

POLITÉCNICO DO PORTO
INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO

Eletrocardiograma (ECG)

Rafael Silva
Manuel Lemos

Mestrado em Engenharia Electrotécnica e de Computadores
Área de Especialização em Automação e Sistemas

ISEP INSTITUTO SUPERIOR
DE ENGENHARIA DO PORTO

DEPARTAMENTO DE ENGENHARIA ELETROTÉCNICA
Instituto Superior de Engenharia do Porto

Fevereiro, 2025

Este relatório satisfaz, parcialmente, os requisitos que constam da Ficha de Unidade Curricular de LAMEC, do 2º ano, do Mestrado em Engenharia Electrotécnica e de Computadores, Área de Especialização em Automação e Sistemas.

Candidato: Rafael Silva, Nº 1200946, 1200946@isep.ipp.pt

Candidato: Manuel Lemos, Nº 1220597, 1220597@isep.ipp.pt

Orientação Científica: António Meireles, aem@isep.ipp.pt

Coorientação Científica: André Fidalgo, anf@isep.ipp.pt



DEPARTAMENTO DE ENGENHARIA ELETROTÉCNICA

Instituto Superior de Engenharia do Porto

Rua Dr. António Bernardino de Almeida, 431, 4200-072 Porto

Fevereiro, 2025

Resumo

O desenvolvimento de sistemas para monitorização da saúde tem vindo a ganhar cada vez mais importância, especialmente no que diz respeito às doenças cardiovasculares. Este trabalho explora o uso de Inteligência Artificial (AI) para a análise de Eletrocardiograma (ECG), com o objetivo de detetar precocemente condições cardíacas. Utilizando a plataforma STM32 e o sensor SEN-12650, foi criado um dispositivo capaz de recolher sinais de ECG, que são processados em tempo real através de algoritmos de AI implementados em *Python*. A combinação destes componentes permite desenvolver um sistema eficaz e de baixo custo para a monitorização remota de pacientes, contribuindo para a prevenção de doenças cardiovasculares. Este trabalho demonstra o potencial das tecnologias utilizadas e de *machine learning* no melhoramento da saúde pública.

Palavras-Chave: Eletrocardiograma, ECG, AI, Python, STM32, SEN-12650, Doenças Cardiovasculares

Abstract

The development of health monitoring systems has become increasingly important, especially in the field of cardiovascular diseases. This work explores the use of AI for the analysis of ECG, aiming to detect cardiac conditions at an early stage. Using the STM32 platform and the SEN-12650 sensor, a device was developed to acquire ECG signals, which are processed in real-time through AI algorithms implemented in Python. The combination of these components enables the creation of an efficient, low-cost system for remote patient monitoring, contributing to the prevention of cardiovascular diseases. This work demonstrates the potential of embedded technologies and machine learning in improving public health.

Keywords: Electrocardiogram, ECG, AI, Python, STM32, SEN-12650, Cardiovascular Diseases

Índice

Lista de Figuras	ix
Lista de Tabelas	xiii
Listagens	xvi
Lista de Acrónimos	xvii
1 Introdução	1
1.1 Contextualização	2
1.2 Definição do Problema	3
1.2.1 Objetivos	3
1.2.2 Resultados esperados	3
1.3 Plano de Trabalho	3
1.4 Organização da Dissertação	4
2 Estado de Arte	7
2.1 ECG	7
2.2 Processos Fisiológicos	13
2.2.1 O processo celular	14
2.2.2 Complexo QRS	16
2.2.3 Princípios Físicos da ECG	17
2.2.4 Triângulo de Einthoven	19
2.3 Características elétricas de um sinal de ECG.	22
2.3.1 Elétrodos	24
2.4 Sistemas de aquisição	24
2.4.1 Rejeição de modo comum	28
2.4.2 Blindagem ativa dos cabos de entrada.	29
2.4.3 Filtros e Conversores de Sinais	29
Conversor Analógico-Digital	31
Filtragem analógica	33
O Filtro Passa-Alto	33
O Filtro Passa-Baixo	34
Filtros Passa-Banda	35

Filtros <i>Notch</i> (50/60 Hz)	35
2.5 Segurança elétrica	36
2.5.1 Normas de segurança	37
2.6 Doenças cardiovasculares	37
2.6.1 Arritmias	37
2.6.2 Distúrbios de Condução	38
2.6.3 Hipertrofia	39
2.6.4 Alterações de ST/T	40
2.6.5 ECG 200+	40
2.6.6 Projeto <i>Do It Yourself</i> (DIY)	42
3 Arquitetura do Projeto	45
3.1 Descrição geral da arquitetura do projeto	45
3.2 Projeto e descrição do <i>Hardware</i>	47
3.2.1 Núcleo STM32F767ZI	47
3.2.2 Sen-12650	49
AD8232	51
3.2.3 Esquema elétrico	54
3.3 Projeto e descrição do <i>Software</i>	55
3.3.1 Núcleo STM32F767ZI	55
3.3.2 Aplicação em <i>python</i>	56
3.3.3 Relação entre núcleo e aplicação	58
3.3.4 Inteligência Artificial	58
3.4 Procedimentos e vetores de Teste	60
3.5 Lista de Componentes e custo de implementação	61
4 Protótipo funcional	63
4.1 Programação do STM32	63
4.1.1 <i>Universal Synchronous Asynchronous Receiver-Transmitter</i> (USART)	65
4.2 Filtros digitais	66
4.2.1 Biblioteca CMSIS-DSP	67
Filtro <i>Finite Impulse Response</i> (FIR)	67
Filtro <i>Infinite Impulse Response</i> (IIR)	68
4.2.2 Filtro FIR	71
Passa-Baixo	71
Passa-Alto	74
Média Móvel	78
4.2.3 Filtro IIR	80
<i>Notch</i>	80
4.2.4 Considerações Finais	82
4.3 Inteligência artificial	83

4.4	Interface gráfica	99
4.5	Testes e resultados	106
4.5.1	Aplicação dos Filtros no STM22	108
4.5.2	Aplicação da AI	111
5	Conclusão	117
5.1	Problemas Encontrados	118
5.2	Desenvolvimentos Futuros	119
	Referências	121
	Anexo A Esquemático Analógico	125
	Anexo B Esquemático Digital	127
	Anexo C Circuito integrado do SEN-12650	129

Lista de Figuras

1.1	Demografia em 1 de janeiro de 2023 [2]	2
1.2	Gráfico de gantt	4
2.1	Eletrómetro capilar de Lippmann [6]	9
2.2	Experiencia de Waller [6]	10
2.3	Ondas eletrofisiológicas do coração [6]	10
2.4	Sistema de Derivações Eletrocardiográficas [6]	11
2.5	Coração [8]	13
2.6	O potencial de ação de uma célula do miocárdio ventricular [8] . . .	15
2.7	Potencial de ação de uma célula marca-passo (nódulo sinusal) [8] . .	15
2.8	Esquema do sistema especializado de excitação e condução do coração que controla as contrações cardíacas [8]	16
2.9	Ondas características [8]	16
2.10	Fluxo de bioeletricidade em circuitos de corrente apresentando quatro segmentos: uma travessia para o exterior da membrana celular, um segmento extracelular, uma travessia para o interior da membrana celular e um segmento intracelular [8]	18
2.11	Eletrocardiograma (ECG) registado a partir das três derivações eletrocardiográficas padrão dos membros: paciente diagnosticado com enfarte do miocárdio [8]	20
2.12	Esquema para a colocação de elétrodos com o objetivo de registrar as derivações precordiais: V1, V2, V3, V4, V5 e V6 [8]	21
2.13	(A) Esquema do circuito aumentado da derivação unipolar dos membros, neste exemplo avR, obtido pela diferença entre o potencial de um único elétrodo R no braço direito e o GT. (B) Representação do tronco com o ECG de 12 derivações [8]	22
2.14	Gama de frequências [9]	23
2.15	Tipos de elétrodos	24
2.16	Componentes do biopotencial medido [8]	25
2.17	Principais componentes de um condicionador de sinal ECG [8] . . .	26
2.18	Amplificador diferencial com seguidores de tensão para aumento da impedância de entrada	26
2.19	Amplificador de instrumentação clássico (pré-amplificação)	27

2.20	Amplificador de instrumentação clássico (pré-amplificação) [8]	29
2.21	O espectro de potência típico do sinal de ECG inclui os seus sub-componentes e artefactos comuns (ruído de movimento e muscular). A potência das ondas P e T (P-T) é de baixa frequência, e o complexo QRS está concentrado na gama de frequências médias, embora a potência residual esta estende-se até 100 Hz [10]	31
2.22	Sinal com um Conversor Analógico-Digital (ADC) de menor resolução [9]	32
2.23	Sinal com um ADC de maior resolução [9]	33
2.24	Filtragem analógica com filtros passivos de 1 ^a ordem e 2 ^a fase de amplificação com amplificador não-inversor [8]	34
2.25	Exemplo de um filtro ativo passa-banda [9]	35
2.26	Esquema de um filtro notch [9]	36
2.27	Taquicardia Auricular Multifocal [8]	38
2.28	Bloqueio Atrioventricular (AV) tipo III [8]	39
2.29	Hipertrofia Arterial [11]	39
2.30	Supradesnívelamento de ST [12]	40
2.31	ECG 200+ [13]	41
2.32	Arquitetura do projeto diy	43
3.1	Componentes do potencial bioelétrico medido.[8]	45
3.2	Arquitetura de software	46
3.3	Diagrama de blocos	46
3.4	Núcleo F767ZI	47
3.5	SEN-12650	49
3.6	AD8232	51
3.7	<i>Pinout</i> AD8232 [17]	52
3.8	Implementação de filtros passa-alto e passa-baixo no AD8232	53
3.9	Esquema Elétrico	55
3.10	Fluxograma STM32F767ZI	56
3.11	Fluxograma aplicação	57
3.12	<i>Mockup</i> da aplicação	57
3.13	Interligação do núcleo e aplicação	58
3.14	Diferentes subáreas da inteligência artificial	59
4.1	Diagrama de blocos filtro <i>Finite Impulse Response</i> (FIR)	67
4.2	Diagrama de Blocos Filtro <i>Infinite Impulse Response</i> (IIR)	69
4.3	Escala dos coeficientes	70
4.4	Resposta ao filtro FIR passa-baixo	73
4.5	Resposta ao filtro FIR passa-alto	76
4.6	Filtro média móvel aplicado	78

4.7	Filtro média móvel aplicado	78
4.8	Resposta ao filtro IIR <i>notch</i>	81
4.9	Resposta aos vários filtros	82
4.10	Arquitetura da Inteligência Artificial (AI)	91
4.11	Interface gráfica	106
4.12	Montagem do Prototipo	107
4.13	Sinal de saída do Sen-12650	108
4.14	Filtro FIR passa alto	108
4.15	Filtro FIR passa baixo	109
4.16	Filtro FIR média móvel	109
4.17	Filtro IIR <i>Notch</i>	110
4.18	Resultado Final	110
4.19	ECG realizado em ambiente hospitalar	111
4.20	Gráfico de precisão de treino	112
4.21	Gráfico da relação de perdas por época	113
4.22	Matriz de Confusão	114
4.23	Confirmação da receção de email	115
A.1	Implementação analógica	126
B.1	Implementação digital	128
C.1	Círcuito integrado do SEN-12650	130

Lista de Tabelas

3.1	Especificações do Nucleo-F767ZI [16]	49
3.2	Especificações do SEN-12650 (Monitor de Frequência Cardíaca AD8232)	51
3.3	Especificações do Chip AD8232 [18]	54
3.4	Tabela de componentes e valores	61
4.1	Correspondência entre classes numéricas e anotações de doenças cardíacas	95
4.2	Métricas de desempenho do Modelo <i>Long Short-Term Memory</i> (LSTM)	112

Listagens

4.1	Configuração do <i>timer</i>	64
4.2	Configuração do ADC	64
4.3	Configuração do <i>Universal Synchronous Asynchronous Receiver-Transmitter</i> (USART)	66
4.4	Filtro FIR passa-baixo	71
4.5	Filtro FIR passa-baixo no STM32	73
4.6	Filtro FIR passa-alto	74
4.7	Filtro FIR passa-baixo no STM32	76
4.8	Filtro Média Móvel	78
4.9	Filtro Média Móvel	79
4.10	Filtro IIR <i>notch</i>	80
4.11	Filtro IIR <i>notch</i> no STM32	82
4.12	Importação de bibliotecas	83
4.13	Diretório do <i>dataset</i>	84
4.14	Carregamento do ECG	85
4.15	Carregamento das anotações do ECG	86
4.16	Definir os tempos entre cada onda r-r e associar a um problema	87
4.17	Imprimir o gráfico de dispersão	88
4.18	Contagem dos ECG	89
4.19	Divisão dos dados	89
4.20	Balanceamento dos dados	90
4.21	Modelo de AI	94
4.22	Dicionário com descrição das classes	95
4.23	Treino do modelo	97
4.24	Método de verificação do input email	99
4.25	Método para calcular os BPM	100
4.26	Método para calcular os BPM por média	100
4.27	Método para obter as portas do computador	100
4.28	Método para ativar a inteligencia artificial e desativar	101
4.29	Método para ativar a linha do raw	101
4.30	Método para iniciar o ECG	101
4.31	Método para parar o ECG	102
4.32	Método para detetar pico r	102

4.33	Método para processar o intervalo r-r	103
4.34	Método atualizar o gráfico e a interface global	103
4.35	Método para enviar email	105
4.36	Verificação do Diagnóstico	114

Lista de Acrónimos

AAMI	<i>Association for the Advancement of Medical Instrumentation</i>
AC	<i>Alternating Current</i>
ADC	Conversor Analógico-Digital
AI	Inteligência Artificial
ARM	<i>Acorn RISC Machine</i>
ART	<i>Adaptive Real-Time</i>
ATP	Trifosfato de Adenosina
AV	Atrioventricular
CAN	<i>Controller Area Network</i>
CMR	<i>Common Mode Rejection</i>
CMRR	<i>Common Mode Rejection Ratio</i>
CNN	<i>Convolutional Neural Networks</i>
CPU	<i>Central Processing Unit</i>
DAC	<i>Digital-to-Analog Converter</i>
DC	<i>Direct Corrent</i>
DCV	Doenças Cardiovasculares
DICOM	<i>Digital Imaging and Communications in Medicine</i>
DIY	<i>Do It Yourself</i>
DMA	<i>Direct Memory Access</i>
ECG	Eletrocardiograma
ECGAR	Eletrocardiograma de Alta Resolução
EMG	Eletromiografia

EN	<i>European Norm</i>
FIR	<i>Finite Impulse Response</i>
FPU	<i>Floating Point Unit</i>
GND	<i>Ground</i>
GPIO	<i>General Purpose Input/Output</i>
GPU	Unidade de Processamento Gráfico
HL7	<i>Health Level 7</i>
I2C	<i>Inter-Integrated Circuit</i>
IAR	<i>Interactive Application Runtime</i>
IDE	<i>Integrated Development Environment</i>
IEC	Comissão Eletrotécnica Internacional
IEEE	<i>Institute of Electrical and Electronics Engineers</i>
IIR	<i>Infinite Impulse Response</i>
IoT	<i>Internet of Things</i>
IP	<i>Ingress Protection</i>
ISO	<i>International Organization for Standardization</i>
LA	<i>Left arm</i>
LAN	<i>Local Area Network</i>
LCD	<i>Liquid-Crystal Display</i>
LCD-TFT	<i>Liquid Crystal Display - Thin-Film Transistor</i>
LED	<i>Light Emitting Diode</i>
LSTM	<i>Long Short-Term Memory</i>
MAC	Multiplicação e Acumulação
MGF	Medicina Geral e Familiar
OTG	<i>On-The-Go</i>
PVC	Contrações Ventriculares Prematuras

RA	<i>Right arm</i>
RC	<i>Resistor–Capacitor</i>
ReLU	<i>Rectified Linear Unit</i>
RL	<i>Right leg</i>
RLD	Amplificador de Perna Direita
RNG	<i>Random Number Generator</i>
RNN	Redes Neurais Recorrentes
RTC	<i>Real-Time Clock</i>
RX	Radiografia
SCP-PDF	<i>Standard Communications Protocol for PDF</i>
SMOTE	<i>Synthetic Minority Over-sampling Technique</i>
SNR	Relação Sinal-Ruído
SPI	<i>Serial Peripheral Interface</i>
SRAM	<i>Static Random-Access Memory</i>
STEMI	<i>ST Elevation Myocardial Infarction</i>
TPU	Unidade de Processamento Tensor
UART	<i>Universal Asynchronous Receiver/Transmitter</i>
USART	<i>Universal Synchronous Asynchronous Receiver-Transmitter</i>
USB	<i>Universal Serial Bus</i>
WiFi	<i>Wireless Fidelity</i>
XGA	<i>Extended Graphics Array</i>
XML-GDT	<i>Extensible Markup Language - Device Data Format</i>

Capítulo 1

Introdução

Cada vez mais está evidente que vivemos numa sociedade “acelerada”, onde o ritmo das transformações gerou uma disruptão do binómio espaço-tempo cujo equilíbrio era suposto preservar a coesão e o progresso. Estamos sobre a pressão de prazos impraticáveis, onde todos nós já ouvimos a expressão “tempo é dinheiro”, onde há uma impulsão para ser produtivo a toda a hora, mostrando que qualquer atividade deve ser capitalizada, fazendo com que deixemos de viver a vida e passemos a produzi-la. Trazendo um novo significado para o tempo, sendo este uma medida de produção e não uma medida abstrata de orientação no dia-a-dia. Passamos a um regime 24 horas sobre 7 dias, onde deixamos o descanso ser severamente afetado, sendo que este deveria ser visto como um processo natural inerte ao ser humano e é cada vez mais visto, como tempo improdutivo.

Esta forma de viver, traz malefícios à saúde, como por exemplo, a síndrome de *burnout*, ou stress crónico, onde este conceito tem vindo a sofrer alterações ao longo do tempo, mas categoricamente, é associado a uma resposta prolongada a situações onde está implícito o stress físico e emocional crónico, que culminam em exaustão e sentimento de ineficácia.

Podemos também referir que a depressão e a ansiedade estão implicitamente conectadas com a falência do sono e com a impulsão crescente para nos produzirmos a nós próprios enquanto indivíduos. A sociedade deixa de se preocupar com as suas necessidades fisiológicas, como por exemplo a alimentação, onde é evidente um consumo excessivo de *fast-food*, comida pré-feita, onde todos os seus componentes apresentam um risco elevado à saúde. As doenças cardíacas, são resultantes

desse consumo excessivo, não menosprezando a genética, tendo um índice de morte elevado mundialmente, com uma percentagem de 75.39% em 2021 na união europeia [1].

O envelhecimento da população tem vindo a aumentar, como é possível observar na figura 1.1, e como as doenças cardiovasculares, afetam as pessoas mais idosas cada vez são mais necessários métodos de diagnósticos prematuros dessas doenças, de modo a conseguir salvar uma grande percentagem de vidas [2].

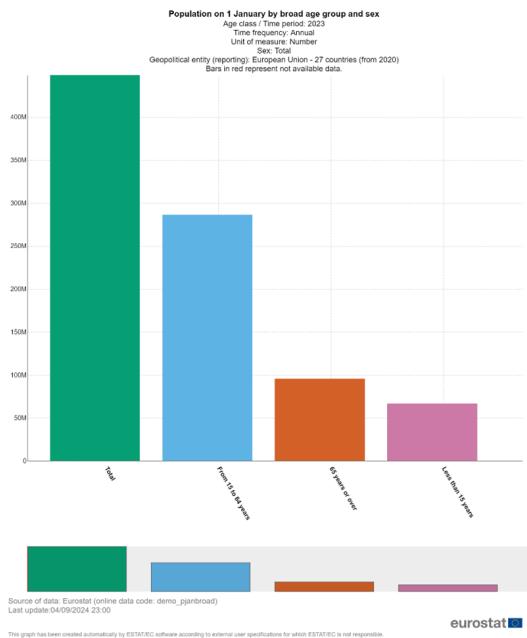


Figura 1.1: Demografia em 1 de janeiro de 2023 [2]

1.1 Contextualização

O ECG é um exame que permite diagnosticar Doenças Cardiovasculares (DCV), permitindo registar a atividade elétrica do coração, que se designa de eletrofisiologia do coração, de forma indolor, não invasiva, e de simples execução, apesar de necessitar de técnicos especializados, pois os elétrodos têm que ser colocados nas regiões corretas. Todas estas características serão abordadas posteriormente.

O ECG é considerado um exame importantíssimo e preliminar tanto em Medicina Geral e Familiar (MGF), como numa unidade de urgências. O mesmo é aplicado em pacientes com dor no peito, tonturas, vertigens, desmaios, palpitações cardíacas, pulsação acelerada ou irregular, falta de ar, fraqueza e cansaço.

Para além disto, com o avançar da idade, e/ou se existir histórico familiar de doenças cardíacas, o médico poderá prescrever uma ECG como forma de rastreio. É

também considerado um exame essencial numa rotina pré-operatória, como análises ao sangue e a Radiografia (RX) ao tórax [3].

1.2 Definição do Problema

Como referido na introdução, as doenças cardiovasculares estão cada vez mais a aumentar, e uma das formas de travar esse crescimento contínuo é um diagnóstico prematuro das doenças, para melhor tratamento dos pacientes. Com isto introduzimos a ECG, um exame indolor, e que permite através da sua interpretação identificar alterações do miocárdio como lesão, isquemia ou enfarte, aumento das câmaras auriculares ou ventriculares, perturbações do ritmo cardíaco, e também alterações intracardíacas como doenças metabólicas, alterações eletrolíticas, efeitos tóxicos ou terapêuticos de fármacos e outras [4].

1.2.1 Objetivos

Este trabalho tem como principal objetivo o estudo e implementação de um ECG, neste caso um protótipo portátil, onde será necessário entender quais os componentes eletrónicos necessários para a mesma e a melhor maneira de representar o sinal eletrofisiológico do coração, com o acréscimo da criação de uma interface de monitorização de anomalias, onde poderá avisar uma pessoa próxima do doente. Como isto dependerá de uma fonte elétrica, poderão surgir ruídos, onde terão de ser estudados os melhores filtros a utilizar, de modo a eliminá-los e perceber todos os sinais adquiridos do ECG. Além do desenvolvimento de uma aplicação que permita a visualização gráfica da onda do ECG e a deteção de possíveis doenças, com o auxílio de inteligência artificial.

1.2.2 Resultados esperados

Com este trabalho espera-se um protótipo de um ECG funcional com aquisição de biossinais fidedignos. Do mesmo modo, a criação de uma aplicação web, de monitorização de dados adquiridos pelo ECG, de modo a detetar problemas de saúde referentes a doenças cardiovasculares, enviando alertas caso seja detetado algo incomum. Perante isso deverá ser possível perceber como o ECG funciona a nível eletrónico, conseguindo assim aprimorar o conhecimento nesta mesma área.

1.3 Plano de Trabalho

Para suporte da calendarização e planeamento da atividades previstas para este projeto, foi utilizado o Microsoft Project, onde foi gerado um diagrama de gantt onde se calendarizaram as macro tarefas definidas para este projeto.

