistema biopac, os eixos são os mesmos mencionados na experiencia de comparação dos sinais entre o sistema do biopac e o sistema do AVR.

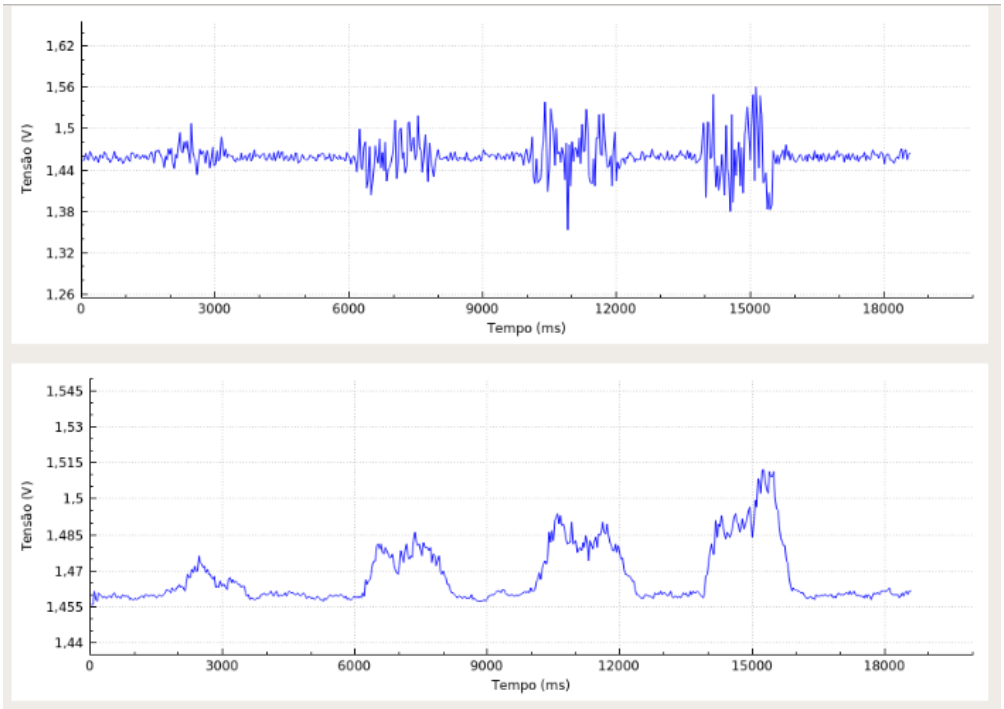
Para o sistema ARM, o eixo do x representa o tempo em ms, com um tempo total de 20 segundos e o eixo do y é representado por V, com uma janela de valores de 1,26 V a 1,62 V e para o sinal tratado o intervalo de valores é de 1,44 V até 1,545 V. O sinal EMG obtido com o sistema ARM, com uma taxa de amostragem de 32 ms, é muito semelhante ao gráfico do sistema do biopac, sendo possível perceber os instantes onde o músculo foi ativado, indicando que este sistema é valido e capaz de ler sinais EMG com sucesso.

A menor densidade de pontos e alguns valores de pico mais notórios, tal como no sistema AVR, é devido ao tempo de amostragem ser inferior ao do sistema biopac.

Fonte: a) print screen do software biopac e b) do programa desenvolvido, elaborado pelo autor.



a)



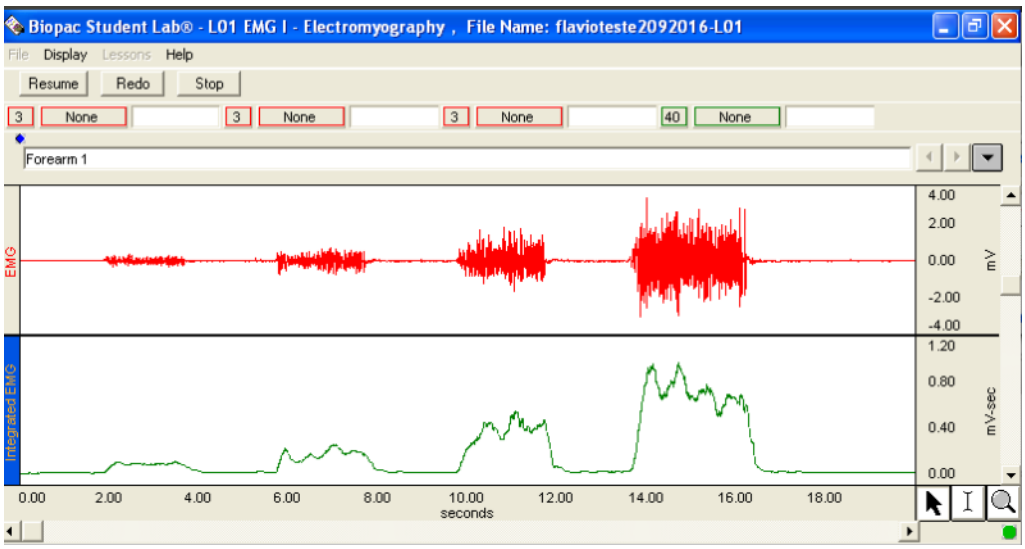
b)

Figura 6.4: *a), sinal do biopac. b), sinal do sistema ARM no modo "tempo real".*

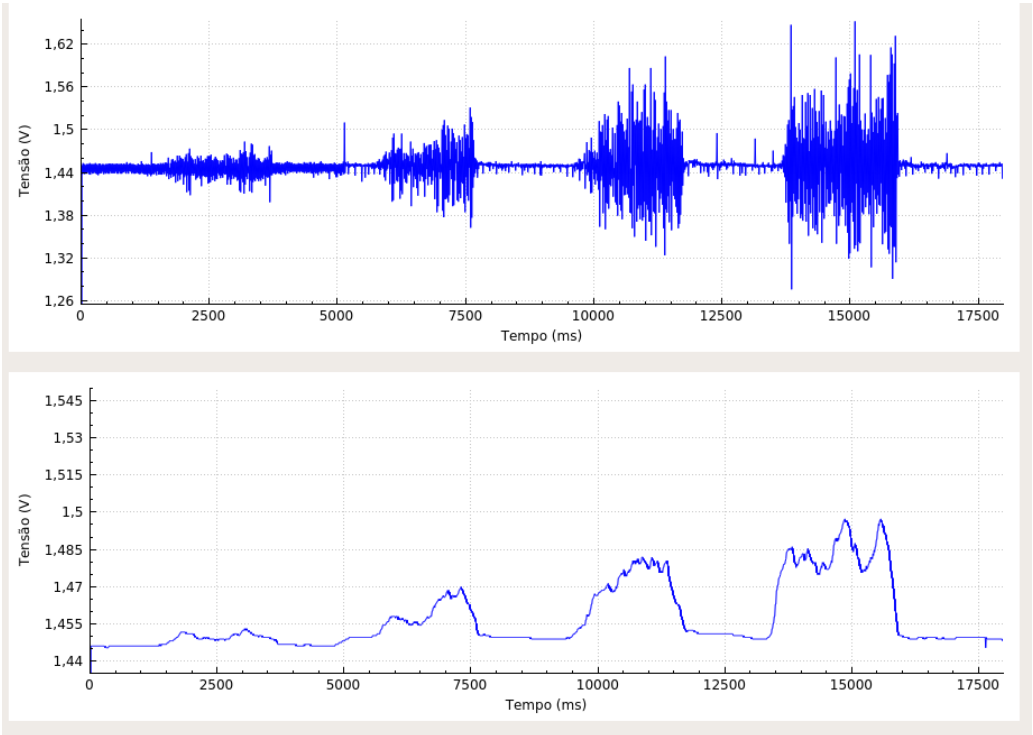
Após apresentados os sinais obtidos com o modo "tempo real", serão agora apresentados os gráficos correspondentes ao modo "guardar dados". Neste modo o tempo de amostragem já é de 1 ms, e o sinal obtido é ainda mais semelhante ao do sinal do sistema biopac, uma vez que possui mais amostras por segundo. Na Figura 6.5 é apresentada a comparação entre o gráfico do sinal EMG do biopack e o sinal EMG do sistema ARM a operar no modo "guardar dados". Os instantes de ativação e repouso do músculo não correspondem exatamente ao intervalo de tempo desejado, de 2 segundos, porque neste modo não existe *feedback* do sinal em tempo real para quem esta a realizar o teste de EMG, para conseguir acertar eficazmente nos intervalos de tempo corretos. Contudo os momentos de ativação e desativação do músculo não ficaram muito distantes do intervalo de tempo pretendido, o que é bastante satisfatório dado as circunstâncias enumeradas.

Com a apresentação do modo de operação "guardar dados"para o sistema ARM, ficam concluídas as demonstrações dos resultados obtidos com este microcontrolador, para os dois modos de funcionamento, com uma nota positiva para ambos os resultados dados por este sistema.