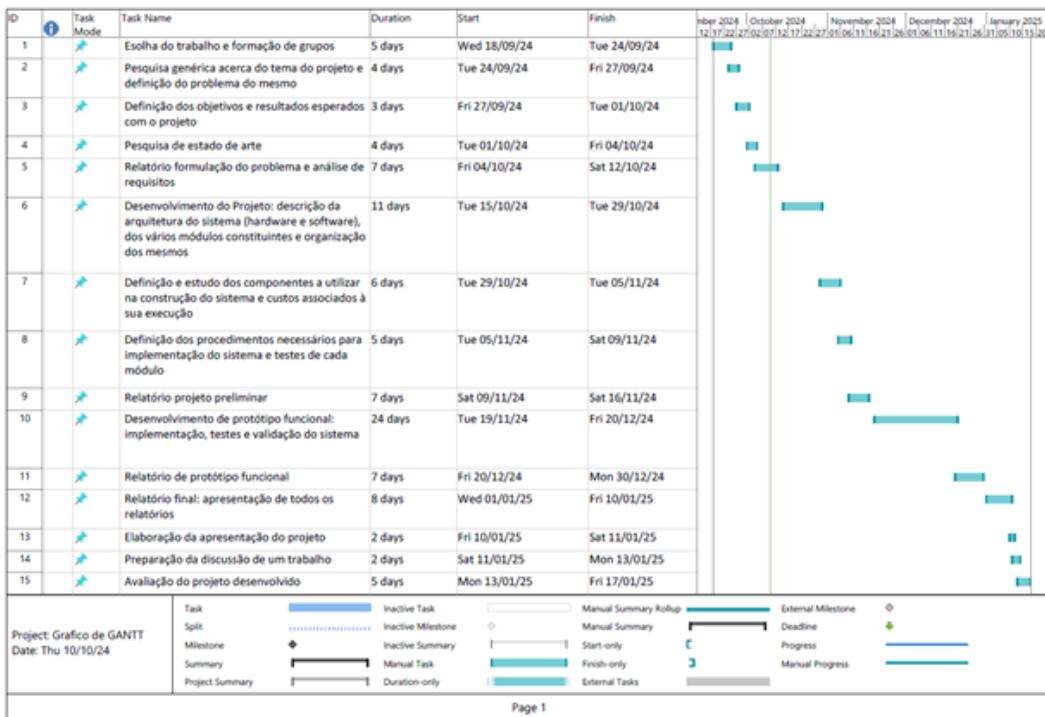


Figura 1.2: Gráfico de gantt

1.4 Organização da Dissertação

Nesta secção do trabalho será apresentada a distribuição de cada capítulo nomeadamente, descrever a finalidade de cada um deles.

O Capítulo 1, denominado "Introdução", contextualiza o tema do projeto, apresentando a motivação que levou à sua realização e os principais objetivos definidos. Este capítulo também delimita o escopo do trabalho, esclarecendo quais aspectos serão abordados, e finaliza com uma visão geral da estrutura do documento.

O Capítulo 2, intitulado "Estado da Arte", consiste numa revisão detalhada da literatura e das tecnologias relacionadas ao tema do projeto. São explorados os trabalhos mais relevantes e os avanços recentes na área, destacando as contribuições existentes.

O Capítulo 3, designado "Arquitetura do Projeto", descreve em detalhe o design e a estrutura do projeto. Inclui a apresentação de diagramas de blocos e fluxogramas, para ilustrar a arquitetura de forma clara e comprehensível. Além disso, são explicadas as escolhas tecnológicas e metodológicas feitas, justificando-as com base nos requisitos do projeto.

No Capítulo 4, "Protótipo Funcional", é apresentado o desenvolvimento do protótipo, destacando as principais funcionalidades implementadas. Este capítulo pode

incluir capturas de ecrã, trechos de código ou outras representações visuais para ilustrar o funcionamento do protótipo. Também são discutidos os desafios enfrentados ao longo do desenvolvimento e as soluções encontradas.

Por fim, o Capítulo 5, denominado "Conclusão", resume os principais resultados alcançados e as contribuições do trabalho. Este capítulo reforça a importância do projeto, analisando o impacto das soluções apresentadas, e sugere possíveis direções futuras para melhorias ou expansão do trabalho desenvolvido.

Capítulo 2

Estado de Arte

O coração é um órgão vital cuja fisiologia é sustentada por processos celulares altamente especializados que garantem a geração e a condução dos impulsos elétricos necessários para o funcionamento coordenado do sistema cardiovascular. A análise dessa atividade elétrica é realizada através do ECG, uma ferramenta essencial para a compreensão da dinâmica cardíaca.

Neste capítulo, será abordada a fisiologia do coração, incluindo os mecanismos celulares que sustentam a sua função, com destaque para os processos de despolarização e repolarização responsáveis pelas ondas características do traçado do ECG. Serão detalhadas as diferentes fases da onda eletrocardiográfica, como a onda P, o complexo QRS e a onda T, e a sua relação com o ciclo cardíaco.

Adicionalmente, será explorada a estrutura do hardware de um sistema de ECG, incluindo os seus componentes principais e os filtros eletrónicos utilizados para eliminar interferências e melhorar a qualidade do sinal. Será ainda apresentada a relevância dos critérios de segurança implementados nestes sistemas, garantindo a proteção dos pacientes durante a realização do exame.

2.1 ECG

A evolução do estudo da irritabilidade e sensibilidade no corpo humano, desde o século XVII até ao surgimento da ECG, reflete um percurso de descobertas fundamentais sobre a interação entre estímulos elétricos e a fisiologia muscular e nervosa. Em meados do século XVII, Francis Glisson observou a reação das fibras musculares

quando estimuladas, um fenômeno que denominou de irritabilidade. Esta descoberta tornou-se o alicerce da teoria que desenvolveu sobre as funções do corpo. Glisson defendia que a sensibilidade e a irritabilidade estavam profundamente interligadas no organismo, de tal modo que uma servia sempre como sinal para ativar a outra.

Albrecht von Haller negava que a sensibilidade e a irritabilidade estivessem ligadas, surgindo em 1752, com o conceito Halleriano, onde presume que irritabilidade era a capacidade das fibras musculares se contraírem quando estimuladas e a sensibilidade, por outro lado, dependia dos nervos e consistia na sensação de dor em resposta aos estímulos.

Luigi Galvani, perante o conceito Halleriano começou a investigar, com o intuito de perceber se a fonte dos movimentos musculares poderia ser atribuída à eletricidade. Utilizou a metade inferior do corpo de uma rã, neste caso as pernas, com nervos expostos e um fio metálico inserido através do canal vertebral, onde após várias experiência concluiu que as contrações não tinham relação com os eventos atmosféricos e que a eletricidade era intrínseca e presente no animal. Percebeu que condutores externos causavam contrações pelo fluxo desta eletricidade interna, ou seja, a eletricidade ficava acumulada principalmente nos músculos. As fibras musculares correspondiam a minúsculas garrafas de Leyden, ou seja, condensadores primitivos usados para armazenar carga elétrica, onde as fibras nervosas penetravam o interior das mesmas e possibilitava um fluxo de eletricidade para o exterior [5].

Alessandro Volta admirado pelas experiências de Galvani decidiu conectar dois pontos do mesmo nervo, sem qualquer contacto com o músculo, obtendo o mesmo resultado que Galvani. Com isto, Galvani decidiu imergir num recipiente a perna de uma rã e noutro um fragmento de um nervo ciático, concluindo que quando os dois recipientes se aproximavam as pernas se contraíam.

Em 1842, foi demonstrado que uma corrente elétrica acompanhava uma contração do coração, por Carlo Matteucci. No ano seguinte, Emil DuBois-Reymond, considerado fundador da eletrofisiologia descreveu o “potencial de ação” e confirmou a descoberta acima mencionada.

O primeiro “potencial de ação” cardíaco foi registado em 1856 pelos fisiologistas Rudolph Von Koelliker e Heinrich Muller, provando assim que uma corrente elétrica acompanhava cada contração do coração.

No início da década de 1870, o físico francês Gabriel Lippmann, representado na Figura 2.1, inventou o eletrômetro capilar, marcando assim o início da ECG.

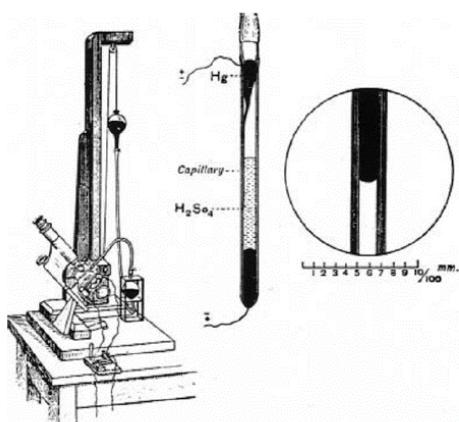


Figura 2.1: Eletrómetro capilar de Lippmann [6]

Estes avanços possibilitaram, em 1878, a descoberta de que cada contração cardíaca era acompanhada por uma variação elétrica, consistindo em "duas fases, a saber, uma perturbação inicial de curta duração, na qual o ápice se torna positivo, e uma segunda fase muito mais longa, na qual o ápice tende à negatividade". Esta foi a primeira descrição da despolarização e repolarização ventricular, pelos fisiologistas britânicos John Burdon Sanderson e Frederick Page.

Seis anos mais tarde, Burdon Sanderson e Page publicaram vários traçados da atividade elétrica do coração registados com um eletrómetro capilar. As ondações que registaram foram posteriormente denominadas complexo QRS e onda T. O primeiro ECG humano realizado foi em 1877, por Augustus D. Waller, com o eletrómetro capilar. Ele conectou elétrodos no torax anterior e posterior, e demonstrou que cada batimento cardíaco era acompanhado por uma oscilação elétrica. Com isto, provou que a atividade elétrica precedia a contração cardíaca, o que descartava a possibilidade de os registos serem artefactos provocados pela alteração do contacto entre os elétrodos e a pele durante os impulsos cardíacos.

Mais tarde, Waller observou que era desnecessário aplicar os elétrodos no peito humano, e de que se podia registrar os potenciais elétricos a partir dos membros submersos em soluções salinas. Esta experiência está retratada na Figura 2.2.

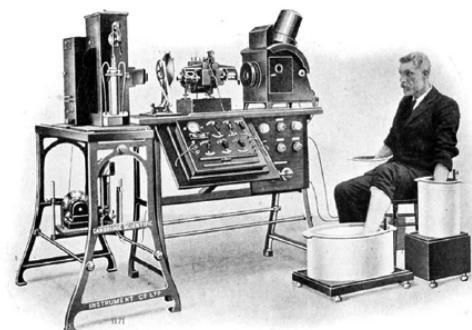


Figura 2.2: Experiencia de Waller [6]

Einthoven, a partir de recursos matemáticos, conseguiu um traçado mais próximo da realidade percebendo assim a limitação de frequência que o eletrómetro capilar continha.

Com isto, efetuou otimizações no mesmo, onde conclui que cada contração era acompanhada por 5 deflexões elétricas distintas, as quais rotulou com P, Q, R, S e T. Estas deflexões podem ser visualizadas na Figura 2.3. A escolha destas letras não foi arbitrária, refletia uma tradição na matemática.

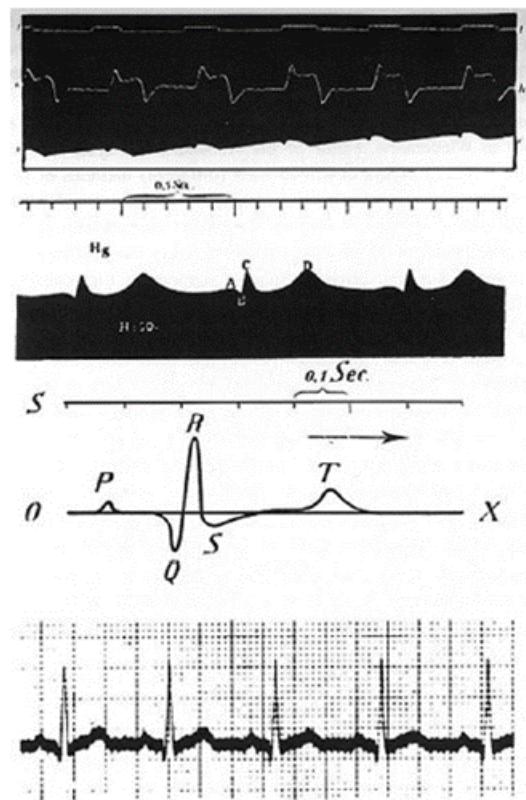


Figura 2.3: Ondas eletrofisiológicas do coração [6]

Em 1901, Einthoven descreveu a modificação realizada por ele, designada de galvanómetro de corda, conseguindo assim uma solução que viria a ser o alicerce da ECG. Este galvanómetro consistia num finíssimo filamento de quartzo recoberto por prata ($< 3\mu$), esticado num campo magnético criado por um eletroíman. Mesmo a corrente elétrica fraca de um potencial cardíaco seria capaz de mover o filamento. A oscilação deste dependia da magnitude e direção da corrente elétrica. As sombras geradas pela movimentação do fio de quartzo eram projetadas num filme fotográfico, que rodava à velocidade de 25 mm/s, de modo que cada deflexão de 1 mm na abcissa representava 0,04 segundos. Ajustando a tensão da corda, cada deflexão de 1 mm na ordenada correspondia a 10^{-4} V, como nos ECG atuais. A relação entre a amplitude do traçado e a voltagem era controlada pela tensão do filamento.

O formato de Einthoven para registar estas deflexões tornou-se o padrão que ainda é usado hoje. Em 1913, Einthoven introduziu o conceito de vetor cardíaco e defendeu o seu uso clínico na distinção entre hipertrofias e mudanças na posição do coração, inaugurando a vetocardiografia.

Frank N. Wilson foi quem mais se destacou nesse período. Introduziu as derivações unipolares em 1934, que consistiam de um elétrodo explorador e um indiferente, com potencial elétrico nulo. Mais tarde, foram adicionadas as derivações unipolares aumentadas (aVF, aVL e aVR).

Em 1931, Wilson descreveu a derivação unipolar, provando matematicamente a possibilidade do registo da atividade elétrica do coração em qualquer parte do corpo. Houve valorização da ECG do ponto de vista clínico, mas era necessário padronizar a disposição dos elétrodos das derivações precordiais. Isso foi definido, em 1938, pela *American Heart Association* e a *Cardiac Society of Great Britain*, quanto às derivações de V1 a V6. As 12 derivações eletrocardiográficas que atualmente são empregues foram estabelecidas graças às contribuições de Einthoven, Wilson e Emanuel Goldberger, o qual, em 1942, introduziu as derivações aumentadas aVR, aVL e aVF, a partir das derivações unipolares originais de Wilson, VR, VL e VF. Na Figura 2.4 é possível perceber as várias variações utilizadas no ECG.

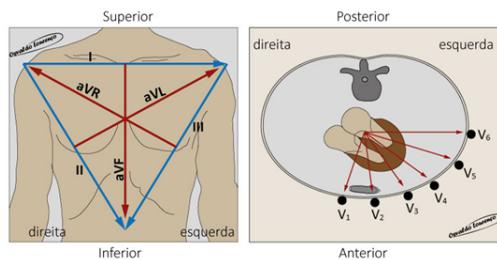


Figura 2.4: Sistema de Derivações Eletrocardiográficas [6]

Durante a década de 1930, a invenção dos ECG com registo direto em papel,

sem o processamento fotográfico, contribuiu para popularizar os aparelhos de ECG. Durante essa época, houve críticas devido ao facto de muitos médicos não estarem adequadamente treinados para a interpretação do ECG. Um acentuado desenvolvimento da eletrofisiologia cardiovascular aconteceu a partir da segunda metade do século XX, destacando-se o monitoramento por holter, os estudos eletrofisiológicos invasivos, o ECG de alta resolução e o mapeamento intracardíaco. Também se destacaram os registos eletrocardiográficos em pacientes submetidos a esforço, os testes ergométricos.

O monitoramento eletrocardiográfico ambulatorial (holter) tornou-se cada vez mais sofisticado desde a sua introdução, em princípios da década de 1960. Os dispositivos tornaram-se mais leves e compactos, a tecnologia digital tornou mais confiável a gravação da ECG, eliminando as distorções provocadas pelas fitas magnéticas e a necessidade de reprodução (*playback*). No princípio da década de 1990, surgiram os primeiros discos rígidos com tamanho reduzido e custo viável, os quais foram substituídos pelos cartões de memória instantânea (*flash*). A qualidade dos registos avançou a ponto de oferecer, inclusive, a ECG de alta resolução. O registo de holter típico apresenta duração de 24 a 48 horas, possibilitando a deteção de anormalidades eletrocardiográficas transitórias no contexto de vida normal do paciente. As aplicações dessa tecnologia são diversas, como a deteção de alterações do segmento ST, efeito terapêutico de antiarrítmicos e drogas anti-isquêmicas, variação da frequência cardíaca, análise de marca-passos e potenciais tardios com ECG de alta resolução. O holter representa também um importante instrumento para a abordagem inicial a pacientes com síncope e representativa probabilidade de apresentar arritmias. As técnicas de mapeamento (*mapping*) possibilitaram a determinação da sequência de ativação durante arritmias, além de permitirem melhor ajuste dos cateteres para marca-passo. As plataformas de registo digital, por meio de computadores, substituíram o registo analógico e de multicanal (técnica de 128 canais, por exemplo) e permitiram a interpretação em três e quatro dimensões das informações acerca da ativação e voltagem. A Eletrocardiograma de Alta Resolução (ECGAR) permite a deteção de sinais de baixa amplitude denominados “potenciais tardios”, impossível nos ECG convencionais. Esses potenciais tardios apresentam impacto no prognóstico de pacientes que sofreram enfarte agudo do miocárdio ou possuem aumento do risco de enfarte agudo do miocárdio subsequente e morte súbita. O exemplo de ECGAR consiste no *Signal-averaged ECG*, o qual realiza o registo representado pela média de vários sinais, reduzindo o ruído do traçado. A gama de frequências dos sinais varia de 0,05 Hz a 300 Hz contra 1 Hz a 80 Hz de um ECG tradicional. O ECGAR permite a realização de diagnósticos antes apenas possíveis com métodos invasivos ou minimamente invasivos.

Concluindo, tudo começou como curiosidade elétrica, o que fez com que fosse possível perceber melhor o corpo humano, possibilitando a evolução da medicina

e consequentemente o diagnóstico de doenças cardiovasculares. Entende-se o ECG como uma evolução do galvanómetro de corda, sendo um equipamento médico essencial para a avaliação da atividade elétrica do coração [6] [7]. O funcionamento do mesmo será referido mais abaixo neste mesmo capítulo.

2.2 Processos Fisiológicos

O coração, tem a função de bombear sangue para todos os órgãos, facilitando a troca de gases, absorvendo o oxigénio e eliminando o dióxido de carbono. O mesmo está dividido em câmaras superiores (aurículas) e inferiores (ventrículos), sendo que pode ser considerado que é composto por dois sistemas de bombagens separados [8]:

- O lado direito que permite que o sangue chegue ao pulmões.
- O lado esquerdo permite que o sangue chegue aos outros órgãos periféricos.

Cada aurícula ajuda a mover o sangue para o ventrículo correspondente, estes fornecem a principal força de bombear o sangue através dos sistemas de circulação pulmonar e periférica [8]. A Figura 2.5 representa o fluxo de sangue no coração.

Para além dos músculos auriculares e ventriculares, existe um tecido ou músculo estriado, designado de miocárdio, este é composto por fibras excitáveis e contrateis capazes de desenvolver atividade elétrica, em cada contração do mesmo, o que origina um batimento cardíaco [8]. O impulso elétrico originado pelo miocárdio, leva uma variação de potencial elétrico à superfície do corpo, o que permite o registo dos vários batimentos. Este registo é designado de ECG [8].

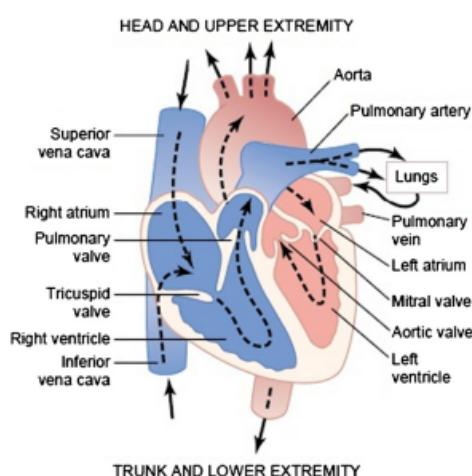


Figura 2.5: Coração [8]

2.2.1 O processo celular

Cada batimento cardíaco é originado a partir de um impulso elétrico, designado de potencial de ação, onde este é conduzido rapidamente através do mesmo, de modo a gerar uma contração [8].

Durante a fase de repouso (Fase 4), a distribuição dos iões de sódio, potássio e cálcio, geram uma diferença de potencial transmembranar, sendo esta espontânea e estável, denominada de potencial de repouso, tendo carga negativa na ordem -90mV, em relação ao exterior da membrana celular. Nesta fase a membrana apresenta uma característica peculiar, na qual a condutância para o íão de potássio é cerca de 10 vezes maior que para outros iões permitindo que o potencial de equilíbrio deste íão seja o potencial de repouso. O sódio encontra-se em altas concentrações fora da membrana e o contrário dentro das células. Além disso, durante esta fase inicial do potencial de ação, a membrana celular é impermeável ao Na+ [8].

Para iniciar a despolarização, um mecanismo complexo de canais iónicos (canal rápido) na membrana celular abre-se momentaneamente (a duração é de aproximadamente 1 ms), permitindo uma rápida entrada de Na+ na célula através do seu gradiente de concentração. Como agora há um fluxo de iões com carga positiva para dentro da célula, esta torna-se eletricamente positiva (aproximadamente +20 mV), enquanto o exterior da célula se torna negativo. Esta parte do potencial de ação é chamada de fase 0 (zero) [8].

Quando a fase 0 ocorre na célula muscular ventricular, ao mesmo tempo, o complexo QRS na ECG é registado. A onda P é gerada na fase 0, quando a mesma ocorre na célula muscular arterial. À medida que os mecanismos dos canais iónicos se fecham gradualmente e a entrada de Na+ diminui progressivamente o seu gradiente de concentração, a carga elétrica interna torna-se menos positiva, iniciando assim o processo de repolarização (fase 1) [8].

Durante a fase 2, o potencial de ação é aproximadamente isoelétrico, e a membrana celular permanece despolarizada. Neste ponto, uma pequena quantidade de Na+ entra através do canal rápido, enquanto o Ca+ e, possivelmente, uma quantidade menor de Na+, entram pelo canal lento. Esta fase ocorre quando o segmento ST está a ser registado na ECG [8].

A fase 3 representa a rápida repolarização, durante a qual o interior da célula volta a ser negativo. Isto é causado pelo aumento da saída ou movimento do K+ do interior para o exterior da célula. A fase 3 na célula do miocárdio ventricular ocorre durante o registo da onda T no ECG. Na Figura 2.6 é possível observar o potencial de ação de uma célula do miocardio ventricular.

A repolarização é concluída no final da fase 3. O interior da célula volta a ser aproximadamente -90 mV [8]. Durante a fase 4, um mecanismo especial de bombagem na membrana celular é ativado. Este transporta Na+ do interior para o exterior e traz o K+ de volta para dentro da célula. Este mecanismo depende de

Trifosfato de Adenosina (ATP) como fonte de energia e o processo volta-se a repetir [8]. O coração deve contrair-se aproximadamente a cada 0,75 segundos em repouso e até três ou mais vezes por segundo durante exercícios intensos [8].