Fonte: a) print screen do software biopac e b) do programa desenvolvido, elaborado pelo autor.



a)



b)

Figura 6.5: *a), sinal do biopac. b), sinal do sistema ARM, modo "guardar dados".*

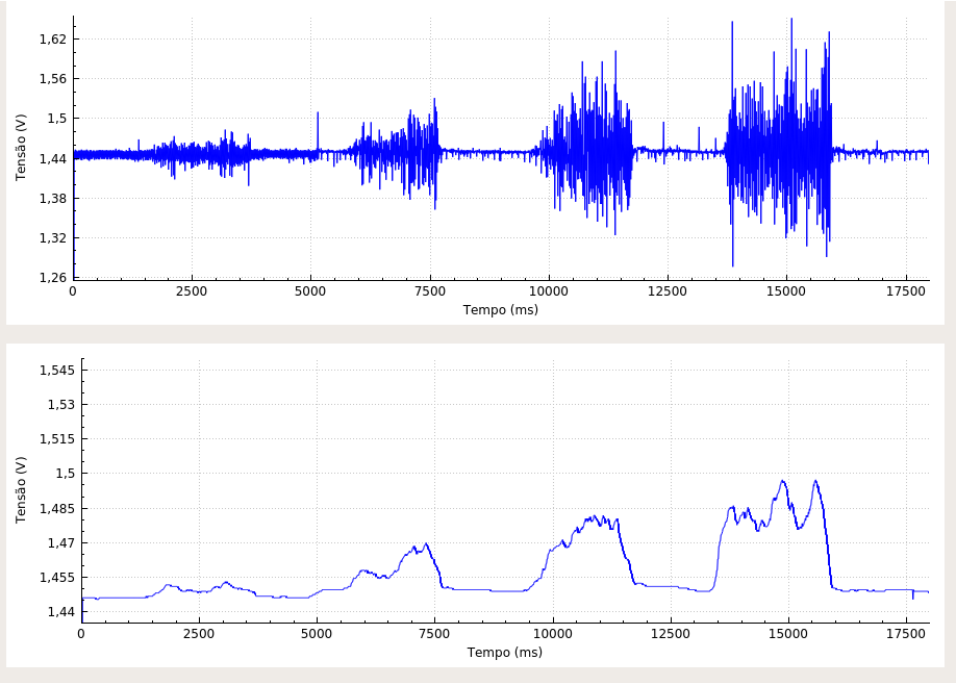
Relativamente ao sinal do sistema AVR no modo "guardar dados", surge um problema. Como a ATmega32 possui apenas uma SRAM de 2kbytes e cada amostra da conversão é uma variável de 2 bytes (a resolução do ADC é de 10 bits), este sistema, teoricamente apenas consegue armazenar em memória os valores do sinal durante 1 segundo, uma vez que a taxa de amostragem neste modo é de 1 ms. Contudo, a Atmega32 precisa de guardar outras variáveis, que embora não sejam em tão grande número como as amostras do sinal, ocupam o seu espaço na memória, o que reduz ainda mais o espaço na SRAM disponível para as amostras.

Com o código completamente funcional, o espaço deixado na SRAM permite criar um vetor com 900 posições, correspondendo a um tempo de aproximadamente 900 ms. Como o objetivo deste projeto, numa fase inicial, passa por registar o sinal EMG durante uma queda livre, ou seja, a ação da tampa abrir e chegar a um batente de forma a simular uma entorse, o tempo para que esse acontecimento aconteça ronda os 500 ms e os 700 ms. Embora a ATmega32 consiga registar todos os valores dentro desse intervalo de tempo, a ideia deste projeto é ser versátil e que seja ajustável a novas necessidades, como adicionar ao sistema de simulação de entorse, um mecanismo que controle a velocidade com que a tampa abra, levando a que o tempo da experiência aumente e a ATmega32 já não consiga cumprir com o seu objetivo de registar o sinal EMG durante o teste. Concluindo, foi comprovado que este microcontrolador é suficiente para um projeto inicial, contudo, não dota o produto com capacidade de evoluir. Na Figura 6.6 está representado o sinal EMG da ATmega32 com o modo "guardar dados", onde embora pareça possuir menos amostras do que o sistema com o ARM, a verdade é que a janela de amostragem é bastante inferior ao do sistema com o ARM.

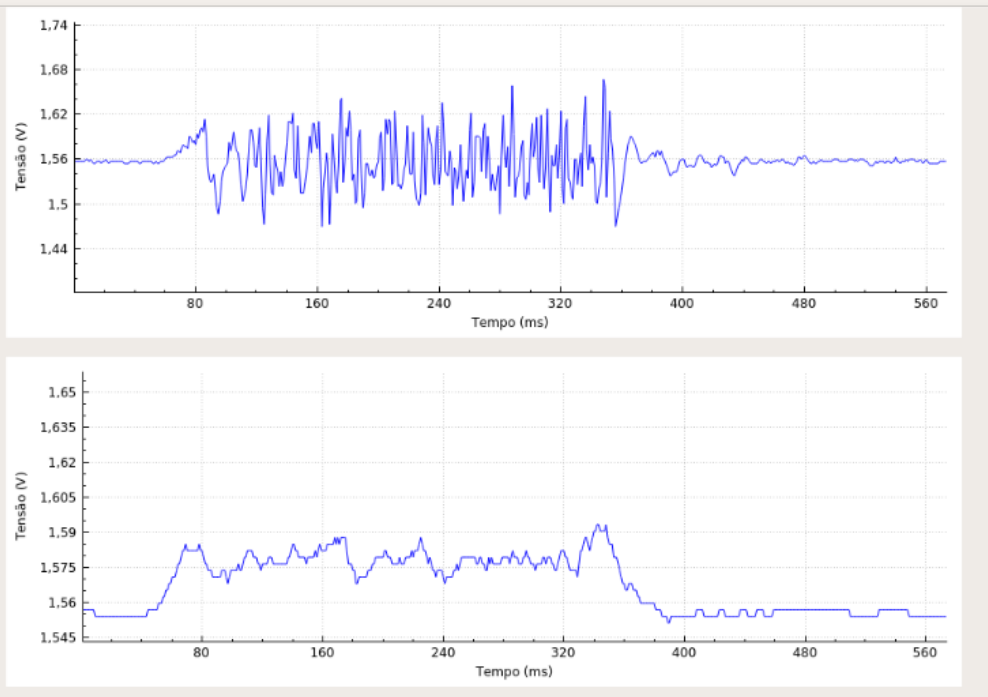
Nesta experiência o tempo total foi de, aproximadamente, 560 ms enquanto no sistema com o ARM tempo da experiência anda à volta dos 18 segundos. Em relação ao eixo dos y, este é expresso em V, com o intervalo de valores iguais a experiência anterior.

Com a apresentação do gráfico obtido com o sistema AVR no modo de "guardar dados", fica concluída a fase de apresentação de resultados com um balanço extremamente positivo, pois o objetivo de leitura do sinal EMG foi conseguido com sucesso nos dois sistemas implementados. De referir que o utilizador pode aumentar e diminuir as escalas do gráfico, interagindo com a bola do rato, movendo-a num sentido de sul para norte para para focar num ponto do gráfico ou diminuir, através da ação de rodar a roda do rato num sentido de norte para sul, para conseguir obter uma vista mais abrangente de todo o gráfico.

Fonte: print screen do programa desenvolvido, elaborado pelo autor.



a)



b)

Figura 6.6: *a), sinal do sistema ARM, modo "guardar dados", b), sinal do sistema AVR, modo "guardar dados".*

Caso o utilizador deseje guardar os registos obtidos do teste de eletromiografia realizado, basta clicar sobre o botão "guardar registo atual", onde surge uma nova janela, em que é pedido ao utilizador que introduza um nome para ficar associado ao registo do sinal, Figura 6.7, nesta experiência o nome atribuído foi de "*testelog_1*".

Fonte: print screen do programa desenvolvido, elaborado pelo autor.

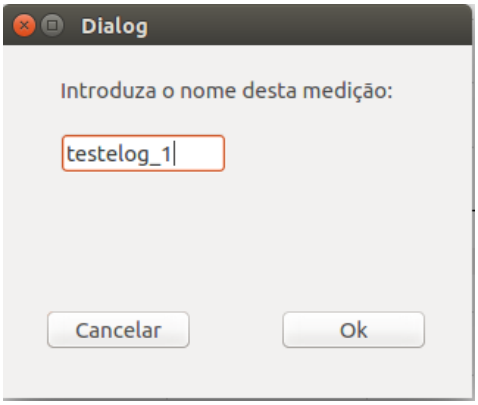


Figura 6.7: *Janela onde é pedido o nome para atribuir ao registo.*

Depois de introduzido o nome dado à experiência, é criada uma pasta com o nome atribuído pelo utilizador que contém um ficheiro de texto com os valores dos tempos e duas imagens, cada uma corresponde a um dos gráficos. Na Figura 6.8 é apresentada uma pasta exemplo, do registo efetuado.

Fonte: print screen da pasta do registo, elaborado pelo autor.

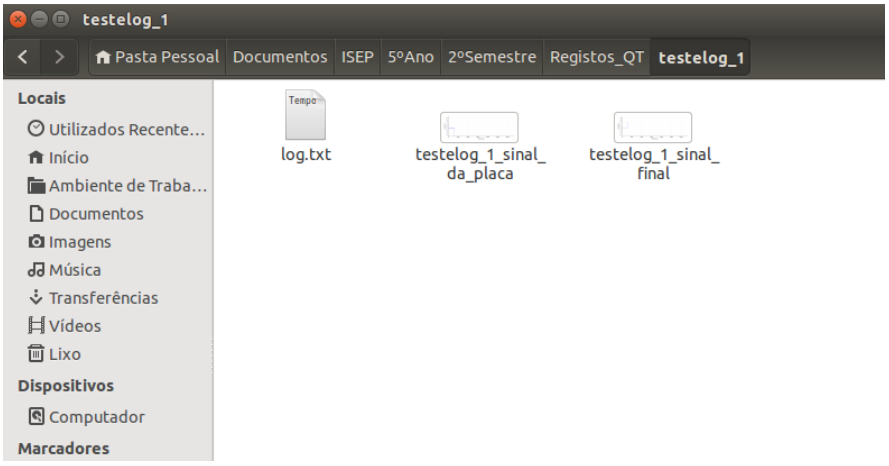


Figura 6.8: *Aspeto da pasta onde são guardados os registos.*

Na Figura 6.8 é visível as duas figuras dos gráficos, onde a imagem do sinal original é denominada de "*testelog_1_sinal_da_placa*"e da imagem com o sinal já tratado fica com o nome de "*testelog_sinal_final*". O ficheiro "*log.txt*"corresponde ao ficheiro de texto com os registos dos tempos e da tensão do sinal, em que pode ser visto um pequeno excerto na Figura 6.9, que corresponde a um teste efetuado no modo "tempo real"com o sistema ARM.

Fonte: print screen do ficheiro log, elaborado pelo autor.

Tempo (ms)	Tensão(V)
0	1.47266
32	1.48965
64	1.47549
96	1.48115
128	1.49248
160	1.47549
192	1.48398
224	1.47832
256	1.48115
288	1.47549
320	1.47832
352	1.47832
384	1.47832
416	1.47832
448	1.47549
480	1.47549
512	1.47549
544	1.47549
576	1.47549
608	1.47832
640	1.47549
672	1.47549
704	1.47549
736	1.48115
768	1.47832
800	1.47832
832	1.48115
864	1.47832

Figura 6.9: *Aspeto do ficheiro de texto onde são guardados os registos.*

A interface gráfica foi desenvolvida no sistema operativo *Ubuntu* 14.04, contudo é compatível com o *Windows* 10. Para que esta interface fosse compatível com o Windows 10, foi necessário remover todos os caracteres especiais que não são reconhecidos na passagem do programa do *linux* para o sistema operativo *Windows* 10. Na Figura 6.10 é apresentada a interface gráfica completa, semelhante à Figura 5.31, mas com os gráficos obtidos durante o teste com o sistema ARM no modo "tempo real".

Fonte: print screen do programa desenvolvido, elaborado pelo autor.

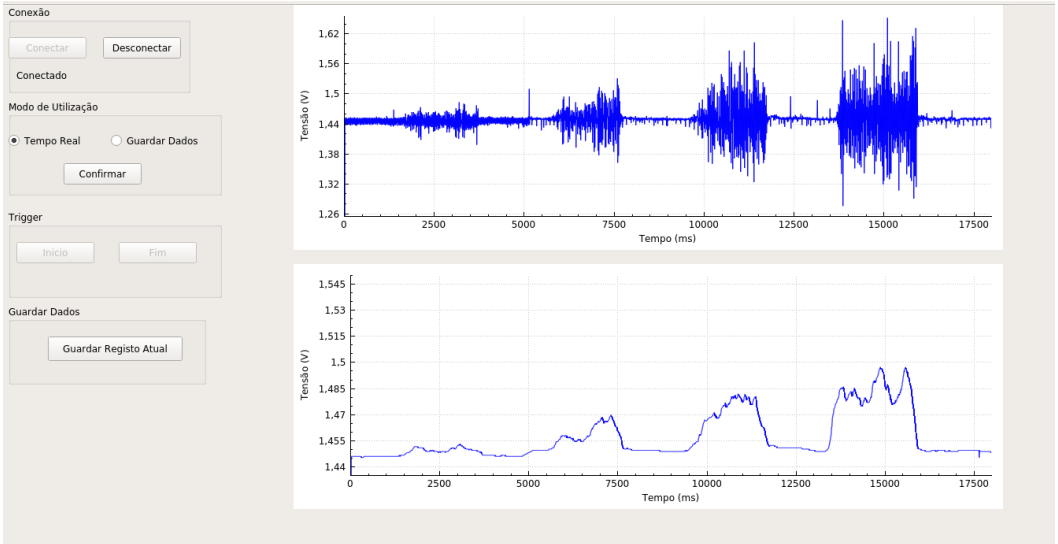


Figura 6.10: *Apresentação completa da interface gráfica.*

7
Conclusão

Durante o período da dissertação da tese de mestrado foi criado e testado um sistema de aquisição e registo de um sinal EMG. A montagem do circuito de aquisição do sinal foi baseada na pesquisa bibliográfica efetuada durante o estudo de outros produtos semelhantes [57][58][59]. Depois de o sinal estar condicionado, foi criado o código para os microcontroladores de forma a que conseguissem receber e enviar o sinal para a interface gráfica. A interface gráfica foi desenvolvida à imagem do *software* biopac, produto que possuiu um papel bastante relevante no decorrer deste projeto, pois foi a introdução ao género de sinal que era espectável obter.

Os resultados obtidos durante os testes efetuados permitiram perceber se o sistema desenvolvido era capaz de ler um sinal EMG com sucesso. Com base nas experiências realizadas com o sistema biopac, o sistema desenvolvido foi colocado nas mesmas condições de teste, com os mesmos elétrodos, no mesmo paciente e com o mesmo procedimento para realizar a leitura. Depois de comparados e analisados os resultados obtidos quer pelo sistema biopac e pelo sistema desenvolvido, é possível concluir que o projeto foi um sucesso. O sinal dos dois sistemas é bastante semelhante, cumprindo com o objetivo proposto inicialmente de realizar um sistema para a recolha de um sinal eletromiográfico.

Um dos obstáculos enfrentados, foi a capacidade de memória do microcontrolador ATmega32, que devido ao seu limite de memória, 2 kBytes não é suficiente

quando queremos registar um maior número de amostras, uma vez que não possui capacidade para as guardar em grande número, enquanto que no caso do microcontrolador ARM, não existiu esse problema. Relativamente à criação da interface gráfica, esta era uma área que não tinha muita experiência, uma vez que sai fora do âmbito do ensino providenciado pelo curso e onde todo o conhecimento foi adquirido durante a elaboração deste trabalho. A otimização dos tempos de amostragem, no modo tempo real, foi um factor que podia ter sido melhorado, caso houvesse um maior conhecimento prévio da manipulação dos dados recebidos pela porta série.

Como trabalho futuro seria interessante adicionar mais sensores ao sistema, tais como um acelerómetro e/ou uma balança de carga, de forma a ajudar o médico fisioterapeuta a conseguir extrair mais informações acerca do teste realizado, tornado assim a sua análise mais assertiva, uma vez que é baseada em mais fatores de medição.

Concluindo, este projeto foi considerado um sucesso, pois conseguiu atingir todas as metas que foram propostas para a elaboração do trabalho. O sinal EMG foi lido com sucesso, foram implementados os dois modos de operação e uma interface gráfica onde é possível visualizar o sinal. O desenvolvimento deste projeto foi bastante enriquecedor tanto a nível académico como profissional, em que foi possível aprender novos tópicos, em novas áreas onde a elétronica pode surgir para melhorar a qualidade de vida dos seres humanos. Foi uma honra ter feito parte deste projeto.

Bibliografia

[1] Academia, "Fisioterapia - Entorse do Tornozelo. Disponível em: *http* : *//www.academiadefutebol.pt/index.php/servicos_fisioterapia/servico/*14. Acedido em 23-03-2016.,", 2011.

[2] F. manual., "Entorse de tornozelo com diferentes níveis de gravidade. Disponível em: *http* : *//fisioterapiamanual.com.br/blog/artigos/entorse − de − tornozelo − diferentes − niveis − de − gravidade/*. Acedido em: 28-03-2106.,", 2016.

[3] A. D. da Silva, A. F. Mesquita, F. Gramaxo, M. E. Santos, and L. B. e José Mário Félix, in *Terra, Universo de Vida*, P. Editora, ed., (Bloco Gráfico, Lda, Porto, Portugal, 2007), No. 978-972-0-42170-8.

[4] S. Cagliano, in *O Corpo Humano*, Hiperlivro, ed., (Asa Editores II, S.A., Matosinhos, Portugal, 2002), No. 972-8735-41-3.

[5] O. L. B. R. V. M. Teresa, "Aptidão músculo esquelético. Disponível em: *http* : *//suavidaemforma.blogspot.pt/*2012/11/*aptidao − musculo − esqueletica.html*. Acedido em: 28-03-2106.,", 2016.

[6] C. da web., "Biologia histologia - Tecido Muscular. Disponível em: *http* : *//www.coladaweb.com/biologia/histologia/tecido − muscular*. Acedido em: 29-03-2106.,", 2016.

[7] E. Daniela., "Músculo cardíaco (Diapositivos). Disponível em: *http* : *//pt.slideshare.net/Danielaestremorpino/diapositivas − musculo − cardiaco*. Acedido em: 28-03-2106.,", 2016.