A curva na fase 4 do potencial de ação de uma célula marcapasso, demonstrado na Figura 2.7, é um fator importante na formação do impulso. Quanto maior a inclinação, mais rápida é a frequência de formação do impulso, sendo que o contrário também se aplica [8].

Existem dois mecanismos básicos relativos à origem do impulso elétrico no miocárdio. O primeiro mecanismo é a despolarização, apresentado anteriormente, e o segundo é a reentrada que consiste em dois caminhos de condução, onde um dos quais tem um bloqueio unidirecional (ou um longo período refratário), e o outro tem uma condução lenta, onde o tempo de transmissão ao redor do circuito é grande o suficiente para que o período refratário das células de Purkinje se recupere, criando assim um circuito de transmissão circular [8]. O impulso cardíaco normal origina-se no nó sinusal. A condução a partir do nó sinusal ocorre através de três vias internodais. A velocidade de condução através do átrio é de aproximadamente 1000 mm/s [8].

O grupo mais importante de células marcapasso, ou *pacemaker*, são as encontradas no nó sinusal, nó AV e no sistema de condução ventricular. A frequência de disparo (despolarização espontânea) difere em cada um desses locais [8]. Na Figura 2.8 é possível perceber estes dois caminhos especificados.

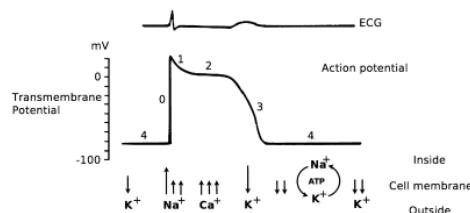


Figura 2.6: O potencial de ação de uma célula do miocárdio ventricular [8]

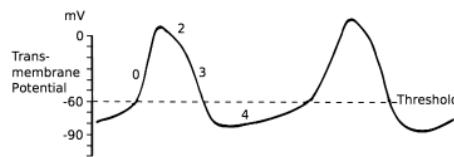


Figura 2.7: Potencial de ação de uma célula marca-passo (nóculo sinusal) [8]

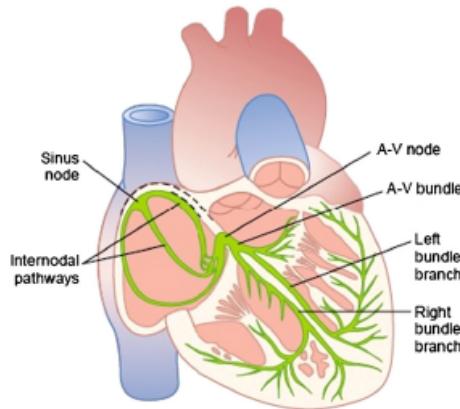


Figura 2.8: Esquema do sistema especializado de excitação e condução do coração que controla as contrações cardíacas [8]

2.2.2 Complexo QRS

O sinal ECG é um registo das diferenças de potencial produzidas pela atividade elétrica das células cardíacas. O corpo por si só atua como um gigantesco condutor de corrente elétrica e quaisquer dois pontos no corpo podem ser ligados por elétrodos elétricos para registar um ECG ou monitorizar o ritmo cardíaco. O traçado medido e registado utilizando equipamento adequado refere-se à atividade elétrica do coração e forma uma série de ondas e complexos que foram arbitrariamente denominados onda P, complexo QRS, onda T e onda U. As ondas ou deflexões são separadas por intervalos regulares [8]. As características das várias ondas são visíveis na Figura 2.9 .

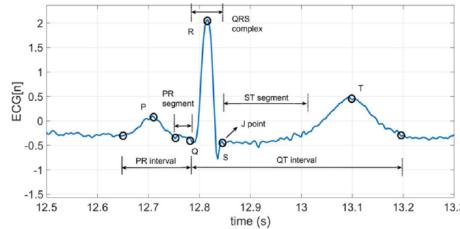


Figura 2.9: Ondas características [8]

A despolarização da aurícula produz a onda P e a despolarização dos ventrículos produz o complexo QRS. A repolarização ventricular causa a onda T. O significado da onda U é incerto, mas pode dever-se à repolarização do sistema de Purkinje [8].

O intervalo PR (iPR) estende-se desde o início da onda P até ao início do complexo QRS. Este não deve exceder 0,20 segundos medidos em papel de ECG, onde cada pequeno quadrado representa 0,04 segundos. O limite superior da duração normal do complexo QRS é < 0,12 segundos. Uma duração inferior a 0,12 segundos significa que o impulso foi iniciado no nó AV ou acima dele (supraventricular).

Uma duração do QRS > 0,12 segundos pode significar um impulso com origem no ventrículo ou com origem no tecido supraventricular, mas com condução prolongada através do ventrículo. As doenças cardiovasculares são detetadas através da leitura dos intervalos, ondas e complexos [8].

- Onda P: Se, por alguma razão, o nódulo sinusal falhar como célula normal de marcapasso, outro foco auricular pode assumir o controlo e, nesse caso, a onda P pode apresentar alterações na sua morfologia. Alternativamente, um segundo foco pode ser ativado, ou seja, um foco secundário do marcapasso (por exemplo, o nódulo AV), de modo a obter um ritmo de escape;
- Intervalo PR: quando a condução através da aurícula, nódulo AV ou feixe de His é lenta, o intervalo PR aumenta. As alterações na condução do nódulo AV são as causas mais comuns do aumento do PR;
- Complexo QRS: se houver um atraso ou interrupção na condução dentro dos ramos, o complexo QRS tipicamente alarga-se. Isto é, ocorre o bloqueio do ramo direito ou o bloqueio do ramo esquerdo. Um foco ectópico que inicia um impulso no ventrículo também pode alterar a forma do complexo QRS. Quando o foco ectópico tem início acima do feixe de His ou não, mas não nos ramos, os ventrículos são ativados normalmente, e o complexo QRS permanecerá o mesmo, assumindo que não há atraso na condução nos ramos. Se o foco ocorrer abaixo, o complexo QRS alargará devido a sequências de condução diferentes.

2.2.3 Princípios Físicos da ECG

O modelo matemático mais simples para relacionar a atividade elétrica cardíaca com os potenciais da superfície corporal é o modelo de dipolo único. A atividade elétrica total do coração pode em qualquer instante ser representada por uma distribuição de dipolos de corrente ativos [8].

Este modelo consiste em dois componentes: uma representação da atividade elétrica do coração e a geometria e propriedades elétricas do corpo. No que diz respeito à representação da atividade elétrica do coração, à medida que um potencial de ação se propaga através de uma célula, existe uma corrente intracelular associada e, consequentemente, uma propagação na direção da interface entre o tecido em repouso e o tecido em despolarização, que é a fonte elétrica elementar do ECG de superfície, referida como o dipolo de corrente. Considerando que também existe uma corrente extracelular igual a fluir contra a direção de propagação, a carga é conservada e todos os circuitos de corrente nos meios condutores fecham-se sobre si mesmos, formando um campo dipolar [8].

Na realidade, a bioeletricidade normalmente flui em circuitos de corrente, como ilustrado na Figura 2.10. Um circuito inclui quatro segmentos: uma travessia para

fora da membrana celular, um segmento extracelular, uma travessia para dentro da membrana celular e um segmento intracelular. Cada um dos segmentos contém vários aspectos que lhe conferem uma importância única para o circuito de corrente [8].

Para fins de modelação, podemos considerar que todos os dipolos de corrente individuais têm origem num único ponto no espaço, e a atividade elétrica total do coração pode ser representada como um único dipolo equivalente, cuja magnitude e direção é a soma vetorial de todos os dipolos individuais. Designando este momento dipolar resultante como o vetor cardíaco $M(t)$ e à medida que cada onda de despolarização se espalha pelo coração, o vetor resultante muda em magnitude e direção em função do tempo [8].

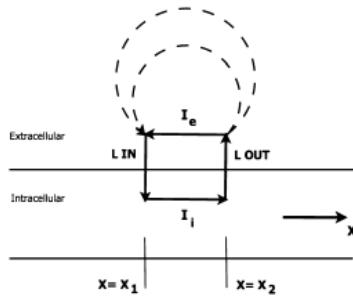


Figura 2.10: Fluxo de bioeletricidade em circuitos de corrente apresentando quatro segmentos: uma travessia para o exterior da membrana celular, um segmento extracelular, uma travessia para o interior da membrana celular e um segmento intracelular [8]

A configuração da distribuição superficial de correntes e potenciais depende das propriedades elétricas do tronco. Como uma aproximação razoável, o modelo dipolar pode considerar o corpo como um condutor linear, isotrópico, homogéneo, com condutividade σ , e esférico de raio R . Considerando que a fonte do potencial de superfície é representada como um único dipolo de corrente que varia lentamente no tempo, localizado no centro da esfera, a equação de Laplace pode ser resolvida para fornecer a distribuição de potencial no tronco como [8]

$$\theta(t) = \cos \theta(t) \cdot \frac{3|M(t)|}{4\pi\sigma R^2} \quad (2.1)$$

O ângulo $\theta(t)$ é o ângulo entre a direção do vetor cardíaco $M(t)$ e o vetor de derivação que une o centro da esfera ao ponto de observação ou medição. Portanto, a diferença de potencial entre dois pontos na superfície do tórax é dada por [8]

$$V_{AB}(t) = M(t) \cdot L_{AB}(t) \quad (2.2)$$

onde $L_{AB}(t)$ refere-se ao vetor condutor que liga diferentes pontos A e B de observação no tórax.

2.2.4 Triangulo de Einthoven

O ECG é tradicionalmente registado a partir de dois elétrodos localizados em diferentes lados do coração. Como resultado da atividade elétrica das células do miocárdio, a corrente flui dentro do corpo e estabelecem-se diferenças de potencial na superfície corporal [8].

Inicialmente, quando as aurículas se despolarizam, a onda de despolarização desce através de ambas as aurículas. O vetor pode ser decomposto em componentes apontando para baixo, sendo a componente mais predominante para a esquerda e ligeiramente para trás. Quando o impulso elétrico passa pelo nódulo AV, não há atividade elétrica mensurável na superfície do corpo, o que é referido como atraso no nódulo AV. Subsequentemente, a atividade elétrica despolariza o feixe de His e os ramos do feixe. De seguida, ocorre a despolarização septal, na qual à medida que a onda de potencial de ação entra no miocárdio septal, esta tende a propagar-se da esquerda para a direita. Portanto, o vetor cardíaco resultante aponta para a direita. A seguir, ocorre a despolarização apical, quando o vetor resultante aponta para o ápice do coração, que predominantemente aponta para baixo, para a esquerda do sujeito e ligeiramente anterior. Subsequentemente, observamos a despolarização esquerda, para a qual observamos atividade elétrica tanto no ventrículo esquerdo como no direito, mas predominantemente no muito mais volumoso ventrículo esquerdo [8].

Após todo o miocárdio se despolarizar, começa a contrair-se, e há um período em que nenhum potencial de ação se propaga (período refratário absoluto), e, portanto, não há vetor cardíaco mensurável. Após esse período de *plateau*, as células começam a repolarizar-se e outra onda de carga espalha-se pelos ventrículos, fazendo o coração retornar ao seu estado de repouso (repolarização ventricular). Como tanto a polaridade como a direção de propagação da fase de repolarização são invertidas em relação às da despolarização, as ondas T (relacionadas com as ondas de repolarização) no ECG são geralmente da mesma polaridade que os complexos QRS (relacionados com as ondas de despolarização) [8].

O vetor cardíaco dependente do tempo, com direção e magnitude variáveis, é projetado em 12 linhas diferentes com orientação bem definida, que são chamadas de 12 derivações [8].

Cada derivação fornece uma medida da magnitude do vetor cardíaco numa direção específica em cada instante de tempo. O primeiro conjunto importante de derivações são as chamadas derivações bipolares dos membros padrão, que compreendem a derivação I, II e III [8]. Na Figura 2.11 é possível observar estas três derivações.

Para cada derivação, a ECG é registada a partir de dois elétrodos localizados em diferentes lados do coração, por exemplo, nos membros. Os elétrodos são colocados em cada uma das quatro extremidades de uma pessoa. Cada par de elétrodos forma um circuito fechado entre o corpo e o ECG. Cada derivação bipolar dos membros é configurada da seguinte forma:

- Derivação I: O terminal negativo do ECG é ligado ao braço direito e o terminal positivo ao braço esquerdo;
- Derivação II: O terminal negativo é ligado ao braço direito e o terminal positivo à perna esquerda;
- Derivação III: O terminal negativo é ligado ao braço esquerdo e o terminal positivo à perna esquerda.

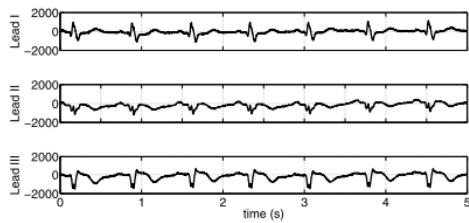


Figura 2.11: ECG registado a partir das três derivações eletrocardiográficas padrão dos membros: paciente diagnosticado com enfarte do miocárdio [8]

De acordo com o modelo do triângulo de *Einthoven*, desenhado em torno da área do coração, os dois braços e a perna esquerda formam os vértices de um triângulo que circunda o coração. Os dois vértices na parte superior do triângulo representam os pontos de ligação entre os dois braços e os fluidos em torno do coração, e o vértice inferior é o ponto em que a perna esquerda se liga aos fluidos. Com base no modelo de *Einthoven*, se os potenciais elétricos de quaisquer duas das três derivações dos membros forem conhecidos num dado instante, a terceira pode ser determinada matematicamente pela soma vetorial [8].

As derivações precordiais ou torácicas registam a atividade cardíaca no plano horizontal, portanto perpendicular ao plano focado pelas derivações dos membros. Isto requer que seis elétrodos sejam colocados em redor do tórax, por exemplo, na superfície anterior do peito, diretamente sobre o coração, num dos pontos mostrados na Figura 2.12 [8]. Cada elétrodo colocado em cada um dos pontos referidos é ligado a um terminal positivo específico do ECG, e ao elétrodo negativo, chamado terminal central. O mesmo princípio aplica-se ao conjunto de derivações torácicas, ligadas com base em resistências elétricas iguais ao braço direito, braço esquerdo e perna

esquerda, todos ao mesmo tempo. Os diferentes registos para as derivações precordiais são conhecidos como derivações V1, V2, V3, V4, V5 e V6. Finalmente, para as derivações unipolares aumentadas dos membros, dois dos membros (extremidades) são ligados com base em resistências elétricas ao terminal negativo do ECG, e o terceiro membro é ligado ao terminal positivo. Podemos interpretar esta configuração como a medição de um potencial numa dada extremidade em relação à média dos potenciais das outras duas extremidades [8].

As derivações unipolares precordiais fazem parte do sistema de derivações torácicas humanas, comumente utilizadas na prática clínica por seis derivações precordiais. Cada uma delas mede a diferença de potencial entre um elétrodo exploratório positivo (colocado numa posição específica no tórax e denominadas de V1 a V6) e a referência do Terminal Central de *Wilson*. Por último, as derivações unipolares aumentadas dos membros medem o potencial de um membro (elétrodo positivo) referenciado contra uma combinação dos outros elétrodos dos membros (desconsiderando a perna direita). Os elétrodos positivos para estas derivações aumentadas estão localizados no braço direito (chamada de aVR), no braço esquerdo (aVL) e na perna esquerda (aVF). Na prática, os elétrodos positivos exploratórios utilizados nas derivações unipolares aumentadas dos membros são os mesmos elétrodos utilizados para as derivações bipolares dos membros Figura 2.13 [8].

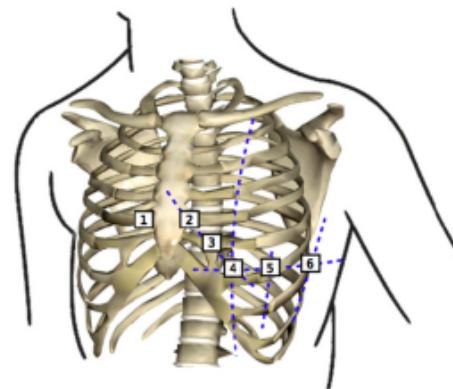


Figura 2.12: Esquema para a colocação de elétrodos com o objetivo de registar as derivações precordiais: V1, V2, V3, V4, V5 e V6 [8]

Cada derivação unipolar aumentada dos membros é configurada da seguinte forma:

- Derivação aVR: O terminal positivo está no braço direito, e os elétrodos colocados na perna esquerda e no braço esquerdo estão ligados ao terminal negativo (média dos potenciais na perna esquerda e no braço esquerdo);

- Derivação aVL: O terminal positivo está no braço esquerdo, e os elétrodos colocados na perna esquerda e no braço direito estão ligados ao terminal negativo (média dos potenciais na perna esquerda e no braço direito);
- Derivação aVF: O terminal positivo está na perna esquerda, e os elétrodos colocados no braço esquerdo e no braço direito estão ligados ao terminal negativo (média das derivações dos braços).

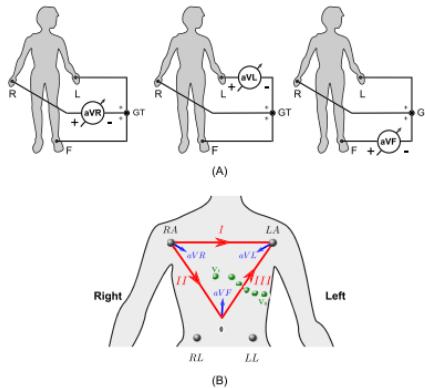


Figura 2.13: (A) Esquema do circuito aumentado da derivação unipolar dos membros, neste exemplo avR, obtido pela diferença entre o potencial de um único elétrodo R no braço direito e o GT. (B) Representação do tronco com o ECG de 12 derivações [8]

Uma vez que as derivações unipolares aumentadas dos membros são uma combinação linear das derivações bipolares dos mesmos, e de acordo com a equação do triângulo de *Einthoven*, qualquer derivação bipolar dos membros é calculada em função das outras duas sendo possível concluir que com apenas duas derivações bipolares dos membros, a terceira e as derivações unipolares aumentadas dos membros são obtidas por uma simples manipulação matemática. Isto é de significativa importância para os sistemas de ECG computorizada e para a poupança de memória [8]. A ilustração de todas as derivações contemplando o ECG padrão de 12 derivações está agora resumida na Figura 2.13B.

2.3 Características elétricas de um sinal de ECG.

Sendo necessária a maior precisão possível, de modo a obter dados fidedignos e que sejam capazes de manter a qualidade do sinal recolhido mesmo em ambientes desfavoráveis, implica que a resolução e o processamento de sinal possibilitem distinguir um sinal cujos parâmetros são:

- Gama de batimentos cardíacos – Entre 30 a 200 bpm;

- Amplitude do complexo QRS – 0.1 a 20 miliVolts;
- Amplitude da onda P – 5% a 40% da amplitude do complexo QRS (miliVolts);
- Amplitude da onda T – 10% a 80% da amplitude do complexo QRS (miliVolts).

Com isto é perceptível que é necessário suprimir ao máximo o ruído elétrico e selecionar os melhores componentes de modo ao sinal não ser corrompido [9].

A gama de frequências do sinal de um ECG de um humano situa-se entre 0.05 e 150 Hz.

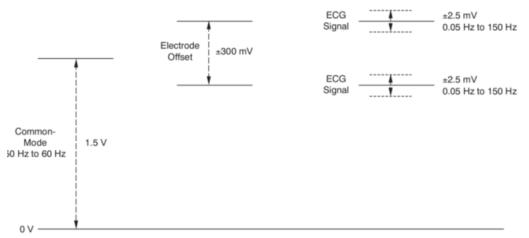


Figura 2.14: Gama de frequências [9]

Na Figura 2.14 é possível visualizar as características elétricas de um sinal de ECG.

Percebe-se que existem valores elevados para o *offset*, mas isso explica-se pela tensão que é gerada nos elétrodos de Ag/AgCl, sendo estes os mais utilizados numa ECG, onde podem atingir tensões de +/-300mV.

Como as amplitudes que se pretendem adquirir estão na ordem dos poucos mV, adicionando uma interferência de 50/60 Hz proveniente da energia fornecida, torna-se evidente a dificuldade que se encontra na aquisição e no processamento de sinal.

Sendo assim é necessário detetar as origens dos ruídos existentes no sinal. Existindo três fontes que incrementam o ruído no sinal e que devem ser combatidas.

- *Baseline Drift*: o *offset* proveniente dos elétrodos colocados no utilizador, denominada por *baseline drift* e que é causada por movimentos corporais mínimos ou pela distribuição heterogénea do potencial elétrico no local em que é feito o exame.
- Interferência da alimentação energética(50Hz): É possível remover esta interferência aplicando um filtro *notch* a uma frequência de 50Hz, no domínio digital.
- Interferência proveniente dos músculos.

Uma vez que o sinal tem uma amplitude tão baixa é necessário amplificar o sinal para que possa ser amostrado e esta tensão de *offset* vai limitar o valor máximo de

ganho que poderia ser obtido através da utilização de um amplificador de instrumentação. Para combater a tensão de modo comum é necessário utilizar amplificadores de instrumentação com elevado rácio de rejeição de modo comum, na ordem dos 100 dB. É também necessário recorrer à utilização de filtros, que podem ser feitos analogicamente, utilizando *hardware*, ou digitalmente, recorrendo a *software* [9].

2.3.1 Elétrodos

O mecanismo da condutividade elétrica no corpo envolve o movimento de iões portadores de carga, sendo necessário fazer a transdução desta corrente iônica para corrente elétrica. Este processo de transdução elétrica é feito por elétrodos, transdutores elétricos que convertem em corrente elétrica os potenciais elétricos do coração e que resultam das movimentações iônicas que ocorrem ao longo das suas células.

Os elétrodos usados para uma monitorização continua de sinais bioelétricos, implica requisitos específicos, tais como apresentação de uma interface estável com a superfície do corpo onde se encontram instalados, optando-se, habitualmente, por elétrodos não-polarizados para este tipo de aplicações. Através da otimização da estabilidade mecânica dos elétrodos consegue-se reduzir o ruído existente e, dessa forma, foram encontradas várias técnicas que permitem minimizar a deterioração do sinal através da interface. Um exemplo é a utilização de um fluido ou gel [9].

Existem 3 tipos de elétrodos:

- Elétrodos ECG peito de sucção - Ventosas
- Pinça
- Convencionais/descartaveis ou de repouso



Figura 2.15: Tipos de elétrodos

2.4 Sistemas de aquisição

Atualmente, existem diversas abordagens para registrar o sinal de ECG. Estas podem ser divididas em três classes:

- No interior da pessoa: existem equipamentos concebidos para serem utilizados dentro do corpo humano, tais como os implantados cirurgicamente, aplicações

subdérmicas ou mesmo ingeridos sob a forma de comprimidos. Estes dispositivos são utilizados quando abordagens menos invasivas não são aplicáveis ou eficazes

- Sobre a pessoa: o utilizador precisa de usar o sensor, tendo-o em contacto com o seu corpo.
- Fora da pessoa: Medem o ECG sem contacto com a pele ou com contacto mínimo. Sistemas sem contacto baseados em sensores capacitivos, que medem os pequenos campos elétricos variáveis no tempo associados à atividade bioelétrica do coração. Estes sensores não requerem contacto direto com o corpo do utilizador e podem ser concebidos para medir o ECG a distâncias de 1 cm ou mais, mesmo com roupa entre o corpo e o sensor.