[8] A. de anatomia., "Sistema muscular - Músculos do membro inferior. Disponível em: *http* : *//www.auladeanatomia.com/novosite/sistemas/sistema − muscular/musculos − do − membro − inferior/musculos − do − pe/*. Acedido em: 28-03-2106.,", 2016.

[9] M. Eduardo., "O complexo articular do tornozelo. Disponível em: *http* : *//pt.slideshare.net/MauroEduardo/o − complexo − articular − do − tornozelo*. Acedido em: 28-03-2106.,", 2016.

[10] V. R. A. Wardand and J. L. e Ann Reed, in *Eletroterapia Explicada*, B. Heinemann, ed., (Elsevier Editora Ltda, Rio de Janeiro, Brasil, 2009), No. 978-85-352-3122-9.

[11] Benedetti, "M. Muscle activation intervals and EMG envelop in clinical gait analysis," IEEE Eng Med Biol Mag **20,** 4–33 (2001).

[12] A.-N. L. and G. N. e Sinkjaer T., "The influence os muscle length on muscle fibre condution velocity and development of muscle fatigue," Electroencephalogr Clin Neurophysiol **85,** 166–172 (1992).

[13] D. V., "Electromyographic biofeedback and recovery of quadriceps femoris muscle function following anterior cruciate ligament reconstruction," Phys Ther **70,** 11–17 (1990).

[14] N. RM and S. G. e Urbscheit NL, "Alteration of motor-unit discharge characteristics in aged humans," Phys Ther **64,** 29–34 (1984).

[15] Z. MJ and D. G. e Stegeman DF., "Recent progress in the diagnostic use of surface EMG for neurological diseases," J Electromyogr and Kinesiology **10,** 287–291 (2000).

[16] M. R, F. D, and G. M. e Schieroni MP., "Effect of age on muscle functions investigated with surface electromyography," Muscle and Nerve **25,** 65–76 (2002).

[17] C. R, R. A, and N. J. e Bellotti P, "Can continuous physical training counteract aging effect on myoelectric fatigue? A surface electromyography study application," Arch Phys Med Rehabil **84,** 513–517 (2003).

[18] B. HE and L. L. e Tesch PA, "Lower limb skeletal muscle function after 6 wk of bed rest," J Appl Physiol **82,** 182–188 (1997).

[19] Classroom., "Disponível em: $http://classroom.orange.com/pt/exploracoes-complementares.html$. Acedido em: 05-04-2106.,", 2016.

[20] Biomec., "Disponível em: $http://biomec.paginas.ufsc.br/$. Acedido em: 05-04-2106.,", 2016.

[21] Neurobase., "Elétrodo de agulha. Disponível em: $http://neurobase.com.br/loja/produtos/eletrodo-de-agulha-concentrica-ambu-neuroline-740.html$. Acedido em: 06-04-2106.,", 2016.

[22] Intechopen., "Disponível em: $http://www.intechopen.com/books/$. Acedido em: 06-04-2106.,", 2016.

[23] D. S. Day, "Important Factors in Surface EMG Measurement," Bortec Biomedical Incorporated .

[24] Intechopen., "Disponível em: $http://122155.us.all.biz/goods$. Acedido em: 06-04-2106.,", 2016.

[25] R. Merletti, G. Rau, and D. C. D.-K. e G.M. Hägg, "SENIAM. Disponível em: $http://seniam.org/sensorlocation.html$. Acedido em 24-03-2016.,", 2016.

[26] SENIAM., "Disponível em: *http* : *//seniam.org/gastrocnemiuslateralis.html*. Acedido em: 05-04-2106.,", 2016.

[27] ISEP, *Introdução aos amplificadores operacionais, electrónica II* (ISEP, 2015).

[28] NRSign., "Disponível em: *http* : *//www.nrsign.com/monopolar−vs−bipolar− emg − readings/*. Acedido em: 10-04-2106.,", 2016.

[29] A. M. S. F. Galhano, in *Modulacao de amplitude*, ISEP, ed., (Instituto Superior de Engenharia do Porto, 2007).

[30] A. M. S. F. Galhano, in *Introducao AM*, ISEP, ed., (Instituto Superior de Engenharia do Porto, 2007).

[31] A. M. S. F. Galhano, in *Ruido*, ISEP, ed., (Instituto Superior de Engenharia do Porto, 2001).

[32] Interfacebus., "Disponível em: *http* : *//www.interfacebus.com/opamp−high− pass − active − filter − design.html*. Acedido em: 10-04-2106.,", 2016.

[33] Chegg., "Disponível em: *http* : *//www.chegg.com/homework − help/questions − and − answers/rc − high − pass − filter − shown − figure − r − 155 − k − c − 00550 − f − 3 − db − frequency − circuit − frequency − ratio − −q*2428509. Acedido em: 11-04-2106.,", 2016.

[34] Interfacebus., "Disponível em: *http* : *//www.interfacebus.com/opamp − low − pass − active − filter − design.html*. Acedido em: 12-04-2106.,", 2016.

[35] Hyperphysics., "Disponível em: *http* : *//hyperphysics.phy − astr.gsu.edu/hbase/electric/filcap*2*.html*. Acedido em: 12-04-2106.,", 2016.

[36] Interfacebus., "Disponível em: *http* : *//www.interfacebus.com/opamp − band − pass − active − filter − design.html*. Acedido em: 13-04-2106.,", 2016.

[37] J. P. Baptista, in *MICROCONTROLADORES*, ISEP, ed., (Instituto Superior de Engenharia do Porto, 2007).

[38] M. garden., "Disponível em: *https* : *//microcontrollergarden.blogspot.pt*. Acedido em: 13-04-2106.,", 2016.

[39] RTOS, "Quality RTOS and Embedded Software. Disponível em: *http* : *//www.freertos.org/about − RTOS.html*. Acedido em 31-03-2016.,", 2011.

[40] A. Corporation, "Microcontroladores Atmel AVR de 8 bits e 32 bits. Disponível em: *http* : *//www.atmel.com/pt/br/products/microcontrollers/avr/default.aspx*. Acedido em 31-03-2016,", 2016.

[41] Atmel., "Disponível em: *http* : *//www.atmel.com/Images/doc*2503*.pdf*. Acedido em: 13-04-2106.,", 2016.

[42] N. M. S. Nogueira, Ph.D. thesis, Instituto Superior de Engenharia do Porto, 2009.

[43] Microchip., "Disponível em: *http* : *//www.microchip.com*. Acedido em: 13-04-2106.,", 2016.

[44] ISEP, *Slides, sistemas embebidos.* (ISEP, 2015).

[45] W. C. S. Co, "INTRODUCTION TO RS232 SERIAL COMMUNICATION. Disponível em: *https* : *//wcscnet.com/tutorials/introduction* − *to* − *rs*232 − *serial* − *communication/*. Acedido em: 01-04-2016.,", 2016.

[46] IEEE, in *Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*, L. S. Committee, ed., (IEEE Computer Society, 2012).

[47] H. health provider., "Disponível em: *http* : *//www.homehealthprovider.com/doc/font*−*colorbluemyotrac*−3*font*−0001. Acedido em: 16-04-2106.,", 2016.

[48] M. Plus, "Aparelho Electroterapia/Biofeedback e EMG Mod. 2U2P. Disponível em: *http* : *//www.medicalplus.pt/*. Acedido em: 18-04-2016.,", 2016.

[49] Medigalia., "Disponível em: *http* : *//www.medigalia.com/corrientes* − *y* − *terapia* − *combinada* − *c*102*x*2583943. Acedido em: 16-04-2106.,", 2016.

[50] U. do Porto, *Processamento de Sinais Fisiológicos* (Universidade do Porto, 2015).

[51] Datasheetdir., "Disponível em: *http* : *//www.datasheetdir.com/INA*128*P* + *Instrumentation* − *Amplifiers*. Acedido em: 26-04-2106.,", 2016.

[52] Intersil., "Disponível em: *http* : *//www.intersil.com*. Acedido em: 26-04-2106.,", 2016.

[53] IPP., "Disponível em: *http* : *//recipp.ipp.pt/handle/*10400.22/2682. Acedido em: 26-04-2106.,", 2016.

[54] M. electronics design., "Disponível em: *http* : *//masteringelectronicsdesign.com/the* − *transfer* − *function* − *of* − *the* − *summing* − *amplifier* − *with* − *n* − *input* − *signals/*. Acedido em: 27-04-2106.,", 2016.

[55] S. S. Rahman, "AT command mode of HC-05 and HC-06 Bluetooth module. Disponível em: *http* : *//www.instructables.com/id/AT* − *command* − *mode* − *of* − *HC* − 05 − *Bluetooth* − *module/*. Acedido em: 04-05-2106.,", 2014.

[56] INTECH, "Chapter 5. Influence of Different Strategies of Treatment Muscle Contraction and Relaxation Phases on EMG Signal Processing and Analysis During Cyclic Exercise. Disponível em: *http* : *//www.intechopen.com/books/computational* − *intelligence* − *in* −

electromyography − analysis − a − perspective − on − current − applications − and − future − challenges/influence − of − different − strategies − of − treatment − muscle − contraction − and − relaxation − phases − on − emg − signa. Acedido em: 04-04-2106.,", 2014.

[57] N. Nogueira, *Desenvolvimento de um sistema de avaliação da actividade muscular (Eletromiografia)* (Instituto Superior de Engenharia do Porto, Porto, Portugal, 2009).