A categoria mais utilizada é a sobre a pessoa, que funcionam através da fixação de um dispositivo, ou alguns dos seus componentes, externamente à superfície do corpo. Por exemplo, *holters* e sistemas ambulatórios, podemos também citar *t-shirts* inteligentes e outros formatos vestíveis [8].

Um sistema de instrumentação de derivações de ECG é concebido para medir diferenças sutis de potencial na pele resultantes da onda de despolarização do coração. Como se pode observar na Figura 3.1, é difícil medir apenas o biopotencial do ECG. O que é efetivamente medido é uma soma do ECG desejado com vários outros sinais, tais como a atividade biopotencial dos músculos e nervos sob os elétrodos, a interferência da rede elétrica de 60 Hz/50 Hz, artefactos de movimento de baixa frequência, potenciais de contacto dos elétrodos *Direct Current* (DC) (potencial de meia-célula), e todo o tipo de ruído de alta frequência produzido por ondas eletromagnéticas [8].

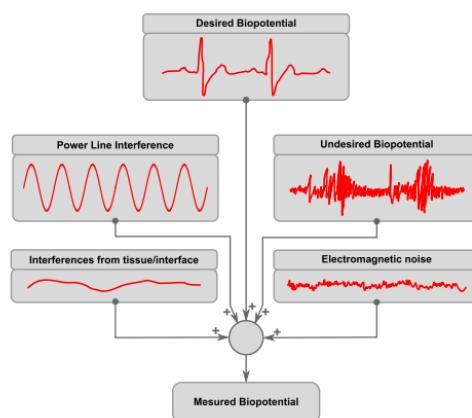


Figura 2.16: Componentes do biopotencial medido [8]

Os potenciais de repouso das células cardíacas são da ordem das décimas de milivolts, mas a atenuação produzida pelos músculos e tecido cutâneo faz com que

o sinal de ECG disponível no elétrodo seja apenas da ordem de alguns milivoltos. No entanto, a soma de outros potenciais indesejados é frequentemente uma ordem de grandeza (ou mais) superior a isso. É por isso que precisamos de um circuito condicionador de sinal entre o elétrodo e a fase de digitalização. O condicionador de sinal é um conjunto de circuitos analógicos que manipula um sinal analógico para satisfazer os requisitos da fase seguinte, que, neste contexto, é a conversão analógica para digital. Adicionalmente, o condicionador de sinal deve cumprir outros requisitos de design [8]:

- amplificar o sinal de ECG sem distorção;
- rejeitar interferências externas;
- filtrar artefactos produzidos pelo corpo humano;
- proteger o paciente e o operador contra o risco de choque elétrico;
- evitar danos por sobretensão produzida por desfibriladores e equipamentos eletrocirúrgicos.

Estes componentes podem ser visualizados na Figura 2.17

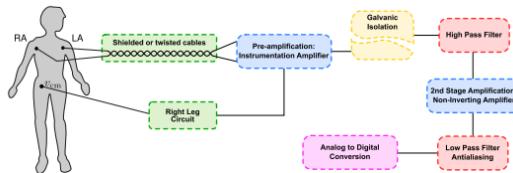


Figura 2.17: Principais componentes de um condicionador de sinal ECG [8]

A primeira e mais importante parte do condicionador de sinal é a etapa de pré-amplificação. Uma vez que uma derivação de ECG é a diferença de potencial entre dois elétrodos, a solução mais simples seria utilizar um amplificador diferencial [8]. Para aumentar a impedância de entrada, dois *buffers* são colocados na entrada, como mostrado na Figura 2.18 .

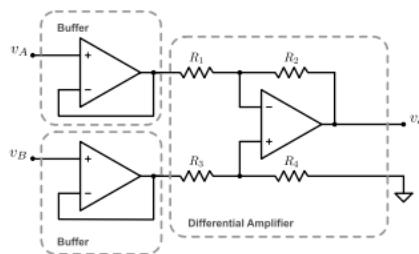


Figura 2.18: Amplificador diferencial com seguidores de tensão para aumento da impedância de entrada

Um amplificador operacional ideal é um amplificador com ganho em malha aberta infinito, resistência de entrada infinita e resistência de saída nula. A consequência destas suposições é que as entradas não consomem corrente ($i_+ = 0, i_- = 0$) e o amplificador operacional tenta fazer o que for necessário para tornar a diferença de tensão entre as entradas igual a zero ($v_+ = v_-$) [8].

Considerando o modelo ideal de amplificador operacional, a relação entre a saída v_o e as entradas v_A e v_B do amplificador diferencial é apresentada na Figura 2.19. A partir desta equação, se a relação $R_1/R_2 = R_3/R_4$ for observada, o ganho de amplificação pode ser definido escolhendo a razão entre as resistências R_2 e R_1 :

$$v_o = \frac{R_2(1 + R_1/R_2)}{R_1(1 + R_3/R_4)}v_A - \frac{R_2}{R_1}v_B \quad (2.3)$$

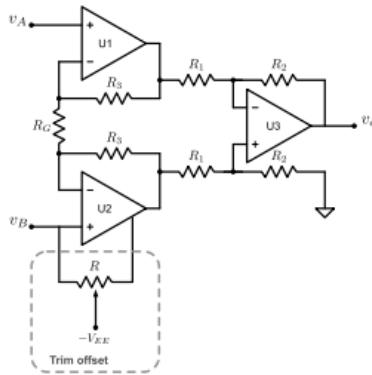


Figura 2.19: Amplificador de instrumentação clássico (pré-amplificação)

Uma desvantagem do circuito da Figura 2.19 é que, como os seguidores de tensão têm ganho unitário, toda a rejeição de modo comum é deixada para o amplificador diferencial. Isto só pode ser alcançado com uma correspondência de resistências muito precisa. A Figura 2.19 apresenta um circuito com uma solução engenhosa para este problema, conhecido como o amplificador de instrumentação clássico. Os amplificadores operacionais U1 e U2 são usados para aumentar a amplitude do sinal e também para reduzir a representação de qualquer potencial que ambos os sensores possam ter em comum, em relação ao terra da fonte de alimentação. Assim, a saída representa um sinal com redução substancial no sinal de modo comum quando comparado ao circuito na Fig. 2.19. Se $R_1 = R_2$, então a relação entrada-saída na Figura 2.19 é dada pela seguinte equação [8]:

$$v_o = \left(1 + \frac{2R_3}{R_G}\right)(v_B - v_A) \quad (2.4)$$

Note-se que o ganho $G = 1 + 2R3/RG$ multiplica tanto as componentes *Alternating Current* (AC) como DC de $(v_B - v_A)$. Como consequência, se a componente DC for grande, pode saturar o amplificador operacional U3. Por esta razão, em aplicações de ECG, é conveniente colocar um potenciómetro na entrada v_B para ajustar os desvios produzidos pelos potenciais de meia-célula. Um circuito integrado com o amplificador de instrumentação mostrado na Fig. 2.20 está disponível de vários fabricantes. Exemplos típicos são o INA128/129 de baixo consumo e uso geral, e o AD624 de alta precisão.

2.4.1 Rejeição de modo comum

Num circuito elétrico, é comum que o potencial absoluto da entrada v_A mude na mesma quantidade que a entrada v_B . O amplificador de instrumentação tenta amplificar $v_B - v_A$, mas esta diferença flutua ao longo do tempo em relação à terra com uma tensão desconhecida $v_{cm}(t)$. Isto é exatamente o que acontece quando desejamos medir o potencial de uma derivação, mas o potencial do corpo flutua aleatoriamente em relação à terra, porque o corpo atua como uma antena que capta a radiação eletromagnética circundante [8]. No caso ideal, a saída de um amplificador diferencial seria uma função apenas da diferença.

$$v_B + \delta \cdot v_{cm} - (v_A + \delta v_{cm}) = v_A - v_B \quad (2.5)$$

No entanto, a saída de um amplificador real é dada pela equação:

$$v_o = G_D(v_B - v_A) + G_C \left(\frac{v_B + v_A}{2} \right) \quad (2.6)$$

onde G_D é o ganho diferencial, e G_C é o ganho em modo comum. Isto significa que uma pequena fração do potencial médio entre as entradas é inadvertidamente adicionada à saída. Isto acontece porque é difícil obter uma correspondência exata entre as resistências. Por exemplo, no amplificador diferencial da Figura 2.19, se

$$R1/R2 - R3/R4 \quad (2.7)$$

não for exatamente zero, então o termo $G_C(v_B + v_A)/2$ é adicionado à equação. O *Common Mode Rejection Ratio* (CMRR) é uma figura de mérito criada para indicar quão bem um amplificador pode rejeitar uma porção da média das entradas que é inadvertidamente adicionada à saída. É definida como a razão entre o ganho diferencial G_D e o ganho de modo comum G_C . Alternativamente, esta quantidade é também expressa em decibéis. Neste caso, é referida simplesmente como *Common Mode Rejection* (CMR) [8].

2.4.2 Blindagem ativa dos cabos de entrada.

A interferência eletromagnética está presente em todo o lado, especialmente em ambientes hospitalares. Como o corpo humano é eletricamente condutor, uma corrente elétrica i_c pode ser induzida por ondas eletromagnéticas que atingem o corpo. Se a resistência do corpo à terra for R_0 , então desenvolve-se um potencial de interferência de modo comum $v_{cm} = i_c \cdot R_0$. Este potencial elétrico está presente em ambas as derivações de um elétrodo diferencial. Parte desta interferência é rejeitada pelo amplificador de instrumentação, mas é possível uma melhoria adicional através da proteção dos cabos de entrada [8]. Se a resistência de ganho R_G da Figura 2.19 for dividida em duas resistências de igual valor $R_G/2$, formamos um divisor de tensão com o potencial médio entre v_A e v_B no nó v_g . A proteção da entrada pode ser feita colocando um *buffer* para seguir esta tensão na blindagem metálica de um cabo coaxial. Esta situação é ilustrada na Figura 2.20 e é usada para reduzir os efeitos da capacidade e fuga do cabo [8]. Outra melhoria é alcançada pelo que é conhecido na literatura como circuito da perna direita. O circuito da perna direita remove ativamente a tensão de modo comum v_{cm} do corpo usando o princípio de realimentação negativa. A ideia principal é inverter e amplificar a saída do *buffer* e realimentá-la novamente para o corpo, com um amplificador operacional inversor. A derivação da perna direita, por convenção padrão, é utilizada como terra ou referência do circuito [8].

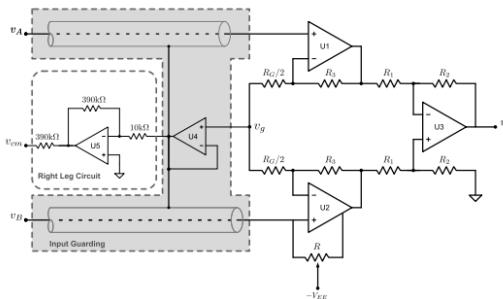


Figura 2.20: Amplificador de instrumentação clássico (pré-amplificação) [8]

2.4.3 Filtros e Conversores de Sinais

O sinal de ECG contém, para além do conteúdo clínico e fisiológico cardíaco relacionado com o complexo QRS, ondas P e T, conteúdo de ruído relacionado com interferência de 50/60 Hz da rede elétrica, sinal de Eletromiografia (EMG) da atividade muscular esquelética, artefactos de movimento da interface elétrodo-pele e outras fontes de interferência de equipamentos eletrocirúrgicos na clínica ou sala de operações. Apesar das variações morfológicas inerentes e típicas dentro das ondas características do ECG, de batimento para batimento num único registo de paciente,

e mais evidentemente de paciente para paciente, o complexo QRS, a onda P e a onda T são analisados como características distintivas, e a identificação automática destas características é, até certo ponto, uma questão abordável. No entanto, remover e até quantificar a informação de ruído no ECG não é simples e nem sempre é uma tarefa alcançável. Isto deve-se parcialmente ao facto de existirem tantos tipos diferentes de interferência e artefactos que podem ocorrer simultaneamente, e também porque estes ruídos são frequentemente transitórios e imprevisíveis em termos do seu início e duração [8].

Em resumo, o ruído do sinal de ECG pode ser classificado como:

- Interferência da rede elétrica: ruído de 50/60 Hz da rede;
- Ruído de contacto ou desconexão do elétrodo: perda de contacto entre o elétrodo e a pele, manifestando-se como alterações bruscas;
- Artefactos de movimento paciente-elétrodo: movimento dos elétrodos para longe da área de contacto na pele, levando a variações na impedância entre o elétrodo e a pele, causando variações potenciais no ECG e geralmente manifestando-se como saltos rápidos da linha de base ou saturação completa por até 0,5 s;
- Ruído de EMG: atividade elétrica devida a contrações musculares com conteúdo espectral entre 0 e 10.000 Hz, com uma amplitude média de 10% da deflexão de escala completa;
- Desvio da linha de base: geralmente da atividade respiratória com uma amplitude de cerca de 15% da deflexão de escala completa em frequências entre 0,15 e 0,3 Hz [8].

Na Figura 2.21 está apresentada uma onda típica de um ECG, as respetivas fases da mesma separadas e ainda são apresentadas ondas características dos ruídos, nomeadamente o ruído que os músculos produzem e o ruído dos movimentos do corpo, que existem ao se recolher os impulsos elétricos de um paciente para se observar um ECG.

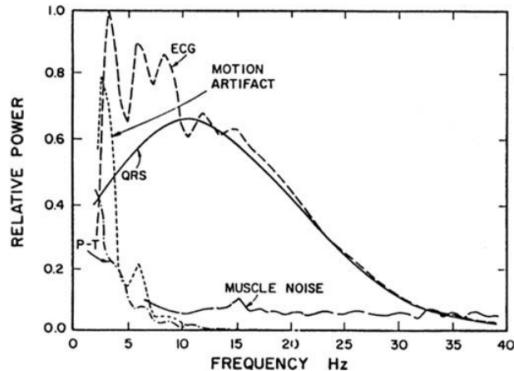


Figura 2.21: O espectro de potência típico do sinal de ECG inclui os seus subcomponentes e artefactos comuns (ruído de movimento e muscular). A potência das ondas P e T (P-T) é de baixa frequência, e o complexo QRS está concentrado na gama de frequências médias, embora a potência residual estende-se até 100 Hz [10]

De acordo com o seu estudo, como observado em Thakor et al. (1984), o conteúdo espectral do ECG clínico estende-se de cerca de 2,5 Hz a cerca de 40 Hz. O ruído muscular sobrepõe o conteúdo de frequência relacionado tanto com o complexo QRS como com as ondas P/T, enquanto o artefato de movimento sobrepõe, basicamente, conteúdo de frequência relacionado com as ondas P/T [8]. Relativamente ao complexo QRS, que é a onda característica do ECG mais enfatizada e sendo a sua deteção o primeiro passo na análise automática do ECG, foi demonstrado que:

- Num QRS típico de duração normal, praticamente toda a potência está contida em frequências abaixo de 30 Hz, com o pico de potência ocorrendo na faixa de 4 a 12 Hz;
- Contrações Ventriculares Prematuras (PVC) tipicamente contêm menos potência de alta frequência do que batimentos normais, com praticamente toda a potência contida em frequências abaixo de 12 Hz, com pico de potência em cerca de 4 Hz;
- O QRS é acompanhado por um alargamento definido da densidade espectral de potência com potência significativa presente na faixa de 30-100 Hz, embora a potência adicional ainda seja muito menor do que a contida na banda abaixo de 30 Hz.

Conversor Analógico-Digital

Um ADC é um dispositivo que tem como objetivo converter uma quantidade física num valor digital. Esta conversão, que habitualmente envolve variáveis elétricas, é feita periodicamente e origina uma sequência de valores numéricos que traduzem a variação do parâmetro físico em função do tempo. Um dos parâmetros mais

importante que diferenciam um ADC é a resolução (número de valores discretos disponíveis para amostrar o sinal, expresso em número de bits) [9].

ADC de menor resolução (≤ 16 bits) vs. ADC de maior resolução (24 bits)

Uma das grandes dificuldades do processamento de sinais de ECG advém do facto destes serem adquiridos com valores de amplitude baixos e com altos níveis de ruído/interferência. Assim sendo, relativamente à escolha dos ADC, vão existir duas possibilidades que passam pela opção por um ADC de baixa (≤ 16 bits) ou alta (24 bits) resolução. Aquando da utilização de *Local Area Network* (LAN) pode-se obter um elevado ganho sobre o sinal inicial, na ordem de 500, podendo ser utilizado um ADC de baixa resolução, como apresentado na Figura 2.22. Aqui, deve ser tido em consideração que o ruído do amplificador que é inserido no sinal durante a sua amplificação não domina o ruído geral do sistema.

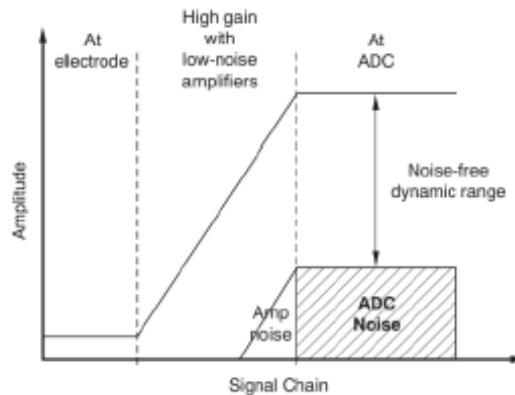


Figura 2.22: Sinal com um ADC de menor resolução [9]

Por sua vez é possível utilizar um ganho menor (na ordem de 5) e um ADC de alta resolução, ainda que o *noise-free dynamic range* se mantenha igual. Ainda assim, esta decisão tem um impacto bastante significativo nas especificações dos componentes individuais do sistema e no seu custo total, que é um dos fatores referenciados a ter em conta. Aqui, a utilização de um ADC de alta resolução reduz consideravelmente o *hardware* necessário e, consequentemente, as implicações relativamente ao custo e ao consumo. Na Figura 2.23 encontra-se especificada a amplitude do sinal de ECG e do respetivo ruído com a utilização de um ADC de alta resolução [9].

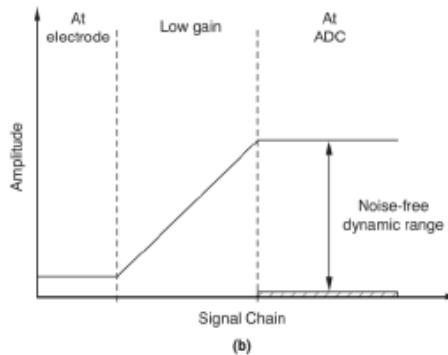


Figura 2.23: Sinal com um ADC de maior resolução [9]

Filtragem analógica

Mesmo utilizando proteção de cabos e realimentando a média do potencial medido para o corpo, com o circuito da perna direita, algum ruído é inevitavelmente captado durante a fase de pré-amplificação, representado na Figura 2.17. Isto significa que o amplificador irá multiplicar tanto o sinal como o ruído. Esta é uma das razões para dividir a amplificação em duas fases. Se fosse utilizada apenas uma fase de amplificação, o ruído seria amplificado tanto quanto o sinal, o que resultaria numa fraca Relação Sinal-Ruído (SNR). Por norma, recomendam utilizar um ganho de 25x na primeira fase para uma SNR adequada. O circuito de amplificação da segunda fase é um simples amplificador operacional não-inversor. Esta segunda fase de amplificação aumenta ainda mais a SNR do sinal e pode ser configurada para elevar a tensão do sinal para uma amplitude adequada à gama dinâmica do conversor A/D. O filtro passa-alto é preferencialmente colocado entre a primeira e a segunda fases de amplificação [8].

O Filtro Passa-Alto

Os sinais de ECG são frequentemente contaminados por diferenças de potencial de meia-célula na interface pele-eléktrodo (desvio DC), alterando também a impedância do eléktrodo. Isto pode ser causado pelo movimento do eléktrodo resultante da respiração do paciente, movimento, tremor muscular, alterações de temperatura ou humidade da pele e fuga de gel [8]. As alterações na interface pele-eléktrodo resultam em flutuações do potencial do ECG que têm mais conteúdo no espectro de baixa frequência do que as apresentadas num ECG "padrão". No domínio do tempo, observa-se uma lenta derivação da linha de base. Este fenómeno pode ser atenuado por um filtro passa-alto colocado após a primeira fase do amplificador. Existem várias formas de implementar filtros analógicos passa-alto e passa-baixo

e a 2^a fase de amplificação. Uma forma simples consiste em utilizar filtros *Resistor–Capacitor* (RC) analógicos passivos de primeira ordem em série com um amplificador operacional não-inversor, como demonstrado na Figura 2.24. Pode-se escolher frequências de corte (f_{ca} e f_{cb}) ajustando resistências e condensadores de modo que $f_{ca} = 1/(2 \cdot \pi R_a C_a)$ e $f_{cb} = 1/(2 \cdot \pi R_b C_b)$, e definindo o ganho do amplificador não-inversor como $G = 1 + R_2/R_1$. Uma das desvantagens deste desenho é a fraca inclinação dos filtros de primeira ordem (-20 dB/década). Se o critério de *Nyquist* fosse estritamente cumprido, isto é, se a frequência de amostragem fosse o dobro da frequência de corte do filtro passa-baixo, algum *aliasing* poderia ocorrer. A resposta transitória de um filtro é tão importante quanto a sua resposta em frequência, uma vez que a precisão do complexo QRS deve ser preservada para um diagnóstico correto do paciente. Por exemplo, quanto mais a resposta do filtro se afastar de um filtro ideal, menos acentuado poderá ser o pico R no complexo QRS. Adicionalmente, se a resposta em frequência do filtro estiver longe de um filtro ideal, podem estar presentes oscilações próximas da frequência de corte (ressonância). O filtro ativo de *Bessel* de fase linear é um filtro adequado para aplicações de ECG porque a sua resposta em frequência característica é livre de ressonância e tem excelente fase linear. Embora estas características contribuam para preservar a morfologia do ECG, este filtro sofre de uma inclinação lenta na banda de transição. Para ultrapassar esta limitação, podem ser utilizadas ordens superiores e, para cada incremento na ordem do filtro, a inclinação aumenta em -20 dB. É recomendado um filtro passa-alto de *Bessel* de oitava ordem nesta fase. A frequência de corte desejada do filtro passa-alto depende da aplicação particular envolvida. A banda de filtragem de frequência sugerida para o sinal de ECG, sem introduzir perturbações, situa-se entre 0,05 Hz e 250 Hz [8].

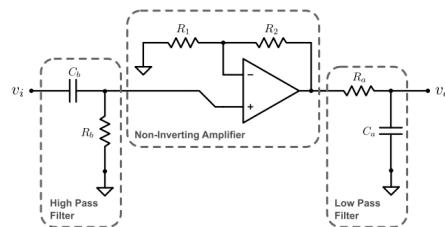


Figura 2.24: Filtragem analógica com filtros passivos de 1^a ordem e 2^a fase de amplificação com amplificador não-inversor [8]

O Filtro Passa-Baixo

A atividade elétrica dos músculos esqueléticos pode adicionar componentes indesejáveis aos ECG, especialmente durante testes de esforço ou sob monitorização contínua 24/7 (monitorização *Holter*). A maioria dos componentes espectrais destas fontes está acima do conteúdo do espectro de frequência do ECG e os seus efeitos podem

ser atenuados por um filtro passa-baixo. Se pacientes adultos estiverem sob monitorização, o corte deste filtro passa-baixo poderia ser 150 Hz. O filtro do tipo analógico também pode ser o filtro de *Bessel*, uma vez que este filtro minimiza a distorção de fase. Outra vantagem da implementação do filtro passa-baixo é reduzir a influência do efeito de *aliasing* que surge no processamento digital do ECG derivado do amplificador analógico [8].