[58] C. Mendes, *Desenvolvimento e validação de um sistema de recolha de sinal eletromiográfico baseado em placa de aquisição* (Instituto Superior de Engenharia do Porto, Porto, Portugal, 2012).

[59] M. Ramos, *Detetor de batimentos cardíaco* (Instituto Superior de Engenharia do Porto, Porto, Portugal, 2013).

Anexo A. Programação da ATmega32

Neste anexo são apresentados os diversos ficheiros do código desenvolvido para os microcontroladores e para a interface gráfica.

```
#define F_CPU 16000000UL

#include <avr/io.h>
#include <stdlib.h>
#include <math.h>
#include <stdio.h>
#include <avr/wdt.h>
#include <avr/interrupt.h>
#include <util/delay.h>
#include "USART.h"
#include "ADC.h"
#define sbi(a, b) (a) |= (1 << (b))
#define cbi(a, b) (a) &= ~(1 << (b))
volatile char flag_enviar=0, flag_fim=0, flag_guardar_dados=0,
flag_tempo_real=0,a='0';
char buf[6];
volatile uint16_t f=0, Valor_AD_gd[900],Valor_AD;
void inicio()
{
        //Timer
        //Timer - Tempo Real
        TCCR1A=0b00000000;
        TCCR1B=0b00001010;
        OCR1A=63999;
        //Interrupcao 16ms
        //Timer - Guardar dados
        TCCR2=0b00001100;
        OCR2=249;
        //USART
        usart_config(115200,16000000);
        //ADC
        adc_config(0b01000000,0b10000111);
        //Interrupcoes
        sei();
}
ISR(USART_RXC_vect)
```

```
{
        a=USART_Rececao();
        switch(a)
        {
                case('r'):
                        flag_tempo_real=1;
                        flag_guardar_dados=0;
                        break;
                case('g'):
                        flag_guardar_dados=1;
                        flag_tempo_real=0;
                        break;
                case('t'):
                        flag_enviar=1;
                        break;
                case('p'):
                        flag_enviar=0;
                        flag_fim=1;
                        break;
        }
}
ISR(TIMER1_COMPA_vect)
{
                Valor_AD=Ler_ADC(0);
                sprintf(buf,"%d\n",Valor_AD);
                USART_envia_string(buf);

}
ISR(TIMER2_COMP_vect)
{
        Valor_AD_gd[f]=Ler_ADC(0);
        f++;
}
int main()
{
        uint16_t i;
        char buffer[32];
        inicio();
        while(1)
        {
                if(flag_tempo_real==1)
                {
                        if(flag_enviar==1)
                        {
                                TIMSK=0b00010000;
```

```
                        }
                }
                if(flag_guardar_dados==1)
                {
                        if(flag_enviar==1)
                        {
                                TIMSK=0b10000000;
                        }
                }
                if(flag_fim==1)
                {
                        TIMSK=0b00000000;
                        if(flag_guardar_dados==1)
                        {
                                for(i=0;i<=f;i++)
                                {
                                sprintf(buffer,"%d\n",Valor_AD_gd[i]);
                                USART_envia_string(buffer);
                                }
                        }
                        USART_envia_string("00000\n");
                        flag_enviar=0;
                        flag_fim=0;
                        f=0;
                        flag_tempo_real=0;
                        flag_guardar_dados=0;
                }
        }
        return 0;
}
```