Filtros Passa-Banda

Como já foi referido previamente, o filtro passa-banda resulta do agrupamento de um filtro passa-alto e de outro filtro passa-baixo [9]. Na Figura 2.25 está representado a implementação analógica de um filtro passa-banda.

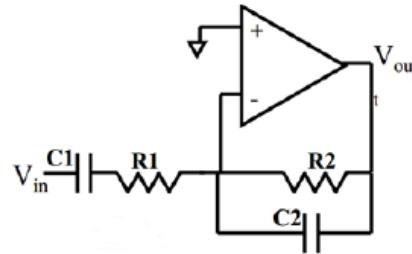


Figura 2.25: Exemplo de um filtro ativo passa-banda [9]

Neste caso, iremos ter para o ganho do circuito,

$$\frac{V_{out}}{V_{in}} = \frac{R_2 \parallel \frac{1}{C_2 \cdot s}}{R_1 + \frac{1}{C_1 \cdot s}} = - \left(\frac{1}{1 + R_2 \cdot C_2 \cdot s} \right) \left(\frac{R_2 \cdot C_1 \cdot s}{1 + R_1 \cdot C_1 \cdot s} \right) \quad (2.8)$$

E, consequentemente, as frequências de corte virão iguais a:

$$f_{cL} = \frac{1}{2\pi R_1 C_1} \quad (2.9)$$

$$f_{cH} = \frac{1}{2\pi R_2 C_2} \quad (2.10)$$

Filtros *Notch* (50/60 Hz)

Um dos maiores problemas no que diz respeito à deteção e ao processamento de biopotenciais é a interferência proveniente da alimentação de energia. Estas frequências são usualmente simples de remover com recurso a um simples amplificador diferencial, que servem para eliminar sinais de modo comum através do *Right Leg Drive* que vem descrito à frente com maior pormenor. Contudo, e infelizmente, esta interferência de 50/60 Hz corresponde à banda de frequências onde os biopotenciais têm a sua maior energia, tendo de se recorrer a um filtro *notch* para rejeitar, tal como foi

previamente dito, esse intervalo de frequências [9]. O esquema do filtro *notch* está apresentado na Figura 4.8.

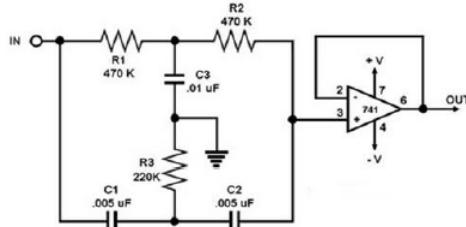


Figura 2.26: Esquema de um filtro notch [9]

2.5 Segurança elétrica

O equipamento de ECG expõe os pacientes a riscos elétricos porque o elétrodo está em contacto direto com a pele do paciente e a resistência da interface elétrodo-pele é significativamente reduzida pelo gel do elétrodo. Isto pode criar um caminho de baixa impedância entre a linha de alimentação e o paciente, aumentando o risco de choque elétrico em caso de falha do circuito elétrico do ECG. Um segundo mecanismo de risco ocorre quando o operador toca simultaneamente em partes metálicas do equipamento e no paciente. Como a impedância do corpo é finita, qualquer corrente de fuga pode fluir do *rack* do equipamento para o paciente, passando pelos braços do operador e retornando à terra através do corpo do paciente. Estes problemas podem ser abordados com o uso de amplificadores de isolamento. Estes dispositivos podem ser usados para quebrar *loops* de terra, eliminar conexões de terra da fonte, fornecer proteção de isolamento do paciente contra a linha de alimentação e proteger o equipamento de ECG de altas tensões de entrada produzidas por desfibriladores ou outros equipamentos eletrocirúrgicos. As alternativas mais comuns para promover o isolamento consistem em aplicar transformadores, condensadores ou acopladores óticos. A ideia básica é separar o circuito frontal (ou seja, a parte do sistema de ECG que está em contacto direto com o paciente) do circuito de retaguarda (que está ligado à linha de alimentação e/ou a um computador ou monitor). Assim, a alimentação do circuito frontal pode ser feita por baterias recarregáveis, enquanto o circuito de retaguarda é alimentado pela linha de energia [8].

Um terceiro mecanismo de falha ocorre quando o equipamento de ECG é usado simultaneamente com outros equipamentos médicos. Desfibriladores e outros equipamentos eletrocirúrgicos aplicam uma corrente variável que pode induzir uma grande tensão transitória no elétrodo do ECG, o que pode danificar amplificadores e circuitos condicionadores. A corrente para o paciente pode ser limitada colocando

resistências de 1Ω no cabo de derivação. Também é recomendado conectar dispositivos limitadores de tensão entre cada elétrodo de medição e a terra elétrica. Estes dispositivos podem ser [8]:

- (i) díodos de silício em paralelo
- (ii) díodos Zener, conectados em oposição ou
- (iii) tubos de descarga de gás

Todos eles aparecem como um circuito aberto quando a tensão excede um limite especificado [8].

2.5.1 Normas de segurança

As normas de segurança para equipamentos médicos podem diferir de país para país. A organização atual que prepara e publica Normas Internacionais para todas as tecnologias elétricas, eletrônicas e relacionadas é a Comissão Eletrotécnica Internacional (IEC), localizada em Genebra, Suíça. Ela publica normas para segurança básica e desempenho essencial de ECG regulares (IEC 60601-2-25:2011, 2011), equipamentos de monitorização eletrocardiográfica (IEC 60601-2-27:2011, 2011) e sistemas eletrocardiográficos ambulatórios (IEC 60601-2-47:2012, 2012). O leitor é direcionado para estes documentos para requisitos detalhados de design seguro de sistemas de ECG [8].

2.6 Doenças cardiovasculares

As doenças cardiovasculares representam uma das principais causas de morbidade e mortalidade a nível global, sendo um desafio significativo para a saúde pública. Este grupo de patologias afeta o coração e os vasos sanguíneos, abrangendo condições como a hipertensão arterial, a doença arterial coronária, as arritmias, e a insuficiência cardíaca. Apesar dos avanços no diagnóstico e no tratamento, os fatores de risco como o sedentarismo, a má alimentação, o tabagismo e o *stress* continuam a contribuir para a elevada prevalência destas doenças. Neste subcapítulo vão ser exploradas as principais características, causas e impactos de algumas das doenças cardiovasculares, destacando ainda um exemplo de uma onda ECG de um paciente com cada uma das doenças apresentadas.

2.6.1 Arritmias

As arritmias são distúrbios no ritmo cardíaco, ocorrendo quando o coração bate de forma irregular, rápida ou lenta. O coração tem um sistema elétrico que controla o seu ritmo, e qualquer falha nesse sistema pode originar uma arritmia. Existem diversos tipos, como a fibrilação atrial (batimentos irregulares nos átrios), taquicardia

(batimentos rápidos), bradicardia (batimentos lentos) e extrasístoles (batimentos extras). Embora algumas sejam benignas, outras podem provocar sintomas como falta de ar, dor no peito ou desmaios, e, em casos mais graves, levar a complicações. O tratamento varia conforme o tipo e a severidade da arritmia, podendo incluir medicamentos, cardioversão ou procedimentos invasivos. Na Fig. 2.27 está apresentado um sinal ECG de um exemplo de uma doença que provoca arritmias.



Figura 2.27: Taquicardia Auricular Multifocal [8]

2.6.2 Distúrbios de Condução

Os distúrbios de condução cardíaca ocorrem quando há um problema no sistema elétrico que transmite os sinais do coração, afetando a forma como os impulsos elétricos são conduzidos. Esses distúrbios podem levar a um atraso ou bloqueio parcial ou total da condução elétrica entre as diferentes partes do coração. Existem vários tipos, como o bloqueio AV, que interfere na comunicação entre os átrios e os ventrículos, e o bloqueio de ramo, que afeta a condução nos ventrículos. Dependendo da gravidade, os distúrbios de condução podem causar sintomas como síncope, tontura ou fadiga, e, em casos mais graves, podem resultar em arritmias ou falência cardíaca. O tratamento pode incluir medicação, marcapasso ou, em situações mais críticas, intervenções cirúrgicas. Na Figura 2.28 está apresentado um sinal ECG de um exemplo de uma doença que provoca distúrbios de condução.

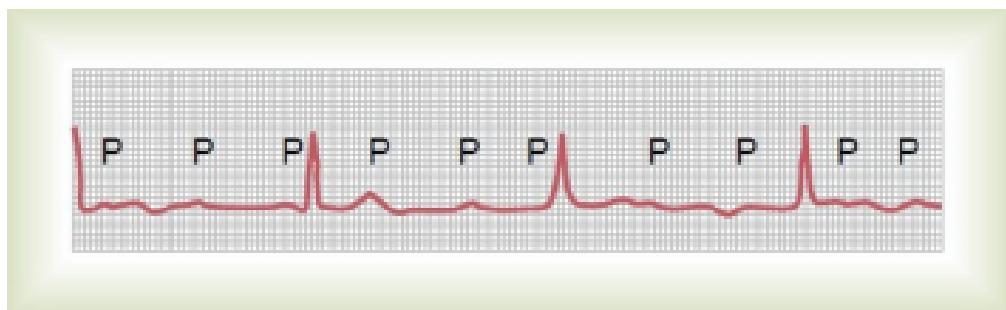


Figura 2.28: Bloqueio AV tipo III [8]

2.6.3 Hipertrofia

A hipertrofia cardíaca é o aumento anormal do tamanho das células musculares do coração, geralmente como resposta a sobrecarga de trabalho. Pode ocorrer nos ventrículos (hipertrofia ventricular) ou nos átrios (hipertrofia atrial). A hipertrofia ventricular esquerda, por exemplo, é frequentemente associada à hipertensão arterial ou doenças das válvulas cardíacas, enquanto a hipertrofia do ventrículo direito pode ser causada por doenças pulmonares crónicas. Essa condição pode afetar a capacidade do coração de bombear sangue de forma eficiente, levando a sintomas como falta de ar, fadiga, dor no peito ou até insuficiência cardíaca. O tratamento depende da causa subjacente e pode incluir medicação, controlo da pressão arterial, e, em casos mais graves, procedimentos cirúrgicos ou o uso de dispositivos como marcapassos. Na Figura 2.29 está apresentado um sinal ECG de um exemplo de uma doença que provoca hipertrofias.

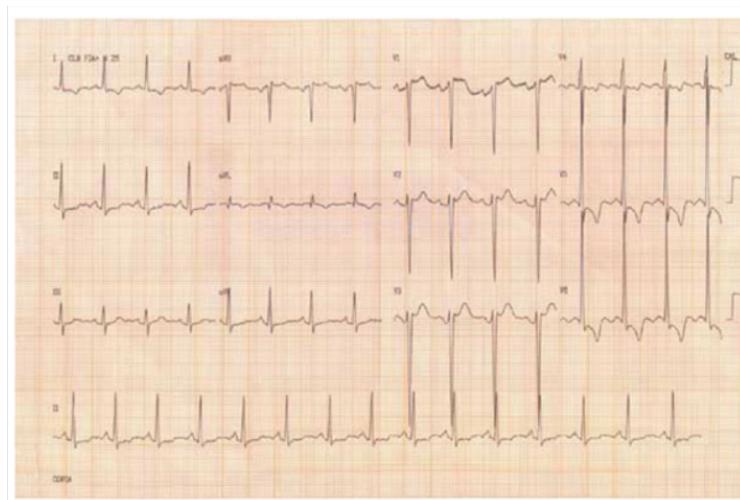


Figura 2.29: Hipertrofia Arterial [11]

2.6.4 Alterações de ST/T

As alterações de ST/T referem-se a alterações no ECG que envolvem a fase de repolarização do coração, especificamente no segmento ST e na onda T. O segmento ST representa a transição entre a despolarização e a repolarização dos ventrículos, enquanto a onda T reflete a repolarização dos ventrículos. Alterações nessas ondas podem indicar uma variedade de condições, como isquemia miocárdica (falta de oxigénio no músculo cardíaco), infarto do miocárdio, pericardite ou distúrbios eletrolíticos. As alterações mais comuns incluem elevações ou depressões do segmento ST e alterações na morfologia da onda T. Essas alterações podem ser sintomáticas, com queixas como dor no peito ou falta de ar, ou ser detectadas de forma incidental durante exames de rotina. O tratamento varia conforme a causa subjacente, podendo envolver medicamentos, mudanças no estilo de vida ou intervenções mais invasivas, como cateterismo cardíaco ou cirurgia. Na Figura 2.30 está apresentado um sinal ECG de um exemplo de uma doença que provoca alterações de ST/T.



Figura 2.30: Supradesnívelamento de ST [12]

2.6.5 ECG 200+

O ECG 200+, este dispositivo é um ECG de diagnóstico com 12 derivações simultâneas que exibe, adquire, imprime e armazena traçados de ECG para adultos e crianças [13].

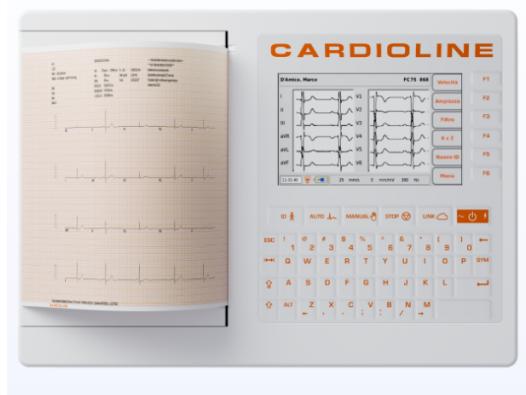


Figura 2.31: ECG 200+ [13]

Contém as seguintes características principais:

- 12 derivações simultâneas
- Exibe, adquire, imprime e armazena traçados de ECG
- Para uso em adultos e crianças
- 15 derivações disponíveis para impressão (12 + derivações de Frank)
- Calcula os principais parâmetros gerais do ECG
- Equipado com conectividade completa: *Universal Serial Bus* (USB), LAN e *Wireless Fidelity* (WiFi) (opcional)

Especificações Técnicas:

- Aquisição de ECG: 15 canais simultâneos
- ADC: 24 bits, 32000 amostras/segundo/canal
- Taxa de amostragem: 32000 amostras/segundo/canal para entrada, 1000 amostras/segundo/canal para análise
- Faixa dinâmica: +/- 400 mV
- Largura de banda: Equivalente a 0,05-300 Hz
- Detecção de marca-passo: Hardware com filtragem digital por convolução
- Proteção contra desfibrilação: Padrões *Association for the Advancement of Medical Instrumentation* (AAMI)/IEC
- Processamento e Análise
- Sistema operacional: Linux

- Medições de ECG: Inclui HR, intervalos PR/QT/QTc, duração QRS, eixos P/R/T, entre outros
- Interpretação de ECG: Programa de Análise de Glasgow para Adultos, Pediátrico e *ST Elevation Myocardial Infarction* (STEMI) (opcional)
- Filtros: Filtros digitais adaptativos de fase linear para interferência AC e passa-baixa

Interface e Armazenamento:

- Display: *Liquid-Crystal Display* (LCD) colorido de 7" com resolução 800x480
- Teclado: Alfanumérico completo
- Armazenamento: Interno para até 100 ECG (expansível para 1000)
- Impressora térmica: Resolução de 8 pontos/mm, vários formatos de impressão

Conectividade:

- USB e LAN (padrão)
- WiFi (opcional)
- Formatos de exportação: *Standard Communications Protocol for PDF* (SCP-PDF), *Extensible Markup Language - Device Data Format* (XML-GDT), *Digital Imaging and Communications in Medicine* (DICOM), *Health Level 7* (HL7)

Outras Características:

- Bateria recarregável com duração de mais de 500 ECG
- Classificação de proteção IP20 (*Ingress Protection* (IP))
- Conformidade com várias normas internacionais (*European Norm* (EN), *International Organization for Standardization* (ISO), IEC)
- Diversos acessórios e opções disponíveis

Tem um preço que ronda dos 1822,13 € [14].

2.6.6 Projeto *Do It Yourself* (DIY)

Existem diversos projetos similares ao que pretendemos executar sendo que muitos dos quais utilizam o AD8232. Este circuito integrado é usado principalmente para amplificar e filtrar sinais elétricos biológicos, como os do coração. Possui dois filtros, ambos com ganho de 100, um passa-alto e um passa-baixo, destinados a eliminar a

baseline drift e o *aliasing*. Mesmo assim, não dispensa o uso de filtros possíveis de serem implementados digitalmente. O artigo “ECG MONITORING SYSTEM USING AD8232 SENSOR” [15], é um excelente exemplo do objetivo principal pretendido com este protótipo. Apesar de acrescentar uma comunicação para um telemóvel através de uma tecnologia de comunicação sem fios, a extração de um ECG está patente no mesmo. Desta forma, é utilizado o AD8232 para extrair sinais de ECG e exibir os dados em tempo real em computador e transmiti-los para o telemóvel, com sistema operativo *android*, via *Bluetooth*. Esta vertente pode ser observada através do diagrama de blocos do sistema proposto, presente na figura 2.32.

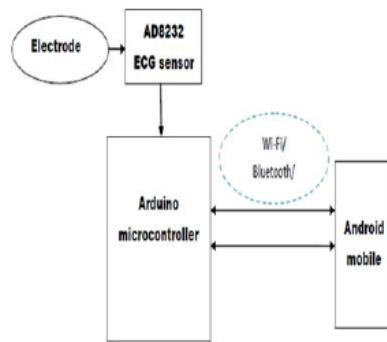


Figura 2.32: Arquitetura do projeto diy

Capítulo 3

Arquitetura do Projeto

Este capítulo aborda os princípios fundamentais para o projeto, como arquitetura, e um projeto para a implementação do problema, tanto a nível de *hardware* como *software*. Serão abordadas outras opções e demonstrado a melhor opção.

3.1 Descrição geral da arquitetura do projeto

Com este projeto é pretendido a aquisição de um ECG e tratamento da informação recolhida, de modo a aplicar um modelo treinado de inteligência artificial, de forma a detetar problemas observados no ECG adquirido.

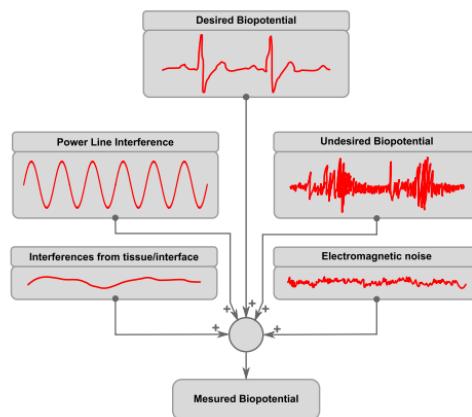


Figura 3.1: Componentes do potencial bioelétrico medido.[8]

Como já referido no capítulo 2, um ECG necessita de um tratamento especial, como pode ser observado na Figura 3.1. O potencial bioelétrico medido é a soma de diversos sinais, como o potencial bioelétrico desejado, a interferência da rede elétrica, os potenciais bioelétricos indesejados, como a atividade muscular ou de outros órgãos, interferências dos tecidos ou interface que resultam da interação entre os elétrodos e a pele, como deslocamentos dos elétrodos. Também as variações no contacto e mudanças na resistência elétrica da interface e ruído eletromagnético influenciam o potencial bioelétrico medido, sendo a interferência gerada por equipamentos eletrônicos próximos ou outras fontes de campos eletromagnéticos. Este sinal necessita de uma filtragem de modo a obter uma forma de onda mais limpa, ou seja, livre de ruídos. Como o sinal adquirido do ECG tem uma amplitude demasiado baixa, entre 0 e os 2.8mV, é necessário amplificá-las, sendo necessário um ganho na ordem dos 1000.

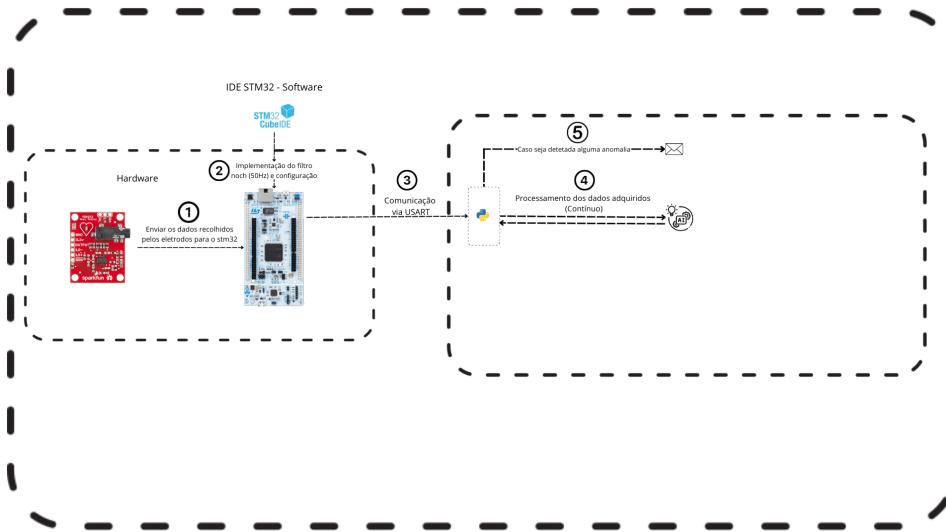


Figura 3.2: Arquitetura de software

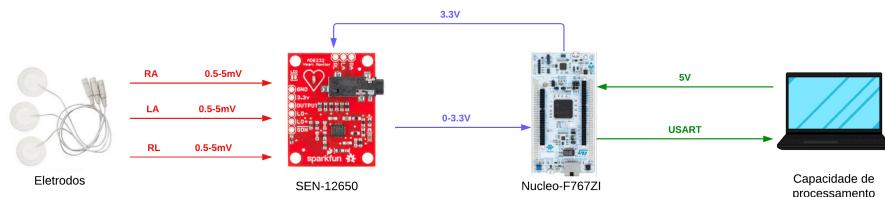


Figura 3.3: Diagrama de blocos

Como o objetivo é identificar, com o auxílio da inteligência artificial, incongruências nos ECGs, identificando assim os problemas dos pacientes, foi optado por uma abordagem mais digital e um maior poder de processamento. Como observado na Figura 3.2, será adotada uma aquisição do sinal pelo sen-12650, cujas especificações

serão detalhadas posteriormente, com um microcontrolador, núcleo STM32F767ZI, onde no mesmo será implementado um filtro notch para 50Hz. Este filtro deve ser implementado no núcleo, já que como irá ser abordado aquando da especificação do ad8232, este componente integrante do sen-12650, não implementa esse filtro.