Anexo B. Programação da ARM

```
/* Standard includes. */
#include <stdio.h>
#include <string.h>
#include <stdint.h>
/* Scheduler includes. */
/* Library includes. */
#include "stm32f4xx.h"
#include "stm32f4xx_conf.h"
#include "tm_stm32f4_ili9341.h"
#include "stm32_eval_legacy.h"
/////////////////////////////////////////////////////////////////////////////
volatile char flag_fim=0, flag_enviar=0, flag_tempo_real=0,
flag_guardar_dados=0,a;
volatile uint16_t i=0,f=0,valor_ad_tr=0;
volatile uint16_t valor_ad[30000];
uint16_t ADC_Val; //Stores the calculated ADC value
/////////////////////////////////////////////////////////////////////////////
//Configuracao do LCD
void LCD_config()
{
  TM_ILI9341_Init();
  TM_ILI9341_Rotate(TM_ILI9341_Orientation_Portrait_2);
  TM_ILI9341_Fill(ILI9341_COLOR_RED);
}
void prvDisplayMessageLCD(int line_number, char *message )
{
        TM_ILI9341_Puts(12, 19*line_number, message, &TM_Font_11x18,
        ILI9341_COLOR_BLACK, ILI9341_COLOR_RED);
}

// Configuracao da freq. do relogio a usar //

/* Relogio de 180 Mhz */
void RCC_Config_HSE_PLL_MAX()
{
        uint32_t HSEStartUpStatus = 0;
        RCC_DeInit();
```

```
        RCC_HSEConfig(RCC_HSE_ON);
        HSEStartUpStatus = RCC_WaitForHSEStartUp();
        if (HSEStartUpStatus == SUCCESS)
        {
        //FLASH_SetLatency(FLASH_Latency_3);
        //FLASH_PrefetchBufferCmd(ENABLE);
        //FLASH_InstructionCacheCmd(ENABLE);
        //FLASH_DataCacheCmd(ENABLE);
        RCC_PLLConfig(RCC_PLLSource_HSE,8/*HSE_VALUE/1000000)*/,360,2,7);
        RCC_HCLKConfig(RCC_SYSCLK_Div1);
        RCC_PCLK2Config(RCC_HCLK_Div2);
        RCC_PCLK1Config(RCC_HCLK_Div4);
        RCC_PLLCmd(ENABLE);
        while (RCC_GetFlagStatus(RCC_FLAG_PLLRDY) == RESET);
        RCC_SYSCLKConfig(RCC_SYSCLKSource_PLLCLK);
        while (RCC_GetSYSCLKSource() != RCC_CFGR_SWS_PLL);

                        }
        else
        {
                        return;
        }
}
////////////////////////////////////////////////////////////////////////
// Grupo de interrupcoes //
// Configuracao das interrupcoes //
void interrupcoes_config()
{
    NVIC_InitTypeDef NVIC_InitStructure;
    NVIC_PriorityGroupConfig(NVIC_PriorityGroup_1);
    //Interrupcao do timer4
        NVIC_InitStructure.NVIC_IRQChannel = TIM4_IRQn;
    NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority = 0;
    NVIC_InitStructure.NVIC_IRQChannelSubPriority = 0;
    NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;
    NVIC_Init(&NVIC_InitStructure);
    //Interrupcao do timer5
    NVIC_InitStructure.NVIC_IRQChannel = TIM5_IRQn;
    NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority = 0;
    NVIC_InitStructure.NVIC_IRQChannelSubPriority = 0;
    NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;
    NVIC_Init(&NVIC_InitStructure);
        //Interrupcao da USART
        NVIC_InitStructure.NVIC_IRQChannel = USART1_IRQn;
        NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority = 0;
```

```
        NVIC_InitStructure.NVIC_IRQChannelSubPriority = 1;
        NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;
        NVIC_Init(&NVIC_InitStructure);
}
////////////////////////////////////////////////////////////////////////////
// Configuracao dos perifericos //
// Configuracao da USART1 //
void USART1_config(uint32_t baudrate)                    //Funcao da
configuracao da USART que recebe como parametros de entrada o
baudrate e define o tamanho da fila de mensagens
{
        USART_InitTypeDef USART_InitStructure;
        GPIO_InitTypeDef GPIO_InitStructure;
        RCC_AHB1PeriphClockCmd(RCC_AHB1Periph_GPIOA, ENABLE);
        RCC_APB2PeriphClockCmd(RCC_APB2Periph_USART1, ENABLE);
        GPIO_PinAFConfig(GPIOA, GPIO_PinSource9, GPIO_AF_USART1);
        GPIO_PinAFConfig(GPIOA, GPIO_PinSource10, GPIO_AF_USART1);
        GPIO_InitStructure.GPIO_Pin = GPIO_Pin_9|GPIO_Pin_10;
        GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AF;
        GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
        GPIO_InitStructure.GPIO_OType = GPIO_OType_PP;
        GPIO_InitStructure.GPIO_PuPd = GPIO_PuPd_UP;
        GPIO_Init(GPIOA, &GPIO_InitStructure);
        USART_InitStructure.USART_BaudRate = baudrate;
        USART_InitStructure.USART_WordLength = USART_WordLength_8b;
        USART_InitStructure.USART_StopBits = USART_StopBits_1;
        USART_InitStructure.USART_Parity = USART_Parity_No;
        USART_InitStructure.USART_HardwareFlowControl =
        USART_HardwareFlowControl_None;
        USART_InitStructure.USART_Mode = USART_Mode_Rx | USART_Mode_Tx;
        USART_Init(USART1, &USART_InitStructure);
        /* Ativa USART1 */
        USART_Cmd(USART1, ENABLE);
        USART_ITConfig(USART1, USART_IT_RXNE, ENABLE);
}
// Configuracao do ADC1 //
void ADC1_config(uint8_t channel, uint32_t prescaler)
{
        if(prescaler==2) prescaler=ADC_Prescaler_Div2;
        if(prescaler==4) prescaler=ADC_Prescaler_Div4;
        if(prescaler==6) prescaler=ADC_Prescaler_Div6;
        if(prescaler==8) prescaler=ADC_Prescaler_Div8;
        RCC_AHB1PeriphClockCmd(RCC_AHB1Periph_GPIOC, ENABLE);
        RCC_APB2PeriphClockCmd(RCC_APB2Periph_ADC1, ENABLE);
        GPIO_InitTypeDef GPIO_InitStructure;
```

```
        /* ADC Channel 11 -> PC1 */
        GPIO_InitStructure.GPIO_Pin = GPIO_Pin_1;
        GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AN;
        GPIO_InitStructure.GPIO_PuPd = GPIO_PuPd_NOPULL ;
        GPIO_Init(GPIOC, &GPIO_InitStructure);
        ADC_CommonInitTypeDef ADC_CommonInitStructure;
        ADC_InitTypeDef ADC_InitStructure;
        /* ADC Common Init */
        ADC_CommonInitStructure.ADC_Mode = ADC_Mode_Independent;
        ADC_CommonInitStructure.ADC_Prescaler = prescaler;
        ADC_CommonInitStructure.ADC_DMAAccessMode =
        ADC_DMAAccessMode_Disabled;
        ADC_CommonInitStructure.ADC_TwoSamplingDelay =
        ADC_TwoSamplingDelay_5Cycles;
        ADC_CommonInit(&ADC_CommonInitStructure);
        ADC_InitStructure.ADC_Resolution = ADC_Resolution_12b;
        ADC_InitStructure.ADC_ScanConvMode = DISABLE; // 1 Channel
        ADC_InitStructure.ADC_ContinuousConvMode = DISABLE;
        ADC_InitStructure.ADC_ExternalTrigConvEdge =
        ADC_ExternalTrigConvEdge_None;
        ADC_InitStructure.ADC_ExternalTrigConv =
        ADC_ExternalTrigConv_T2_TRGO;
        ADC_InitStructure.ADC_DataAlign = ADC_DataAlign_Right;
        ADC_InitStructure.ADC_NbrOfConversion = 1;
        ADC_Init(ADC1, &ADC_InitStructure);
        /* ADC1 regular channel 11 configuration */
        ADC_RegularChannelConfig(ADC1, ADC_Channel_11, 1,
        ADC_SampleTime_144Cycles); // PC1
        /* Enable ADC1 */
        ADC_Cmd(ADC1, ENABLE);
}
void timer4_config()
{
        /* Enable TIM4 clock */
        RCC_APB1PeriphClockCmd(RCC_APB1Periph_TIM4, ENABLE);
        /* Time base configuration */
    TIM_TimeBaseInitTypeDef TIM_TimeBaseStructure;
        TIM_TimeBaseStructure.TIM_Period = 1000;
        TIM_TimeBaseStructure.TIM_Prescaler = 89;
        TIM_TimeBaseStructure.TIM_ClockDivision = 0;
        TIM_TimeBaseStructure.TIM_CounterMode = TIM_CounterMode_Up;
        TIM_TimeBaseInit(TIM4, &TIM_TimeBaseStructure);
        TIM_ARRPreloadConfig( TIM4, ENABLE );
}
void timer5_config()
```

```
{
        /* Enable TIM5 clock */
        RCC_APB1PeriphClockCmd(RCC_APB1Periph_TIM5, ENABLE);
        /* Time base configuration */
    TIM_TimeBaseInitTypeDef TIM_TimeBaseStructure;
    TIM_TimeBaseStructure.TIM_Period = 1000;
    TIM_TimeBaseStructure.TIM_Prescaler = 2879;
        TIM_TimeBaseStructure.TIM_ClockDivision = 0;
        TIM_TimeBaseStructure.TIM_CounterMode = TIM_CounterMode_Up;
        TIM_TimeBaseInit(TIM5, &TIM_TimeBaseStructure);
        TIM_ARRPreloadConfig( TIM5, ENABLE );
}
////////////////////////////////////////////////////////////////////////
// Funcoes caracteristicas da USART //
// Funcao para receber os dados do tipo char, atraves da interrupcao de
rececao da USART //
char USART_obter_Char()
{
}
// Funcao para enviar os dados do tipo char, atraves da interrupcao de
envio da USART //
void USART_enviar_char(char put_char)
{
        USART_SendData(USART1, put_char);

        /* Loop until the end of transmission */
        while (USART_GetFlagStatus(USART1, USART_FLAG_TC) == RESET)
        {}
}
// Funcao para enviar os dados do tipo char formando uma string, atraves
da interrupcao de envio da USART //
void USART_enviar_string(char *put_string)
{
        int i =0;

                        while (put_string[i] != 0x00)
                        {
                                USART_enviar_char(put_string[i]);
                                i++;
                        }

}
////////////////////////////////////////////////////////////////////////
// Funcoes do ADC1 //
uint16_t Ler_ADC( )
```

```
{
        int ADC1ConvertedValue=0;
        ADC_SoftwareStartConv(ADC1);
        while( ADC_GetFlagStatus(ADC1, ADC_FLAG_EOC) != SET );
        ADC1ConvertedValue = ADC_GetConversionValue(ADC1);
        ADC_ClearFlag(ADC1, ADC_FLAG_EOC);
        return ADC_GetConversionValue(ADC1);
}
////////////////////////////////////////////////////////////////////////////
// Funcoes das interupcoes //
void TIM4_IRQHandler()
{
        if(TIM_GetFlagStatus(TIM4,TIM_FLAG_Update) != RESET)
        {
                valor_ad[f]=Ler_ADC();
                f++;
                TIM_ClearFlag(TIM4, TIM_FLAG_Update);
        }
}
void TIM5_IRQHandler()
{
        char buf_tr[32];
        if(TIM_GetFlagStatus(TIM5,TIM_FLAG_Update) != RESET)
        {
                valor_ad_tr=Ler_ADC();
                sprintf(buf_tr,"%d\n",valor_ad_tr);
                USART_enviar_string(buf_tr);
                f++;
                TIM_ClearFlag(TIM5, TIM_FLAG_Update);
        }
}

void USART1_IRQHandler(void)
{
        if(USART_GetITStatus(USART1, USART_IT_RXNE) != RESET)
        {
                        USART_ClearITPendingBit(USART1,USART_IT_RXNE);
                        a = USART_ReceiveData(USART1);
//O valor recebido da USART passa para a varivel a
                        switch(a)
                        {
                                        case('r'):
                                                flag_tempo_real=1;
                                                flag_guardar_dados=0;
                                                break;
```

```
                                        case('g'):
                                                flag_guardar_dados=1;
                                                flag_tempo_real=0;
                                                break;
                                        case('t'):
                                                flag_enviar=1;
                                                break;
                                        case('p'):
                                                flag_fim=1;
                                                flag_enviar=0;
                                                break;
                                }
                                a=0;
        }
        USART_ClearITPendingBit(USART1, USART_IT_RXNE);
}

////////////////////////////////////////////////////////////////////////////////

int main(void)
{
        uint16_t j;
        char buf[32];
        // Configuracao do relogio //
        RCC_Config_HSE_PLL_MAX();

        // Inicializacao do Hardware //
        LCD_config();
        config_Leds();
        USART1_config(115200);
        config_Botao();
        ADC1_config(ADC_Channel_0,4);
        timer4_config();
        timer5_config();
        lcd_info();
        interrupcoes_config();
        //IntExt_Config();
        while (1)
        {
                if(flag_tempo_real==1)
                {
                        if(flag_enviar==1)
                        {
                                TIM_Cmd(TIM5, ENABLE);
                                TIM_ITConfig(TIM5, TIM_IT_Update, ENABLE );
```

```
                        }
                }
                if(flag_guardar_dados==1)
                {
                        if(flag_enviar==1)
                        {
                                TIM_Cmd(TIM4, ENABLE);
                                TIM_ITConfig(TIM4, TIM_IT_Update, ENABLE );
                        }
                }
                if(flag_fim==1)
                {
                        TIM_Cmd(TIM4, DISABLE);
                        TIM_ITConfig(TIM4, TIM_IT_Update, DISABLE );
                        TIM_Cmd(TIM5, DISABLE);
                        TIM_ITConfig(TIM5, TIM_IT_Update, DISABLE );
                        if(flag_guardar_dados==1)
                        {
                                for (j=0;j<=f;j++)
                                {
                                        sprintf(buf,"%d\n",valor_ad[j]);
                                        USART_enviar_string(buf);
                                }
                        }

                    USART_enviar_string("00000\n");
                    flag_fim=0;
                    flag_enviar=0;
                    f=0;
                    flag_tempo_real=0;
                    flag_guardar_dados=0;
                    a=0;
                 }
        }
}
```