Com isto, aquando da aquisição, o mesmo é transmitido via USART para o computador, via USB, e recorrendo a um método desenvolvido em *python*, para ler a porta do USART. As informações adquiridas serão processadas pelo modelo treinado de inteligência artificial, neste caso *Deep Learning*, mais concretamente o LSTM. Cada ECG adquirido terá um campo na interface gráfica, que caso haja alguma anomalia detetada pela IA será preenchido e será enviado um email, a indicar o problema detetado. Os problemas associados a determinado ECG e a própria monitorização do mesmo são algumas das informações visíveis. Tudo isto será possível através da interface gráfica desenvolvida em *python*. O sistema será alimentado a 5V, onde o Nucleo-F767ZI converte esta tensão, através de um módulo interno, para 3.3V que alimentará o SEN-12650. A tensão transmitida pelos elétrodos é de 0.5 a 5mV, o que pode ser observado na Figura 3.3.

3.2 Projeto e descrição do *Hardware*

Desta forma, de modo a proceder à implementação deste protótipo de eletrocardiógrafo. A identificação destes elementos já foi efetuada, concluindo agora com a sua apresentação.

3.2.1 Núcleo STM32F767ZI

O Nucleo-F767ZI, representado na Figura 3.4, é uma placa de desenvolvimento avançada baseada no microcontrolador STM32F767ZI, que oferece um conjunto abrangente de funcionalidades para desenvolvimento de aplicações embebidas de alto desempenho.



Figura 3.4: Núcleo F767ZI

No coração da placa encontra-se o microcontrolador STM32F767ZI, um *Acorn RISC Machine* (ARM) Cortex-M7 de 32 bits que opera a uma frequência máxima de 216 MHz. Este microcontrolador é equipado com 2 MB de memória *Flash* e 512 KB de *Static Random-Access Memory* (SRAM), proporcionando um amplo espaço para programas e dados.

A placa inclui um depurador/programador ST-LINK/V2-1 integrado, que facilita significativamente o processo de desenvolvimento e depuração. Possui também uma interface USB *On-The-Go* (OTG) de alta velocidade com conector micro-AB, permitindo que a placa funcione como *host* ou dispositivo USB.

Em termos de conectividade, o Nucleo-F767ZI oferece uma gama impressionante de opções. Inclui um conector Ethernet compatível com o padrão IEEE-802.3-2002 (*Institute of Electrical and Electronics Engineers* (IEEE)), possibilitando comunicações em rede. A placa também suporta múltiplos protocolos de comunicação série, incluindo USART, *Inter-Integrated Circuit* (I2C), *Serial Peripheral Interface* (SPI), *Controller Area Network* (CAN) e outros.

Para facilitar a prototipagem rápida, a placa incorpora conectores compatíveis com Arduino (ST Zio) e ST Morpho, permitindo o acesso a todas as entradas e saídas do STM32. Isto torna possível a utilização de uma vasta gama de *shields* e módulos de expansão.

A placa dispõe de três LEDs indicadores: um para alimentação, outro para comunicação USB e um *Light Emitting Diode* (LED) passível de ser programado pelo utilizador. Além disso, inclui botões de programação personalizável e de *reset*, bem como um oscilador externo com um cristal de 32,768 kHz para aplicações que requerem temporização precisa.

O Nucleo-F767ZI suporta várias opções de alimentação, incluindo através da porta USB do ST-LINK, via USB OTG ou através de uma fonte externa. Isto proporciona flexibilidade para diferentes cenários de utilização.

Em termos de capacidades de processamento, o microcontrolador STM32F7 incorpora o acelerador *Adaptive Real-Time* (ART) da ST e uma *cache* L1, permitindo a execução de código a partir da memória *Flash* ou externa sem penalizações de desempenho. Oferece também uma unidade de ponto flutuante de precisão dupla, útil para cálculos complexos.

A placa suporta o ambiente de desenvolvimento ARM mbed, facilitando o desenvolvimento de software. Além disso, é compatível com uma ampla gama de *Integrated Development Environment* (IDE), incluindo *Interactive Application Runtime* (IAR), Keil e ferramentas baseadas em GCC.

Finalmente, o Nucleo-F767ZI inclui recursos avançados como um controlador *Liquid Crystal Display - Thin-Film Transistor* (LCD-TFT) capaz de suportar resoluções até *Extended Graphics Array* (XGA), uma interface para câmera paralela, e um gerador de números aleatórios verdadeiros, tornando-o adequado para uma vasta

gama de aplicações, desde sistemas de controlo industrial até dispositivos *Internet of Things* (IoT) complexos.

Na Tabela 3.1 estão presentes as principais características deste componente.

Característica	Especificação
Microcontrolador	STM32F767ZI (ARM Cortex-M7)
Frequência Máxima	216 MHz
Memória Flash	2 MB
Memória SRAM	512 KB
Tensão de Operação	1.8V (VDD)
Pinos <i>General Purpose Input/Output</i> (GPIO)	Até 168
Timers	2x 32-bit, 13x 16-bit, 2x watchdog, 1x SysTick
ADC	3x 12-bit (até 24 canais)
<i>Digital-to-Analog Converter</i> (DAC)	2x 12-bit
UARTs/USARTs	4 <i>Universal Asynchronous Receiver/Transmitter</i> (UART), 4 USART
I2Cs	4
SPIs	6
CANs	2
Ethernet	10/100 Mbps, IEEE-802.3-2002 compatível
USB	OTG Full Speed e High Speed
Interfaces Adicionais	I2S (3), SAI (2), SPDIF (4), HDMI (1), Câmera
<i>Direct Memory Access</i> (DMA)	16 canais
<i>Real-Time Clock</i> (RTC)	1
<i>Random Number Generator</i> (RNG)	Gerador de Números Aleatórios Verdadeiros
Aceleradores	Chrom-ART, ART
<i>Floating Point Unit</i> (FPU)	Sim
Controller	Resolução XGA
Depurador/Programador	ST-LINK/V2-1 integrado
Conectores	ST Zio (compatível Arduino), ST Morpho
LEDs	3 (energia, comunicação USB, utilizador)
Botões	Usuário e reset
Oscilador	Cristal de 32.768 kHz

Tabela 3.1: Especificações do Nucleo-F767ZI [16]

3.2.2 Sen-12650

O SEN-12650, representado na Figura 3.5 é um módulo de pré-processamento de sinais biomédicos, especificamente projetado para medir a atividade elétrica do coração. Este dispositivo utiliza o circuito integrado AD8232, que funciona como um amplificador de instrumentação especializado para sinais biopotenciais.



Figura 3.5: SEN-12650

O módulo é alimentado por uma tensão de 3,3V e fornece uma saída analógica que pode ser facilmente interpretada por microcontroladores como o Arduino. Esta saída representa o sinal de ECG após amplificação e filtragem.

Uma das principais funcionalidades do SEN-12650 é a sua capacidade de extrair, amplificar e filtrar pequenos sinais biopotenciais em condições ruidosas, como

aquelas criadas por movimento ou pela colocação remota dos elétrodos. Isto torna o dispositivo particularmente útil para aplicações onde o ruído é um fator significativo.

O módulo inclui nove conexões externas que podem ser soldadas a pinos, fios ou outros conectores. Entre estas, destacam-se os pinos SDN (para desligar o dispositivo), LO+ e LO- (para deteção de elétrodo solto), OUTPUT (saída do sinal) e 3.3V e *Ground* (GND) (para alimentação). Adicionalmente, o módulo disponibiliza pinos *Right arm* (RA), *Left arm* (LA) e *Right leg* (RL) para a ligação de sensores personalizados.

Uma característica notável é o LED indicador incorporado, que pulsa no ritmo do batimento cardíaco, proporcionando uma indicação visual imediata da atividade cardíaca.

O SEN-12650 inclui um conector *jack* de 3,5mm para facilitar a ligação de sensores biomédicos. É importante notar que os sensores (elétrodos) não estão incluídos e devem ser adquiridos separadamente.

O dispositivo é projetado para facilitar a visualização dos intervalos PR e QT do ECG, que são importantes para a análise da atividade cardíaca. No entanto, é crucial salientar que este não é um dispositivo médico certificado e não deve ser utilizado para diagnóstico ou tratamento de condições médicas.

Em termos de compatibilidade, o SEN-12650 é projetado para funcionar bem com plataformas de desenvolvimento como o Arduino, tornando-o uma ferramenta valiosa para prototipagem rápida e projetos de eletrônica relacionados com a monitorização cardíaca.

Por fim, o módulo é compacto e inclui furos de montagem para uma fácil integração em projetos. A sua simplicidade de uso, combinada com a capacidade de lidar com sinais biopotenciais em ambientes ruidosos, torna o SEN-12650 uma opção atrativa para entusiastas, estudantes e investigadores interessados em monitorização cardíaca não invasiva.

Na Tabela 3.2 estão representadas as principais características do Sen-12650.

Característica	Especificação
Tipo de Dispositivo	Módulo de pré-processamento de sinais biomédicos (Front-end Analógico para ECG)
Chip Principal	AD8232
Aplicação Principal	Medição de <i>ECG</i>
Tensão de Alimentação	3.3V
Tipo de Saída	Analógica
Funcionalidades	Amplificador de instrumentação, filtro de sinal
Pinos Externos	<i>SDN</i> , <i>LO+</i> , <i>LO-</i> (Lead-off detection)
Indicador Visual	LED para ritmo cardíaco
Conector de Sensor	Jack de 3.5mm
Controle	Pino de desligamento (<i>SDN</i>)
Compatibilidade	Arduino e outras placas de desenvolvimento
Uso Médico	Não é um dispositivo médico (não para diagnóstico ou tratamento)
Características do AD8232	Amplificação e filtragem de sinais biopotenciais
Visualização	Facilita a visualização dos intervalos PR e QT
Desempenho em Ruído	Projetado para operar em condições ruidosas
Montagem	Pinos externos para solda
Consumo de Corrente	170 μ A (típico)
Ganho	100 V/V
CMRR	80 dB DC a 60 Hz
Largura de Banda	0.5 Hz a 40 Hz
Pinos de Saída	OUTPUT (analógico)
Filtros Integrados	Passa-alta e Passa-baixa
Dimensões	20 x 20 mm (aproximadamente)
Preço Aproximado (€)	11,69€

Tabela 3.2: Especificações do SEN-12650 (Monitor de Frequência Cardíaca AD8232)

AD8232

O AD8232, apresentado na Figura 3.6, é um circuito integrado especializado, projetado para aplicações de monitorização cardíaca e outras medições biopotenciais. Este *chip* é o componente central do módulo SEN-12650 e desempenha um papel crucial na aquisição e processamento de sinais de ECG. No Anexo C é possível analisar o esquemático pormenorizado do AD8232.

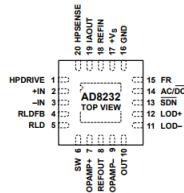


Figura 3.6: AD8232

Funcionando como um bloco de condicionamento de sinal integrado, o AD8232 é capaz de extrair, amplificar e filtrar pequenos sinais biopotenciais, mesmo em condições ruidosas. Esta capacidade é particularmente útil em situações onde há interferências causadas por movimento ou pela colocação remota de elétrodos.

O chip opera com uma tensão de alimentação entre 2,0V e 3,5V, tornando-o compatível com uma variedade de sistemas de baixa potência. O seu consumo de corrente é tipicamente de 170 μ A, o que o torna adequado para aplicações portáteis e alimentadas por bateria.

Uma das características mais notáveis do AD8232 é o seu ganho fixo de 100 V/V. Este ganho elevado permite a amplificação de sinais cardíacos muito fracos, tornando-os facilmente detetáveis e mensuráveis.

O AD8232 incorpora vários recursos para melhorar a qualidade do sinal. Possui uma CMRR de 80 dB (de DC a 60 Hz), o que ajuda a eliminar interferências comuns. A sua largura de banda típica vai de 0,5 Hz a 40 Hz, abrangendo a faixa de frequências mais relevante para sinais de ECG.

O *chip* inclui também uma funcionalidade de deteção de elétrodo solto, que é crucial para garantir medições precisas e confiáveis. Esta função é acessível através dos pinos LO+ e LO-, que podem ser monitorizados para verificar a integridade da conexão dos elétrodos.

Além disso, o AD8232 integra um amplificador operacional de uso geral, que pode ser utilizado para implementar filtros adicionais ou para criar um circuito de *drive* para o Amplificador de Perna Direita (RLD), comum em aplicações de ECG.

O *chip* possui uma função de restauração rápida, que ajuda a reduzir o tempo de recuperação após grandes perturbações no sinal, como aquelas causadas pelo uso de desfibrilhadores.

Em termos de robustez, o AD8232 pode operar numa ampla faixa de temperatura, de -40°C a +85°C, embora a sua faixa de temperatura nominal seja de 0°C a +70°C.

O AD8232 vem num encapsulamento LFCSP de 20 pinos, medindo apenas 4 mm x 4 mm, o que permite a sua integração em dispositivos compactos.

Na Figura 3.7 é possível observar o esquemático deste *Heart Rate Monitor Front End*, é possível observar que além das características já evidenciadas é possível desenvolver ambos os filtros passa-alto e passa-baixo. O filtro passa-alto está acoplado ao amplificador de instrumentação, enquanto que o passa-baixo utiliza um amplificador operacional.

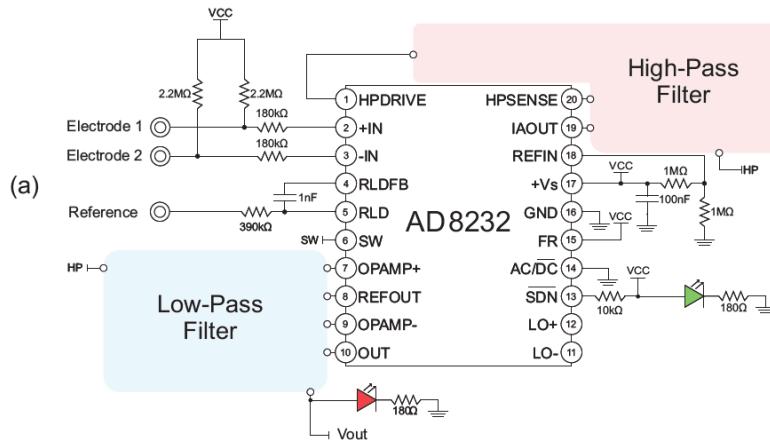


Figura 3.7: Pinout AD8232 [17]

Na Figura 3.8 está representada a implementação de um filtro passa-alto e passa-baixo, utilizando um amplificador de instrumentação de segunda ordem com *offset* na saída [17].

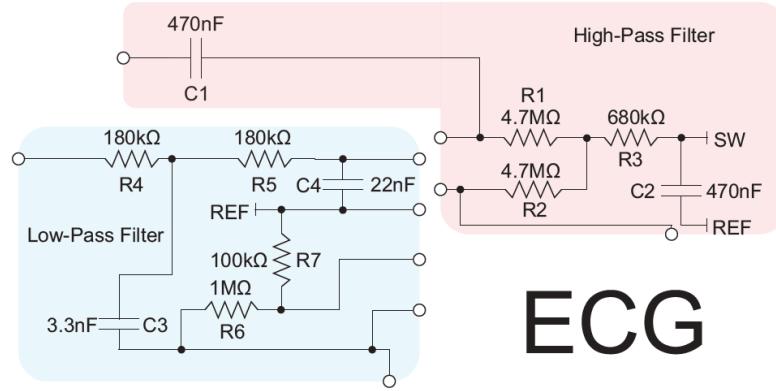


Figura 3.8: Implementação de filtros passa-alto e passa-baixo no AD8232

A frequência de corte do filtro passa-alto é determinada pela seguinte equação [17]:

$$f_{HP} = \frac{10}{2\pi\sqrt{R_1 R_2 C_1 C_2}} \quad (3.1)$$

Este filtro vai atenuar a interferência das baixas frequências devido a movimentos indesejados e potenciais de meia célula. Apesar disto, o AD8232 apresenta uma topologia de segunda ordem neste filtro que precisa de uma resistência de compensação para ajustar o filtro. Esta relação é dada por [17]:

$$R_3 = 0.14R_1 \quad (3.2)$$

O filtro passa-baixo de 2^a ordem pode ser projetado adicionando ao circuito o amplificador operacional. É utilizado para delimitar a banda dos potenciais indesejados e atenuar as altas frequências. A frequência de corte pode ser descrita como [17]:

$$f_{LP} = \frac{1}{2\pi\sqrt{R_4 R_5 C_3 C_4}} \quad (3.3)$$

Como o filtro está ativo, o ganho do circuito poderá ser ajustado pelas resistências R_6 e R_7 [17].

$$\text{Gain}_{LP} = 1 + \frac{R_6}{R_7} \quad (3.4)$$

De acordo com o fabricante, o ganho máximo é de 1100 (60.8dB). Assim, o ganho máximo para o filtro ativo que pode ser projetado é de 11 [17].

É importante notar que, apesar das suas capacidades avançadas de processamento de sinal para ECG, o AD8232 não é certificado como um dispositivo médico. É principalmente utilizado em aplicações de prototipagem, investigação e em produtos de bem-estar não médicos.

É assim possível concluir que o AD8232 é um componente versátil e poderoso para a aquisição de sinais biopotenciais, oferecendo uma solução integrada para muitos dos desafios associados à medição de ECG em ambientes não controlados.

A Tabela 3.3 apresenta as características principais do componente descrito.

Característica	Especificação
Tipo de Dispositivo	Bloco de condicionamento de sinal integrado para ECG
Aplicação Principal	ECG e outras medições biopotenciais
Tensão de Alimentação	2,0V a 3,5V
Consumo de Corrente	170 µA (típico)
Ganho	100 V/V (fixo)
CMRR	80 dB DC a 60 Hz
Largura de Banda	0,5 Hz a 40 Hz (típico)
Impedância de Entrada	5 GΩ
Rejeição de Modo Comum	-95 dB
Detecção de Eletrodo Solto	Sim (pinos LO+, LO-)
Amplificador Operacional Integrado	Sim
Filtros Integrados	Passa-alta e passa-baixa
Função de Restauração Rápida	Sim
RLD	Integrado
Faixa de Temperatura Operacional	-40°C a +85°C
Faixa de Temperatura Nominal	0°C a +70°C
Encapsulamento	LFCSP de 20 pinos
Dimensões do Chip	4 mm x 4 mm
Compatibilidade	Arduino e outras placas de desenvolvimento
Uso Médico	Não é um dispositivo médico certificado

Tabela 3.3: Especificações do Chip AD8232 [18]

3.2.3 Esquema elétrico

Na Figura 3.9 está apresentado o esquema elétrico simplificado que foi implementado, sendo que no Anexo B está apresentado o esquemático mais detalhado de todos os componentes elétricos envolvidos, nomeadamente a conexão entre o Núcleo-F767ZI e o integrado SEN-12650.

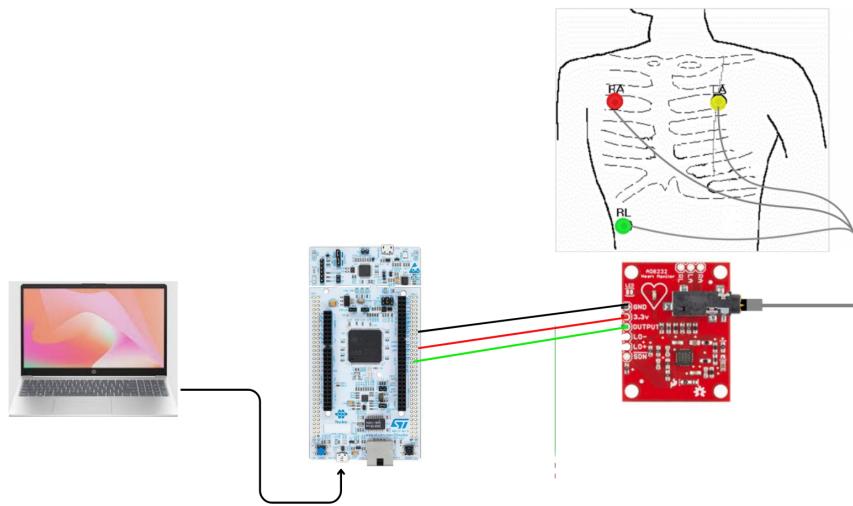


Figura 3.9: Esquema Elétrico

3.3 Projeto e descrição do Software

Depois de todos os elementos físicos apresentados, é também fundamental descrever todo o funcionamento do *software*. Este será responsável pela aquisição dos dados e o respetivo processamento dos mesmos.

3.3.1 Núcleo STM32F767ZI

Desta forma, o núcleo STM32F767ZI será o responsável pela aquisição do sinal proveniente do Sen-12650, respetiva conversão e filtragem do sinal. De forma resumida, este será a peça fundamental deste protótipo de forma a ser possível visualizar claramente um sinal de ECG.

Na Figura 3.10 é possível visualizar o fluxograma com todas as etapas referentes ao processo de recolha e tratamento dos dados por parte do STM32F767ZI.

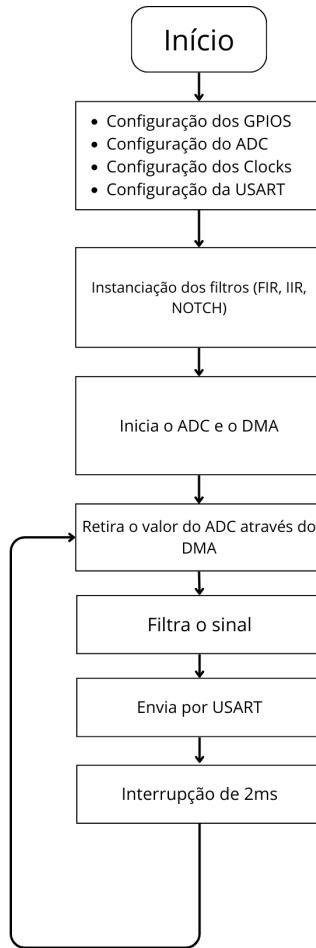


Figura 3.10: Fluxograma STM32F767ZI

3.3.2 Aplicação em *python*

Depois de todos os dados adquiridos e processados, os mesmos são enviados para o computador via USART. Assim, depois de recebidos, são mostrados na interface gráfica desenvolvida em *python*, onde será possível, além de visualizar a onda do ECG, ativar a utilização de inteligência artificial, enviar os dados obtidos por *e-mail* caso haja a deteção de alguma anomalia, visualizar os batimentos por segundo e determinar os meios pelas quais são enviados os dados como a porta COM e o *baud-rate*.

Nas Figuras 3.11 e 3.12 é possível visualizar, respetivamente, o fluxo de operações pretendido para a aplicação a desenvolver e um esboço da interface gráfica da mesma.