Anexo C. Programação do QT Creator

mainwindow.h

```
#ifndef MAINWINDOW_H
#define MAINWINDOW_H
#include <QMainWindow>
#include "segunda_janela.h"
#include "erro_abrir_ficheiro.h"
#include <QThread>
#include <QTimer>
namespace Ui {
class MainWindow;
}
class MainWindow : public QMainWindow
{
    Q_OBJECT
public:
    explicit MainWindow(QWidget *parent = 0);
    ~MainWindow();
private slots:
    void readData();
    void makePlot();
    void on_pushButton_conexao_ligar_clicked();
    void on_pushButton_conexao_desligar_clicked();
    //void realtimeDataSlot();
    void grafico_smoothing();
    void on_pushButton_confirmarmodo_clicked();
    void on_pushButton_trigger_clicked();
    void on_pushButton_trigger_fim_clicked();
    void on_radioButton_temporeal_clicked();
    void on_radioButton_guardardados_clicked();
    void on_pushButton_guardar_registo_clicked();
    void guardar_dados();
    void grafico_acelerometro();
private:
    Ui::MainWindow *ui;
    QTimer *timer;
    QString serialBuffer,nome,valor_EMG_Buffer,Buffer_valor_EMG,
    buffer_tempo,buffer_total;
```

```
    QByteArray va,tempo_gd;
    QVector<int> valor_EMG,valor_EMG_rectificado,valor_EMG_smooth,tempo
    ,valor_EMG_log;
    QStringList BufferSplit;
    quint16 flag_tempo_real,num_amostras,guardar_
    tam_BufferSplit,tam_BufferSplit_antigo,valor_sinal,reg_tempo,
    valor_smooth;
    quint16 cnt,valor_eixo_x,num_amostras_x_inicial,reg_tempo_guardados;
    float valor_sinal_float;
    QVector<float> valor_sinal_float_vetor,valor_EMG_log2;
    int f=0;
    quint32 tam_BufferSplit, tam_serialBuffer;
};
class Sleeper : public QThread
{
public:
    static void usleep(unsigned long usecs){QThread::usleep(usecs);}
    static void msleep(unsigned long msecs){QThread::msleep(msecs);}
    static void sleep(unsigned long secs){QThread::sleep(secs);}
};
#endif // MAINWINDOW_H
#if QT_DEPRECATED_SINCE(5, 0)
QT_DEPRECATED static inline void setGraphicsSystem(const QString &&#41; {}
#endif
```