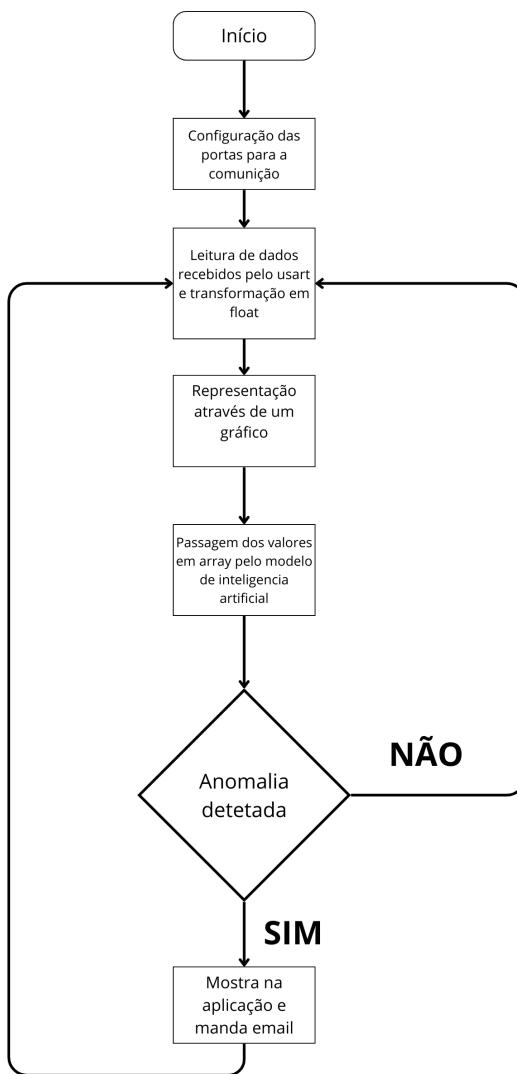


Figura 3.11: Fluxograma aplicação

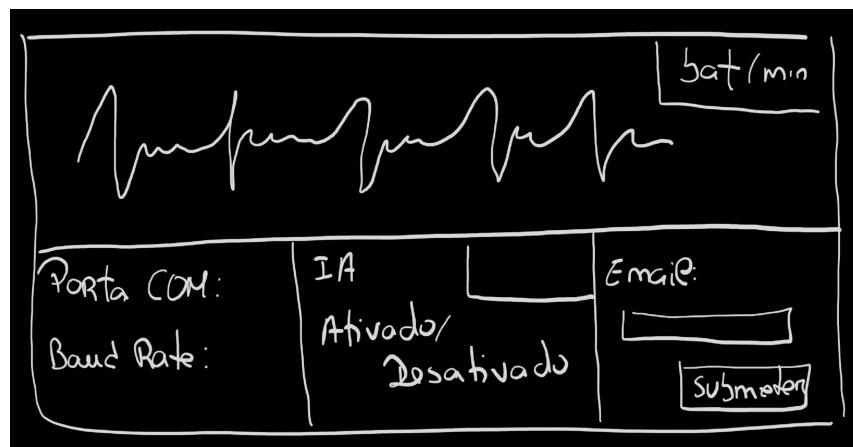


Figura 3.12: Mockup da aplicação

3.3.3 Relação entre núcleo e aplicação

É também importante referir de que forma irá acontecer a interligação entre o núcleo e a aplicação, quais serão as fases a implementar para a correta transferência dos dados.

Desta forma, é possível observar na Figura 3.13 o fluxo de dados e respetivos periféricos utilizados para a transmissão dos dados.

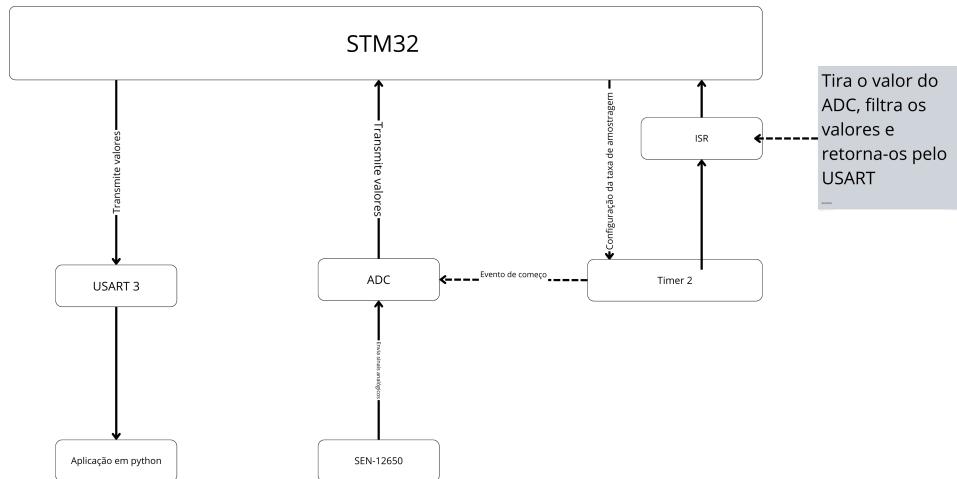


Figura 3.13: Interligação do núcleo e aplicação

3.3.4 Inteligência Artificial

A inteligência artificial nasceu na década de 1950, sendo considerada um campo geral que abrange aprendizagem automática e profunda [19]. Podemos definir a inteligência artificial como a capacidade que uma máquina tem para executar atividades humanas, tanto em raciocínio como em aprendizagem [20]. Existem diferentes subáreas dentro da inteligência artificial, como a *machine learning* e a *deep learning*, como é possível observar na figura 3.14. A *machine learning* foca no desenvolvimento de sistemas capazes de aprender a partir de dados, e fazer previsões ou decisões sem serem explicitamente programados. Dentro desta subárea existe o *deep learning* que se destaca pela capacidade de processar grandes quantidades de dados com arquiteturas de redes neurais complexas.

Desta forma, o *deep learning* é uma técnica de *machine learning* que utiliza as redes neurais complexas, que estas são compostas por camadas de nós, ou neurônios, que simulam o funcionamento do cérebro humano, permitindo que a máquina aprenda através de exemplos fornecidos [21].

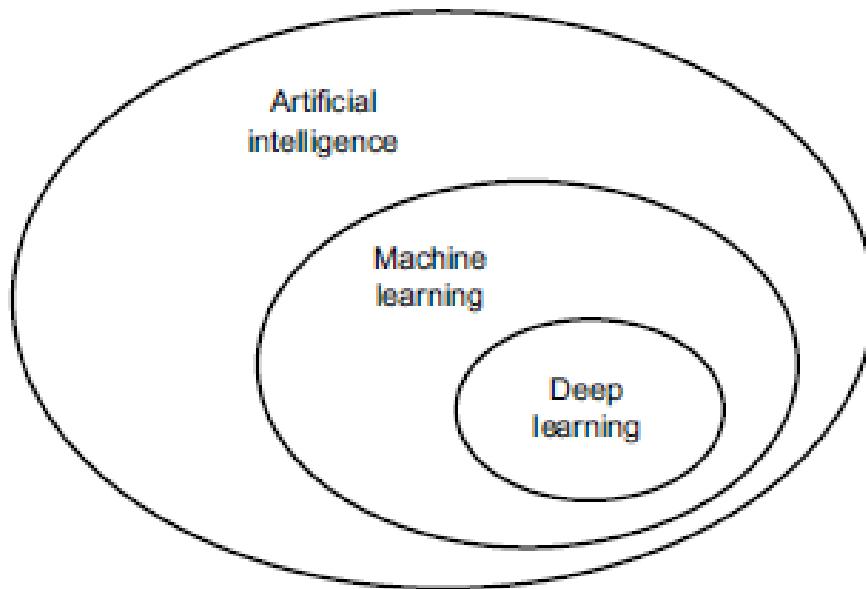


Figura 3.14: Diferentes subáreas da inteligência artificial

Dentro da *deep learning* existem diferentes modelos, como por exemplo as *Convolutional Neural Networks* (CNN), que são poderosas para análise de imagens. As CNN são projetadas para reconhecer padrões visuais diretamente dos *pixels* de imagens com eficiência, utilizando camadas que automaticamente e iterativamente aprendem os níveis de representação de dados adequados para tarefas como classificação de imagens. Estas redes destacam-se pela sua habilidade em identificar padrões invariantes à translação, escala e rotação, tornando-as ideais para tarefas como o reconhecimento facial.

Existem também as Redes Neurais Recorrentes (RNN) que são mais adequadas para dados sequenciais, como texto ou séries temporais. As RNN têm a capacidade única de manter um estado ou memória dos *inputs* anteriores, utilizando *loops* de *feedback* na sua estrutura, o que as torna adequadas para entender contextos e sequências de informações. Contudo, as RNN enfrentam desafios como o problema do desvanecimento do gradiente, que dificulta a aprendizagem em sequências muito longas.

De modo a superar as limitações das RNN, foram desenvolvidas as LSTM. Introduzidas por Hochreiter e Schmidhuber em 1997, as LSTM são uma extensão das RNN que incorporam portas de esquecimento e entradas e saídas para controlar o fluxo de informações. Isso permite que as LSTM retenham informações por períodos mais longos e sejam mais eficazes no tratamento de sequências longas sem perder a relevância de informações passadas [21].

Existem várias formas de treino/aprendizagem de um sistema computacional, através de um modelo.

Será utilizada a subárea *deep learning* com o modelo de RNN, visto possuirmos uma série temporal de dados, fornecido pelo *dataset MIT-BIH Arrhythmia*, onde serão efetuados testes de modo a perceber como o modelo responde a diferentes *inputs* e a diferentes tipos de arquitetura do modelo.

3.4 Procedimentos e vetores de Teste

De forma a testar se os valores obtidos são os corretos, serão utilizados dois vetores de teste.

O primeiro será utilizado numa altura inicial. Um osciloscópio será utilizado diretamente na saída do Sen-12650, de forma a verificar se a forma de onda visualizada na aplicação é a mesma que sai diretamente aquando da aquisição inicial.

Um outro método de teste será a utilização de um oxímetro para confirmar se o valor dos batimentos cardíacos detetado pela aplicação corresponde ou não à realidade.

Em termos de inteligência artificial será utilizada a matriz de confusão, onde dados nunca vistos pelo modelo serão avaliados pelo mesmo e como resultado será obtida uma diagonal com quantos dados foram identificados corretamente pelo modelo.

Existirão também outros indicadores como *accuracy*, *f1-score*, *precision* e *recall*. Cada um destes indicadores é calculado da seguinte forma:

- ***Accuracy Rate (Taxa de Precisão)***: Mede a proporção de previsões corretas relativamente ao total de amostras avaliadas. É definida como:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}.$$

- ***Recall (Sensibilidade ou Taxa de Verdadeiros Positivos)***: Mede a capacidade do modelo em identificar corretamente as amostras positivas. É definida como:

$$\text{Recall} = \frac{TP}{TP + FN}.$$

- ***Precision (Precisão)***: Mede a proporção de previsões positivas corretas relativamente ao total de previsões positivas feitas pelo modelo. É definida como:

$$\text{Precision} = \frac{TP}{TP + FP}.$$

- **F1-Score:** Combina a precisão e o *recall* numa única métrica, representando a sua média harmónica. É definida como:

$$F1 = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall}.$$

3.5 Lista de Componentes e custo de implementação

Para a realização deste protótipo foram utilizados elementos que não estando disponíveis no laboratório, tiveram de ser adquiridos externamente. Na Tabela 3.4 estão especificados os componentes adquiridos e o seu respetivo valor.

Componente	Referência	Quantidade	Valor Unitário (€)	Valor Total (€)
Microcontrolador Nucleo-F767ZI	NUCLEO-144 STM32F767ZI EVAL BRD	1	22,37	22,37
Front end SEN-12650	SPARKFUN SINGLE LEAD HEART RATE	1	11,69	11,69
Total:				34,06

Tabela 3.4: Tabela de componentes e valores

Capítulo 4

Protótipo funcional

Neste capítulo é apresentado o protótipo funcional, assim como a descrição da sua implementação, tendo como base o projeto descrito no capítulo anterior. Em relação a esse capítulo foram realizados alguns melhoramentos, contudo realizou-se um processamento digital de sinal, com a implementação de um filtro IIR e *Notch*, de modo a remover a interferência do sinal e as frequências referentes à corrente elétrica.

4.1 Programação do STM32

A programação do microcontrolador STM32F767ZI foi realizada em C, através do ambiente de desenvolvimento integrado da STMicroelectronics. Para a configuração do mesmo, foi necessário recorrer ao *datasheet* da placa de desenvolvimento. O *datasheet* foi também utilizado para proceder à ligação entre o SEN-12650 com o núcleo. Só assim foi possível entender os pinos do ADC e configura-lo para funcionar através do DMA, disponíveis na mesma. Com isto foram configuradas as portas do USART, configurado para trabalhar também com o DMA, de modo a poupar espaço para o processamento da informação, deixando assim espaço para a execução dos filtros, explicado mais à frente, onde de modo assíncrono se transmite a informação do ADC.

Como representado na Figura 3.10 a implementação em C, incidiu na configuração dos *clocks* e de um *timer* para fazer a amostragem a 500 Hz, de modo a obtermos

um eletrocardiograma, onde as ondas são facilmente visualizadas e qualquer deformação da mesma seja notada. Com isto, foi necessário configurar o *timer*, para que a cada 2 ms o mesmo acione uma interrupção.

A fórmula para calcular a frequência de *update* do *timer* é:

$$f_{\text{timer}} = \frac{f_{APB2}}{(Prescaler + 1) \times (Period + 1)}$$

onde f_{timer} é a frequência com que o timer reinicia após alcançar o valor do período.

Foi escolhido um *prescaler* de 215 para reduzir a frequência para aproximadamente 500Hz:

$$f_{\text{new}} = \frac{108000000}{(215 + 1) * (999 + 1)} \approx 500 \text{ Hz}$$

Para alcançar uma frequência de *update* de 500 Hz, o período deve ser:

$$\text{Period} = \frac{108000000}{108000} - 1 = 999$$

Essa configuração permite que o TIM1 gere uma interrupção a cada 2 ms, equivalente a uma frequência de amostragem de 500 Hz.

Na Listagem 4.1 é possível confirmar as várias características para a correta configuração do *timer*.

```

1 htim2.Instance = TIM2;
2 htim2.Init.Prescaler = 215;
3 htim2.Init.CounterMode = TIM_COUNTERMODE_UP;
4 htim2.Init.Period = 999;
5 htim2.Init.ClockDivision = TIM_CLOCKDIVISION_DIV1;
6 htim2.Init.AutoReloadPreload = TIM_AUTORELOAD_PRELOAD_DISABLE;

```

Listagem 4.1: Configuração do *timer*

Depois de configuradas as interrupções, é necessário configurar o ADC. Desta forma, foi implementado o DMA em modo circular de forma a não utilizar demais recursos do *Central Processing Unit* (CPU). Assim, a cada interrupção é obtido o valor do ADC através do DMA, sendo enviado para o processamento de sinal. Após o processamento vai ser enviado o valor final através da USART.

Na Listagem 4.2 é possível visualizar as características de configuração do ADC.

```

1 hadc1.Instance = ADC1;
2 hadc1.Init.ClockPrescaler = ADC_CLOCK_SYNC_PCLK_DIV4;
3 hadc1.Init.Resolution = ADC_RESOLUTION_12B;
4 hadc1.Init.ScanConvMode = ADC_SCAN_ENABLE;
5 hadc1.Init.ContinuousConvMode = ENABLE;
6 hadc1.Init.DiscontinuousConvMode = DISABLE;

```

```

7  hadc1.Init.ExternalTrigConvEdge = ADC_EXTERNALTRIGCONVEDGE_RISING;
8  hadc1.Init.ExternalTrigConv = ADC_EXTERNALTRIGCONV_T2_TRGO;
9  hadc1.Init.DataAlign = ADC_DATAALIGN_RIGHT;
10 hadc1.Init.NbrOfConversion = 1;
11 hadc1.Init.DMAContinuousRequests = ENABLE;
12 hadc1.Init.EOCSelection = ADC_EOC_SINGLE_CONV;
13 if (HAL_ADC_Init(&hadc1) != HAL_OK)
14 {
15     Error_Handler();
16 }
17
18 /** Configure for the selected ADC regular channel its
19    corresponding rank in the sequencer and its sample time.
20 */
21 sConfig.Channel = ADC_CHANNEL_9;
22 sConfig.Rank = ADC_REGULAR_RANK_1;
23 sConfig.SamplingTime = ADC_SAMPLETIME_480CYCLES;
24 if (HAL_ADC_ConfigChannel(&hadc1, &sConfig) != HAL_OK)
25 {
26     Error_Handler();
27 }
```

Listagem 4.2: Configuração do ADC

4.1.1 USART

A USART é uma interface de comunicação que permite a transmissão de dados tanto de forma síncrona como assíncrona entre o microcontrolador e outros dispositivos, como computadores. Foi implementado de forma a que não seja gasto recursos do CPU do STM32, utilizando o DMA. Para a implementação do mesmo foi necessário ativar os GPIO correspondentes do mesmo, neste caso os GPIO's de TX e RX, PB10 e PB11, respetivamente. Foi configurado para uma comunicação rápida, com 115200 bps (bits por segundo), taxa em que as informações são transferidas, com um comprimento de palavra de 8 bits, o que é suficiente para passar os valores do ADC.

Como é pretendida uma taxa de amostragem de 500Hz, será necessário calcular o *baud rate* mínimo necessário, de modo a não ocorrer perda de dados nem erros de transmissão.

Na seguinte equação é enunciado o raciocínio de cálculo para a obtenção do valor de *baud rate*.

$$\text{Baud Rate necessário} = \text{Frequência de quadros} \times \text{Total de bits por quadro} \quad (4.1)$$

$$\text{Baud Rate necessário} = 500 \times 110\text{bits/amostras} = 55000\text{bps} \quad (4.2)$$

De acordo com esta informação foi definido um *baud rate* de 115200 bps.

A configuração do USART utilizado é apresentada na Listagem 4.3.

```

1  GPIO_InitStruct.Pin = GPIO_PIN_10|GPIO_PIN_11;
2  GPIO_InitStruct.Mode = GPIO_MODE_AF_PP;
3  GPIO_InitStruct.Pull = GPIO_NOPULL;
4  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_VERY_HIGH;
5  GPIO_InitStruct.Alternate = GPIO_AF7_USART3;
6
7
8
9
10 huart3.Instance = USART3;
11 huart3.Init.BaudRate = 115200;
12 huart3.Init.WordLength = UART_WORDLENGTH_8B;
13 huart3.Init.StopBits = UART_STOPBITS_1;
14 huart3.Init.Parity = UART_PARITY_NONE;
15 huart3.Init.Mode = UART_MODE_TX_RX;
16 huart3.Init.HwFlowCtl = UART_HWCONTROL_NONE;
17 huart3.Init.OverSampling = UART_OVERSAMPLING_16;
18 huart3.Init.OneBitSampling = UART_ONE_BIT_SAMPLE_DISABLE;
19 huart3.AdvancedInit.AdvFeatureInit = UART_ADVFEATURE_NO_INIT;
20
21
22 hdma_usart3_tx.Instance = DMA1_Stream3;
23     hdma_usart3_tx.Init.Channel = DMA_CHANNEL_4;
24     hdma_usart3_tx.Init.Direction = DMA_MEMORY_TO_PERIPH;
25     hdma_usart3_tx.InitPeriphInc = DMA_PINC_DISABLE;
26     hdma_usart3_tx.InitMemInc = DMA_MINC_ENABLE;
27     hdma_usart3_tx.InitPeriphDataAlignment = DMA_PDATAALIGN_BYTE;
28     hdma_usart3_tx.InitMemDataAlignment = DMA_MDATAALIGN_BYTE;
29     hdma_usart3_tx.Init.Mode = DMA_NORMAL;
30     hdma_usart3_tx.Init.Priority = DMA_PRIORITY_LOW;
31     hdma_usart3_tx.Init.FIFOMode = DMA_FIFOMODE_DISABLE;

```

Listagem 4.3: Configuração do USART

4.2 Filtros digitais

Para a implementação dos filtros digitais, foi necessária uma pesquisa e um estudo para determinar o modo mais eficaz de implementar os diversos filtros com os respetivos parâmetros corretamente. Deste modo, foram seguidos os seguintes passos para cada um dos filtros.

4.2.1 Bliblioteca CMSIS-DSP

Primeiramente, de modo a realizar um processamento digital mais eficiente do sinal obtido, foi necessária a utilização da biblioteca CMSIS-DSP. Esta biblioteca está destinada a dispositivos com processadores Cortex-M e Cortex-A.

De um modo geral, esta biblioteca é capaz de implementar equações matemáticas básicas e avançadas, funções de filtragem e transformadas. Com isto, de modo a implementar filtros como IIR e FIR, foi utilizada esta mesma biblioteca.

Filtro FIR

O algoritmo do filtro FIR é baseado numa sequência de operações de Multiplicação e Acumulação (MAC). Cada coeficiente é multiplicado por uma variável estável que é igual a uma amostra da entrada anterior. A seguinte equação e a Figura 4.1 demonstram esta característica do filtro em questão [22].

$$y[n] = b[0]*x[n]+b[1]*x[n-1]+b[2]*x[n-2]+\dots+b[numTaps-1]*x[n-numTaps+1] \quad (4.3)$$

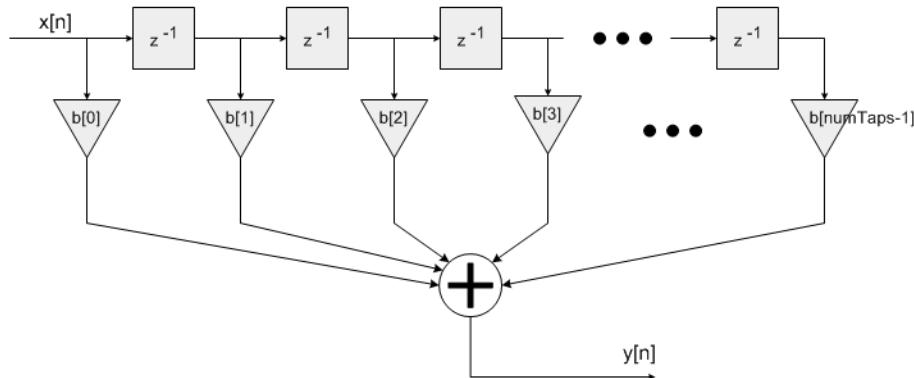


Figura 4.1: Diagrama de blocos filtro FIR

pCoeffs representa um *array* de coeficientes cujo tamanho é o valor de numTaps. Estes coeficientes são armazenados de forma invertida em relação ao seu valor temporal.

$$b[numTaps - 1], b[numTaps - 2], b[N - 2], \dots, b[1], b[0] \quad (4.4)$$

pState aponta para um *array* de estados cujo tamanho é numTaps+blockSize-1. As amostras presentes no *buffer* de estados são armazenados na seguinte ordem.

$$\begin{aligned}
 & x[n - numTaps + 1], x[n - numTaps], x[n - numTaps - 1], x[n - numTaps - 2] \dots \\
 & \dots x[n](== pSrc[0]), x[n+1](== pSrc[1]), \dots, x[n+blockSize-1](== pSrc[blockSize-1])
 \end{aligned} \tag{4.5}$$

É de notar que o comprimento do *buffer* de estados é superior ao do *array* de coeficientes. O aumento do valor do comprimento deste *buffer* de estados permite o endereçamento circular, que é tradicionalmente utilizado em filtros FIR, seja evitado e permita uma melhoria significativa na velocidade. As variáveis de estado são atualizadas depois de cada bloco de dados ser processado, sendo que os coeficientes permanecem intactos.

Inicialização da versão *Helium*

Na versão *Helium* o *array* de coeficientes deve ser preenchido com zeros para obter o número total de faixas de processamento. O comprimento L do *array* deve ser multiplicado por x ($L = x \times a$). Neste caso, para o f32, o valor de x é 4.

Os coeficientes adicionais ($x \times a - numTaps$) devem ser definidos como 0. O *numTaps* continua definido com o seu valor correto na função *init*. Isto significa que a implementação poderá necessitar da leitura de mais coeficientes devido à vetorização e para evitar a gestão de vários casos diferentes no código.