mainwindow.cpp

```
#include <stdlib.h>     //for using the function sleep
#include <stdio.h>
#include <time.h>
#include <unistd.h>
#include <math.h>
#include "mainwindow.h"
#include "ui_mainwindow.h"
#include <QSerialPort>
#include <QDebug>
#include <sstream>
#include <iostream>
#include <fstream>
#include <string>
#include <iomanip>
#include <QMessageBox>
QSerialPort *serial;
MainWindow::MainWindow(QWidget *parent) :
    QMainWindow(parent),
    ui(new Ui::MainWindow)
{
```

```
    ui->setupUi(this);
    serialBuffer="";
    tempo.resize(55500);
    valor_EMG_log.resize(55500);
    valor_EMG.resize(55500);
    valor_EMG_rectificado.resize(55500);
    valor_EMG_smooth.resize(55500);
    valor_EMG_log2.resize(55500);
    valor_sinal_float_vetor.resize(55500);
    reg_tempo=0;
    tempo[0]=0;
    valor_EMG_log[0]=0;
    //valor_EMG[0]=0;
    //valor_EMG_rectificado[0]=0;
    valor_EMG_smooth[0]=0;
    valor_EMG_log2[0]=0;
    valor_sinal_float_vetor[0]=0;
    reg_tempo_guardados=0;
    serial = new QSerialPort(this);
    serial->setPortName("rfcomm1");
    serial->setBaudRate(QSerialPort::Baud115200);
    serial->setDataBits(QSerialPort::Data8);
    serial->setParity(QSerialPort::NoParity);
    serial->setStopBits(QSerialPort::OneStop);
    serial->setFlowControl(QSerialPort::NoFlowControl);
    connect(serial,SIGNAL(readyRead()),this,SLOT(readData()));
}
MainWindow::~MainWindow()
{
    delete ui;
    serial->close();
}
void MainWindow::readData()
{
    va = serial->readAll();
    serialBuffer += QString::fromStdString(va.toStdString());
    Buffer_valor_EMG += QString::fromStdString(va.toStdString());
    if(flag_tempo_real==1)
    {
        if(serialBuffer[serialBuffer.size()-1]== '\n')
        {
            BufferSplit = serialBuffer.split("\n");
            valor_sinal=BufferSplit[0].toInt();
            serialBuffer="";
            if(valor_sinal!=00000)
```

```
            {
                reg_tempo +=32;
                MainWindow::makePlot();
            }
        }
    }
    if(flag_tempo_real==0)
    {
        tam_serialBuffer=serialBuffer.size();
        /* Condicao que verifica se o envio de dados chegou ao fim */
        if( (tam_serialBuffer>7) && (serialBuffer[tam_serialBuffer-6]=='0')
        {
            BufferSplit = serialBuffer.split("\n");
// Dividir o buffer entre os '/n'
            tam_BufferSplit=BufferSplit.size();
            reg_tempo_guardados=0;
            for(int l=0; l<tam_BufferSplit-3; ++l)
            {
                tempo[l] = reg_tempo_guardados;
// Amostras de 1 ms
                reg_tempo_guardados=reg_tempo_guardados+1;

                valor_EMG[l]=BufferSplit[l].toInt();
// Passar de string para int
                valor_sinal_float_vetor[l]=valor_EMG[l]*2.88/4095;
//AVR=valor_sinal_float_vetor[l]=valor_EMG[l]*2.88/1023;

            }
            MainWindow::makePlot();
            serialBuffer="";
        }
    }
}
void MainWindow::makePlot()
{
    if(flag_tempo_real==1)
    {
        if(reg_tempo<20000)
        {
            num_amostras=20000;
            num_amostras_x_inicial=0;
        }else
        {
            //ui->customPlot->graph(0)->removeDataBefore(reg_tempo-20000);
            //ui->customPlot3->graph(0)->removeDataBefore(reg_tempo-20000);
```

```
            num_amostras=reg_tempo;
            num_amostras_x_inicial=reg_tempo-20000;
        }
        valor_sinal_float=valor_sinal*2.88/4095;
        // create graph and assign data to it:
        ui->customPlot->addGraph();
        // give the axes some labels:
        ui->customPlot->xAxis->setLabel("Tempo (ms)");
        ui->customPlot->yAxis->setLabel("Tensao (V)");
        // set axes ranges, so we see all data:
        ui->customPlot->setInteraction(QCP::iRangeDrag, true);
        ui->customPlot->setInteraction(QCP::iRangeZoom, true);
        ui->customPlot->graph(0)->addData(reg_tempo, valor_sinal_float);
        //ui->customPlot->graph(0)->addData(reg_tempo, valor_sinal);
        ui->customPlot->xAxis->setRange(num_amostras_x_inicial,num_amostras)
        ui->customPlot->yAxis->setRange(1.255,1.655);
        //ui->customPlot->yAxis->setRange(1800, 2200);
        ui>customPlot->replot();
        guardar_tam_BufferSplit=tam_BufferSplit;
        MainWindow::grafico_smoothing();
    }
    if(flag_tempo_real==0)
    {
        // create graph and assign data to it:
        ui->customPlot->addGraph();
        ui->customPlot->graph(0)->setData(tempo, valor_sinal_float_vetor );
        // give the axes some labels:
        ui->customPlot->xAxis->setLabel("Tempo (ms)");
        ui->customPlot->yAxis->setLabel("Tensao (V)");
        // set axes ranges, so we see all data:
        ui->customPlot->xAxis->setRange(0,reg_tempo_guardados);
        ui->customPlot->yAxis->setRange(1.255,1.655);
        ui->customPlot->setInteraction(QCP::iRangeDrag, true);
        ui->customPlot->setInteraction(QCP::iRangeZoom, true);
        ui->customPlot->replot();
        guardar_tam_BufferSplit=tam_BufferSplit;
        tam_BufferSplit=0;
        valor_EMG_rectificado=valor_EMG;
        MainWindow::grafico_smoothing();
    }
}
/* Etapa do smoothing do grafico - nesta etapa sao eliminados os valores de
 pico tornando o grafico visualmente mais agradavel
Basicamente esta funcao usa certos n de pontos a volta do ponto de
referencia, soma-os e calcula a media.*/
```

```
void MainWindow::grafico_smoothing()
{
    if(flag_tempo_real==1)
    {
        if(valor_sinal<=2069)
//AVR=555,ARM=2069
        {
            valor_sinal=(2069-valor_sinal)+2069;
        }
        valor_EMG[cnt]=valor_sinal;
        int periodo_smoothing=12;
//Numero de amostras para realizar o smoothing
        if(cnt>=periodo_smoothing)
        {
           for(int j=0;j<periodo_smoothing;j++)
           {
                valor_sinal += valor_EMG[cnt-j];
           }
           valor_sinal/=(periodo_smoothing+1);
           valor_sinal_float=valor_sinal*2.88/4095;
        }
        else
        {
            for(int j=0;j<cnt;j++)
            {
                 valor_sinal += valor_EMG[cnt-j];
            }
            valor_sinal/=(cnt+1);
            valor_sinal_float=valor_sinal*2.88/4095;
//valor_sinal_float=valor_sinal*2.88/1023;
        }
        // create graph and assign data to it:
        ui->customPlot3->addGraph();
        ui->customPlot3->graph(0)->addData(reg_tempo, valor_sinal_float );
        // give the axes some labels:
        ui->customPlot3->xAxis->setLabel("Tempo (ms)");
        ui->customPlot3->yAxis->setLabel("Tensao (V)");
        // set axes ranges, so we see all data:
        ui->customPlot3->xAxis->setRange(num_amostras_x_inicial,num_amostras
        ui->customPlot3->yAxis->setRange(1.435,1.55);
        ui->customPlot3->setInteraction(QCP::iRangeDrag, true);
        ui->customPlot3->setInteraction(QCP::iRangeZoom, true);
        ui->customPlot3->replot();
        valor_EMG_log2[cnt]=valor_sinal_float;
        cnt += 1;
```

```
    }
    if(flag_tempo_real==0)
    {
        for (int i=0; i<(guardar_tam_BufferSplit-3); ++i)
        {
            if(valor_EMG_rectificado[i]<=2050)
//AVR=550,ARM=2069
            {
                valor_EMG_rectificado[i]=(2050-valor_EMG_rectificado[i])
                +2050;
            }
        }
        valor_EMG_smooth=valor_EMG_rectificado;
        int periodo_smoothing=350;
//Numero de amostras para realizar o smoothing
        for(int i=0;i<guardar_tam_BufferSplit;i++)
        {
            if(i+periodo_smoothing+1<guardar_tam_BufferSplit)
            {
                for(int j=1;j<periodo_smoothing;j++)
                {
                    valor_EMG_smooth[i] += valor_EMG_smooth[i+j];
                }
                valor_EMG_smooth[i]/=periodo_smoothing;
            }
            else
            {
                for(int j=1;j<periodo_smoothing;j++)
                {
                    valor_EMG_smooth[i] += valor_EMG_smooth[i-j];
                }
                valor_EMG_smooth[i]/=periodo_smoothing;
            }
            valor_sinal_float_vetor[i]=valor_EMG_smooth[i]*2.88/4095;
        }
       // create graph and assign data to it:
       ui->customPlot3->addGraph();
       ui->customPlot3->graph(0)->setData(tempo, valor_sinal_float_vetor );
       // give the axes some labels:
       ui->customPlot3->xAxis->setLabel("Tempo (ms)");
       ui->customPlot3->yAxis->setLabel("Tensao (V)");
       // set axes ranges, so we see all data:
       ui->customPlot3->xAxis->setRange(0,reg_tempo_guardados);
       //ui->customPlot3->yAxis->setRange(1.13, 1.98);
       ui->customPlot3->yAxis->setRange(1.435,1.55);
```

```
        ui->customPlot3->setInteraction(QCP::iRangeDrag, true);
        ui->customPlot3->setInteraction(QCP::iRangeZoom, true);
         ui->customPlot3->replot();
    }
}
/*************************************************************************
/******************************** Funcoes da interface grafica **********
/*************************************************************************
void MainWindow::on_pushButton_conexao_ligar_clicked()
{
    ui->pushButton_conexao_ligar->setCursor(QCursor(Qt::WaitCursor));
    serial->open(QIODevice::ReadWrite);
    sleep(4);
    if ( serial->isOpen() && serial->isWritable() && serial->isReadable() )
    {
        ui->label_conexao->setText("Conectado");
        ui->label_espera_conexao->setEnabled(false)
        ui->pushButton_conexao_desligar->setEnabled(true);
        ui->pushButton_conexao_ligar->setEnabled(false);
        ui->radioButton_guardardados->setEnabled(true);
        ui->radioButton_temporeal->setEnabled(true);
    }
    else
    {
        QMessageBox::critical(this, tr("Error"), serial->errorString());
    }
    ui->pushButton_conexao_ligar->setCursor(QCursor(Qt::ArrowCursor));
}
void MainWindow::on_pushButton_conexao_desligar_clicked()
{
    serial->close();
    ui->label_conexao->setText("Desconectado");
    ui->pushButton_conexao_desligar->setEnabled(false);
    ui->pushButton_conexao_ligar->setEnabled(true);
    ui->pushButton_confirmarmodo->setEnabled(false);
    ui->pushButton_trigger->setEnabled(false);
    ui->radioButton_guardardados->setEnabled(false);
    ui->radioButton_temporeal->setEnabled(false);
    ui->pushButton_guardar_registo->setEnabled(false);
}
void MainWindow::on_radioButton_temporeal_clicked()
{
    ui->pushButton_confirmarmodo->setEnabled(true);
    flag_tempo_real=1;
}
```

```
void MainWindow::on_radioButton_guardardados_clicked()
{
    ui->pushButton_confirmarmodo->setEnabled(true);
    flag_tempo_real=0;
}
void MainWindow::on_pushButton_confirmarmodo_clicked()
{
    ui->pushButton_trigger->setEnabled(true);
    if(flag_tempo_real==0)
    {
        serial->write("g");
        valor_eixo_x=2000;
    }
    if(flag_tempo_real==1)
    {
        serial->write("r");
        valor_eixo_x=20000;
    }
    ui->customPlot->clearGraphs();
    ui->customPlot3->clearGraphs();
    reg_tempo=0;
    reg_tempo_guardados=0;
    tempo.clear();
    tempo.resize(55500);
    valor_EMG_log.clear();
    valor_EMG_log.resize(55500);
    valor_EMG.clear();
    valor_EMG.resize(55500);
    valor_EMG_rectificado.clear();
    valor_EMG_rectificado.resize(55500);
    valor_EMG_smooth.clear();
    valor_EMG_smooth.resize(55500);
    valor_EMG_log2.clear();
    valor_EMG_log2.resize(55500);
    valor_sinal_float_vetor.clear();
    valor_sinal_float_vetor.resize(55500);
    cnt=0;
    tam_BufferSplit=0;
    num_amostras=0;
    ui->pushButton_guardar_registo->setEnabled(false);
}
void MainWindow::on_pushButton_trigger_clicked()
{
    serial->write("t");
    ui->pushButton_trigger->setEnabled(false);
```

```
    ui->pushButton_trigger_fim->setEnabled(true);
    ui->radioButton_guardardados->setEnabled(false);
    ui->radioButton_temporeal->setEnabled(false);
    ui->pushButton_confirmarmodo->setEnabled(false);
}
void MainWindow::on_pushButton_trigger_fim_clicked()
{
    serial->write("p");
    ui->pushButton_trigger->setEnabled(false);
    ui->pushButton_trigger_fim->setEnabled(false);
    ui->radioButton_guardardados->setEnabled(true);
    ui->radioButton_temporeal->setEnabled(true);
    ui->pushButton_confirmarmodo->setEnabled(true);
    ui->pushButton_guardar_registo->setEnabled(true);
}
void MainWindow::on_pushButton_guardar_registo_clicked()
{
    Segunda_janela janela_guardar_registo;
    janela_guardar_registo.setModal(true);
    janela_guardar_registo.exec();
    nome = janela_guardar_registo.getTextOfInput1();
    MainWindow::guardar_dados();
}
void MainWindow::guardar_dados()
{
    //Escolher o diretoria para o qual serao gravados os dados
    QString outputDir = "/home/flavio/Documentos/ISEP/5Ano/2Semestre/
    Registos_QT";
    QString fileName = nome;
    QDir().mkdir(outputDir+"/"+fileName);
    //Gravar imagens
    ui->customPlot->savePng(outputDir+"/"+fileName+"/"+fileName+"
    _sinal_da_placa",
     0, 0, 1.0, -1);
    ui->customPlot3->savePng(outputDir+"/"+fileName+"/"+fileName+"
    _sinal_final",
     0, 0, 1.0, -1);
    //Guardar log
    buffer_total="Tempo (ms)          Tensao(V)\n"
    /* Log Tempo Real*/
    if(flag_tempo_real==1)
    {
        for(int i=0; i < (cnt); i++)
        {
            tempo[i]=32*i;
```

```
        }
        for (int i = 0; i < (cnt); i++)
        {
            buffer_total+="      ";
            buffer_total+=QString::number(tempo[i]);
            buffer_total+="                  ";
            buffer_total+=QString::number(valor_EMG_log2[i]);
            buffer_total+="\n";
        }
        QFile file( outputDir+"/"+fileName+"/"+"log.txt");
        if ( file.open(QIODevice::WriteOnly) )
        {
             QTextStream stream( &file );
             stream << buffer_total << endl;

        }
        else
        {
            erro_abrir_ficheiro erro_ficheiro;
            erro_ficheiro.setModal(true);
            erro_ficheiro.exec();
        }
    }
    /* Log Guardar Dados */
    if(flag_tempo_real==0)
    {
        valor_EMG_log=valor_EMG_smooth;
        for (int i = 0; i < (guardar_tam_BufferSplit); i++)
        {
                        buffer_total+="      ";
            buffer_total+=QString::number(tempo[i]);
            buffer_total+="                  ";
            buffer_total+=QString::number(valor_sinal_float_vetor[i]);
            buffer_total+="\n";
        }
        QFile file( outputDir+"/"+fileName+"/"+"log.txt");
        if ( file.open(QIODevice::WriteOnly) )
        {
             QTextStream stream( &file );
             stream << buffer_total << endl;
        }
        else
        {
            erro_abrir_ficheiro erro_ficheiro;
            erro_ficheiro.setModal(true);
```

```
                erro_ficheiro.exec();
            }
        }
}
```