Helium state buffer

O estado do *buffer* deverá conter informação adicional temporária utilizada durante a computação, mas que não representa o estado do FIR. As primeiras A amostras são dados temporários. As restantes amostras são o estado do FIR. Desta forma, o *state buffer* tem como comprimento o valor $numTaps + A + blockSize - 1$. O A é o *blocksize* para o f32.

Comportamento do *Fixed-Point*

Deve-se ter muito cuidado com a utilização das versões dos *fixed-points* das funções do filtro FIR. De forma particular, o *overflow* e a saturação do acumulador utilizados em cada função devem ser levados em consideração [22].

Filtro IIR

Este conjunto de funções de filtros recursivos de ordem arbitrária (IIR). Os filtros são implementados como uma cascata de secções *Biquad* de segunda ordem. As funções suportam os tipos de dados Q15, Q31 e *floating-point*.

As funções operam sobre blocos de dados de *input* e *output* e cada chamada à função processa amostras *blockSize* através do filtro. Neste caso, pSrc aponta para o *array* de dados de *input* e pDst aponta para o *array* de dados de *output*.

Cada estado *Biquad* implementa um filtro de segunda ordem utilizando a seguinte equação diferencial:

$$y[n] = b0 * x[n] + b1 * x[n - 1] + b2 * x[n - 2] + a1 * y[n - 1] + a2 * y[n - 2] \quad (4.6)$$

O algoritmo *Direct Form I* é utilizado com 5 coeficientes e 4 variáveis de estado por fase. Na Figura 4.2 está representado o Diagrama de Blocos referente a este algoritmo.

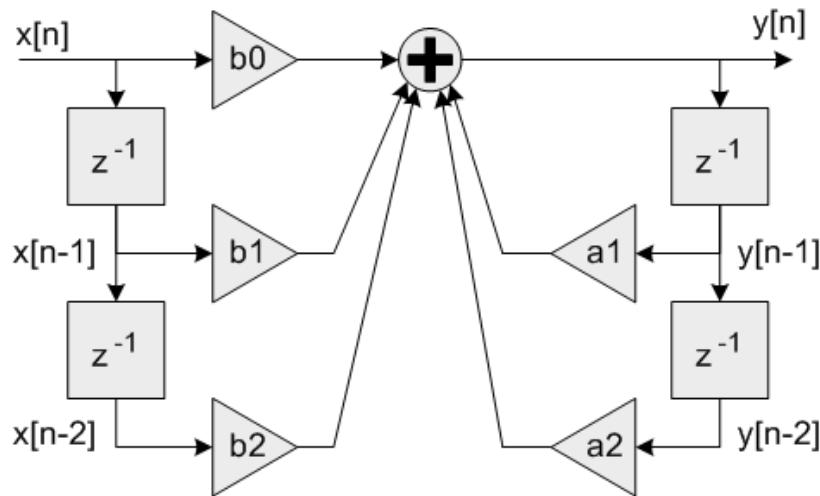


Figura 4.2: Diagrama de Blocos Filtro IIR

Filtro *Biquad* único

Os coeficientes b_0 , b_1 e b_2 multiplicam o sinal de *input* $x[n]$ e são referidos como os coeficientes *feedforward*. Os coeficientes a_1 e a_2 multiplicam o sinal de *output* $y[n]$ e são referidos como coeficientes de *feedback*. Algumas ferramentas de design utilizam a seguinte equação diferencial:

$$y[n] = b0 * x[n] + b1 * x[n - 1] + b2 * x[n - 2] - a1 * y[n - 1] - a2 * y[n - 2] \quad (4.7)$$

Neste caso, os coeficientes de *feedback*, a_1 e a_2 , devem ser negados quando usados na biblioteca CMSIS DSP. Filtros de ordens mais altas são implementados como uma cascata de secções de segunda ordem. *numStages* representa o número de fases de segunda ordem utilizadas.

Comportamento *Fixed-Point*

Deverá ter-se cuidado aquando do uso das versões Q15 e Q31 das funções do filtro em cascata *Biquad*. Questões como a escala dos coeficientes, o ganho do filtro, o *overflow* e a saturação devem ser tomadas em consideração.

Escala dos coeficientes

Os coeficientes dos filtros são representados como valores fracionários restritos ao intervalo $[-1 + 1)$. As funções *fixed-point* contêm um parâmetro de escala adicional, *postShift*, que permite que os coeficientes do filtro excedam o intervalo estipulado.

Na saída do acumulador do filtro está presente um *shift register* que desloca o resultado em bits *postShift*, como pode ser observado na figura 4.3.

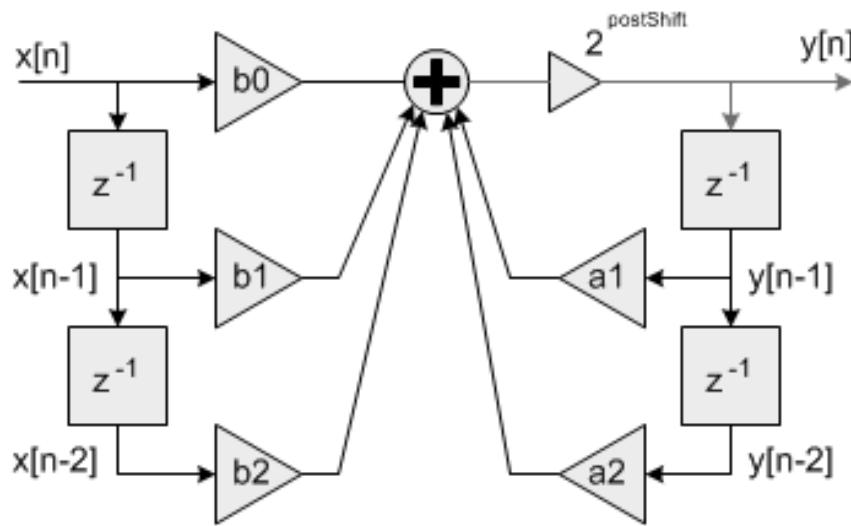


Figura 4.3: Escala dos coeficientes

Fixed-point Biquad com deslocamento através de bits *postShift* depois do acumulador

Essencialmente, é possível escalar os coeficientes do filtro por $2^{\text{postShift}}$. Por exemplo, para obter os coeficientes $(1.5, -0.8, 1.2, 1.6, -0.9)$, apenas é necessário definir o array *pCoeffs* como $(0.75, -0.4, 0.6, 0.8, -0.45)$ e o *postShift* = 1.

Ganho do filtro

A resposta em frequência de um filtro *Biquad* é determinada em função dos seus coeficientes. É possível que o valor do ganho do filtro exceda 1.0, o que significa que o filtro aumenta a amplitude de certas frequências.

Desta forma, um sinal de *input* com amplitude menor que 1.0 resulta num *output* maior que 1.0, sendo que estas ficarão saturadas ou em *overflow* dependendo da implementação do filtro. Para evitar este tipo de comportamentos, o filtro necessita de ser reduzido para que o pico do ganho seja menor que 1.0 ou o sinal de *input*

deve ser reduzido para que a combinação do *input* e do filtro nunca ultrapasse o seu valor máximo.

Irá ajudar a implementar os filtros que irão ser descritos posteriormente de uma forma mais eficaz [22].

4.2.2 Filtro FIR

Os filtros FIR, são normalmente utilizados em aplicações onde a fase linear é importante. São utilizados em aplicações cujo objetivo é melhorar sinais de áudio e biomédicos. A sua estrutura garante que nunca se tornem instáveis para qualquer tipo de sinal de entrada [23].

Passa-Baixo

- **Frequências de corte:** $f_c = 200$ Hz: para eliminar ruídos de alta frequência, ou seja, *aliasing*.
- **Taxa de amostragem (f_s):** definida conforme o sistema de aquisição de dados, $f_s = 500$ Hz.

De acordo com o Teorema de Nyquist a frequência de corte normalizada deverá ser metade da taxa de amostragem, neste caso 250 Hz. Este filtro terá a principal função de remover o *aliasing*, o que será implementado um corte a 200 Hz, sendo que as frequências cardíacas significativas andam abaixo de 150 Hz. De forma a posteriormente implementar este filtro no STM32, recorreu-se ao MATLAB retirando uma *sample* do ECG, de modo a aplicar o filtro e obter os coeficientes que são necessários para a implementação deste mesmo filtro no STM32.

Na Listagem 4.4 está representado o código usado para a implementação do filtro FIR passa-baixo no MATLAB.

```

1 % Configuraões Iniciais
2 fs = 500;      % Frequencia de amostragem
3 t_original = (0:length(dataStore)-1) / fs;    % Vetor de tempo para
plotagem
4
5 % Filtro FIR Passa-Baixo de 200 Hz
6 ordem_lp = 40;          % Ordem do filtro
7 f_nyquist = fs / 2;    % Frequencia de Nyquist
8 lp_fir = fir1(ordem_lp, 200 / f_nyquist, 'low');  % Design do
filtro com frequencia normalizada
9 data_low_passed = filter(lp_fir, 1, dataStore);    % Aplicacao do
filtro
10 fprintf('Filtro FIR Passa-Baixo\nNumero de Taps: %d\nCoeficientes:
%s\n\n', length(lp_fir), mat2str(lp_fir));
11 [h_lp, w_lp] = freqz(lp_fir, 1, 1024, fs);

```

```
12 figure;
13 subplot(3,1,1); % Adiciona um terceiro grafico
14 plot(dataStore, 'b');
15 hold on;
16 plot(data_low_passed, 'r');
17 title('Dados Originais vs. FIR Passa-Baixo');
18 xlabel('Amostras');
19 ylabel('Valor');
20 legend('Dados Originais', 'FIR Passa-Baixo');
21 subplot(3,1,2);
22 plot(w_lp, 20*log10(abs(h_lp)));
23 title('Resposta em Magnitude - FIR Passa-Baixo');
24 xlabel('Frequencia (Hz)');
25 ylabel('Magnitude (dB)');
26 subplot(3,1,3);
27 plot(w_lp, angle(h_lp)*180/pi);
28 title('Resposta de Fase - FIR Passa-Baixo');
29 xlabel('Frequencia (Hz)');
30 ylabel('Fase (graus)');
```

Listagem 4.4: Filtro FIR passa-baixo

Na Figura 4.4 está representado graficamente a resposta ao filtro FIR passa-baixo e a resposta em amplitude e fase, respetivamente.

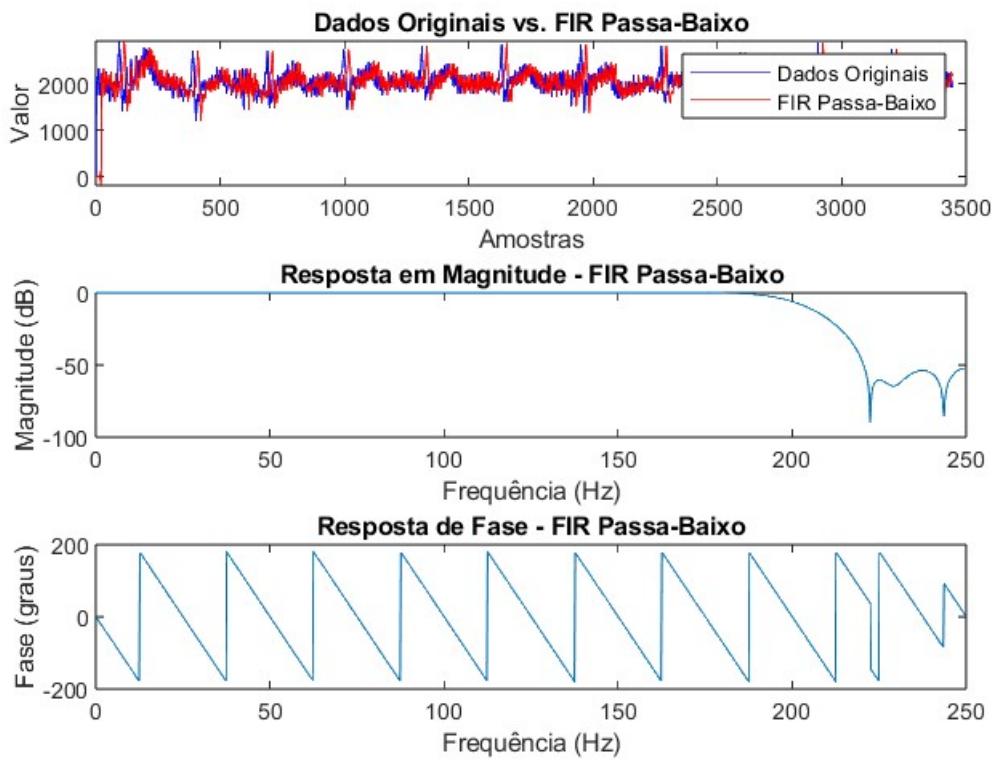


Figura 4.4: Resposta ao filtro FIR passa-baixo

De acordo com as informações geradas através do MATLAB, procedeu-se à respectiva implementação do filtro no STM32. Assim, a Listagem 4.5 representa os coeficientes necessários para a aplicação do filtro em STM32.

```

1 #define BLOCK_SIZE 1      // Processa um dado por vez na
                           // interrupcao
2 #define NUM_TAPS_LP 41    // Numero de taps no FIR Passa-Baixo
3 float32_t firStateLP[NUM_TAPS_LP + BLOCK_SIZE - 1];

```

```

4 float32_t firCoeffsLP[NUM_TAPS_LP] = { -2.49582062811498e-18 ,
-0.000843883995071432 , 0.00172480322214108 ,
-0.00231835925473622 , 0.00196357837057156 , -6.69912201787522e
-18 , -0.0036046519659237 , 0.0077148790607059 ,
-0.010040824322041 , 0.00796400731979479 , -1.68467892397761e
-17 , -0.0127268712987161 , 0.025823635119039 ,
-0.0323979350039468 , 0.0252801324472093 , -2.6994456461677e
-17 , -0.0426821670692607 , 0.0958889551620961 ,
-0.148016257409132 , 0.186112126299831 , 0.80031766663488 ,
0.186112126299831 , -0.148016257409132 ,
0.0958889551620961 , -0.0426821670692607 , -2.6994456461677e
-17 , 0.0252801324472093 , -0.0323979350039468 ,
0.025823635119039 , -0.0127268712987161 , -1.68467892397761e
-17 , 0.00796400731979479 , -0.010040824322041 ,
0.0077148790607059 , -0.0036046519659237 , -6.69912201787522e
-18 , 0.00196357837057156 , -0.00231835925473622 ,
0.00172480322214108 , -0.000843883995071432 , -2.49582062811498e-18
};

5 arm_fir_instance_f32 S_FIR_LP;
6 arm_fir_init_f32(&S_FIR_LP , NUM_TAPS_LP , firCoeffsLP , firStateLP ,
BLOCK_SIZE);
7 arm_fir_f32(&S_FIR_LP , &inputSample , &tempBuffer1 , BLOCK_SIZE);

```

Listagem 4.5: Filtro FIR passa-baixo no STM32

Passa-Alto

- **Frequências de corte:** $f_c = 0.5$ Hz: para eliminar a *baseline drift*.
- **Taxa de amostragem (f_s):** definida conforme o sistema de aquisição de dados, por exemplo, $f_s = 500$ Hz.

O filtro passa-alto é essencial para cancelar a *baseline drift* em sinais de ECG, porque ele atenua eficazmente as frequências baixas, geralmente entre 0.5 Hz e 1 Hz, que são típicas da deriva da linha de base. Esse tipo de filtro é crucial para garantir que o sinal de ECG reflete mais precisamente a atividade elétrica do coração, sem as distorções causadas por movimentos lentos ou outras fontes de ruído de baixa frequência. Na Listagem 4.6 está representado o código usado para a implementação do filtro FIR passa-baixo no MATLAB.

```

1 % Filtro FIR Passa-Alto de 0.5 Hz
2 f_corte = 0.5;           % Frequencia de corte
3 ordem_pa = 100;          % Ordem do filtro, pode ser ajustada
4 f_norm = f_corte / f_nyquist;    % Frequencia normalizada
5 pa_fir = fir1(ordem_pa , f_norm , 'high');    % Design do filtro
6 filteredData = filter(pa_fir , 1 , dataStore);  % Aplicacao do
filter

```

```
7 filteredData_2 = filter(pa_fir, 1, data_notched_2); % Aplicacao
8 do filtro
9 fprintf('Filtro FIR Passa-Alto\nNumero de Taps: %d\nCoeficientes:
10 %s\n\n', length(pa_fir), mat2str(pa_fir));
11 [h_pa, w_pa] = freqz(pa_fir, 1, 1024, fs);
12 figure;
13 subplot(3,1,1); % Adiciona um terceiro grafico
14 plot(dataStore, 'b');
15 hold on;
16 plot(filteredData, 'r');
17 title('Dados Originais vs. FIR PASSA-ALTO');
18 xlabel('Amostras');
19 ylabel('Valor');
20 legend('Dados Originais', 'FIR PASSA-ALTO');
21 subplot(3,1,2);
22 plot(w_pa, 20*log10(abs(h_pa)));
23 title('Resposta em Magnitude - FIR Passa-Alto');
24 xlabel('Frequencia (Hz)');
25 ylabel('Magnitude (dB)');
26 subplot(3,1,3);
27 plot(w_pa, angle(h_pa)*180/pi);
28 title('Resposta de Fase - FIR Passa-Alto');
29 xlabel('Frequencia (Hz)');
30 ylabel('Fase (graus)');
```

Listagem 4.6: Filtro FIR passa-alto

Na Figura 4.5 está representada graficamente a resposta ao filtro FIR passa-alto e a sua resposta em amplitude e fase, respectivamente.

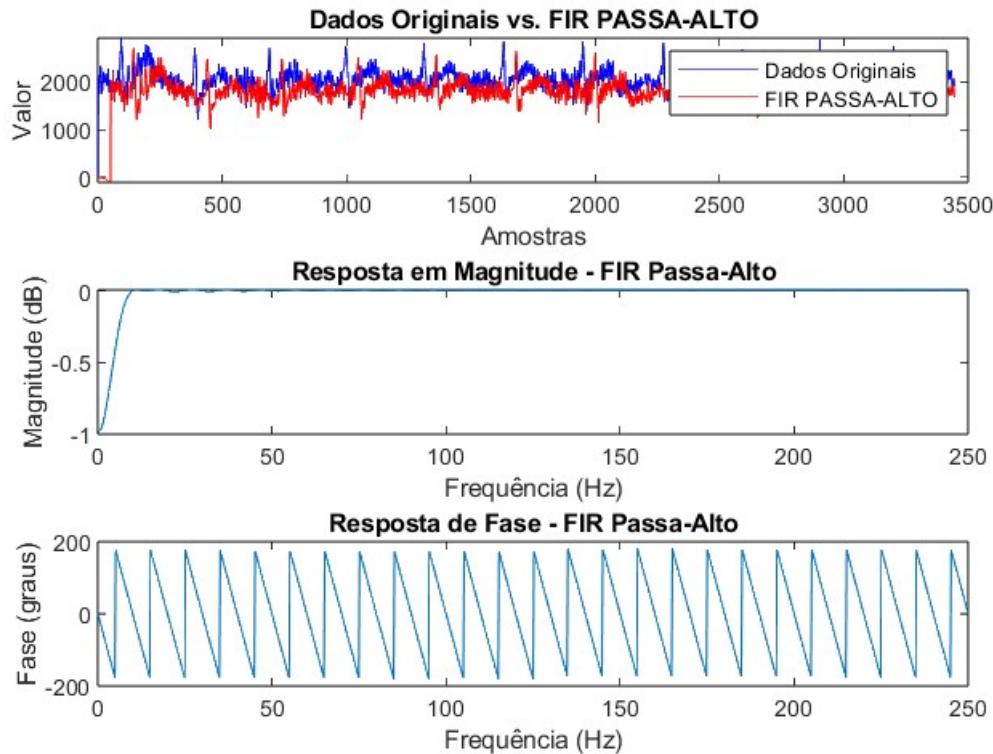


Figura 4.5: Resposta ao filtro FIR passa-alto

De acordo com as informações geradas através do MATLAB, procedeu-se à respetiva implementação do filtro no STM32. Assim, a Listagem 4.7 representa os coeficientes necessários para a aplicação do filtro em STM32.

```

1 #define BLOCK_SIZE           1      // Processa um dado por vez na
2                           // interrupcao
2 #define NUM_TAPS_HP         101    // Numero de taps no FIR Passa-
3                           // Alto
3 float32_t firStateHP[NUM_TAPS_HP + BLOCK_SIZE - 1];
4 arm_fir_instance_f32 S_FIR_HP;
5 arm_fir_init_f32(&S_FIR_HP, NUM_TAPS_HP, firCoeffsHP, firStateHP,
6                   BLOCK_SIZE);
6 arm_fir_f32(&S_FIR_HP, &tempBuffer2, &tempBuffer1, BLOCK_SIZE);
7 float32_t firCoeffsHP[NUM_TAPS_HP] =
8   {-0.000157405827432331, -0.000159296208787861,
9   -0.000164756372485308, -0.000173771824085955,
10  -0.000186313716983139, -0.000202338951392229,
11  -0.000221790331446695, -0.000244596779823655,
12  -0.000270673609085908, -0.000299922848691336,
13  -0.000332233626392146, -0.000367482602519124,
14  -0.00040553445542883, -0.000446242416176931,
15  -0.00048944885027906, -0.000534985884221702,
```

```

15 -0.000582676074202163,-0.000632333114397873,
16 -0.000683762581902517,-0.000736762715311279,
17 -0.000791125223798025,-0.000846636123397069,
18 -0.000903076597091444,-0.000960223875203619,
19 -0.00101785213250446,-0.00107573339837952,
20 -0.00113363847633987,-0.00119133786912079,
21 -0.00124860270558837,-0.0013052056656613,
22 -0.00136092189946731,-0.00141552993696599,
23 -0.00146881258431719,-0.00152055780331749,
24 -0.00157055957030379,-0.00161861871099773,
25 -0.00166454370786934,-0.00170815147670307,
26 -0.0017492681091829,-0.00178772957844312,
27 -0.00182338240469029,-0.00185608427815808,
28 -0.0018857046368413,-0.00191212519662763,
29 -0.00193524043165451,-0.00195495800291107,
30 -0.00197119913332778,-0.00198389892780654,
31 -0.00199300663688009,-0.00199848586291141,
32 0.998157039286965,-0.00199848586291141,
33 -0.00199300663688009,-0.00198389892780654,
34 -0.00197119913332778,-0.00195495800291107,
35 -0.00193524043165451,-0.00191212519662763,
36 -0.0018857046368413,-0.00185608427815808,
37 -0.00182338240469029,-0.00178772957844312,
38 -0.0017492681091829,-0.00170815147670307,
39 -0.00166454370786934,-0.00161861871099773,
40 -0.00157055957030379,-0.00152055780331749,
41 -0.00146881258431719,-0.00141552993696599,
42 -0.00136092189946731,-0.0013052056656613,
43 -0.00124860270558837,-0.00119133786912079,
44 -0.00113363847633987,-0.00107573339837952,
45 -0.00101785213250446,-0.000960223875203619,
46 -0.000903076597091444,-0.000846636123397069,
47 -0.000791125223798025,-0.000736762715311279,
48 -0.000683762581902517,-0.000632333114397873,
49 -0.000582676074202163,-0.000534985884221702,
50 -0.00048944885027906,-0.000446242416176931,
51 -0.00040553445542883,-0.000367482602519124,
52 -0.000332233626392146,-0.000299922848691336,
53 -0.000270673609085908,-0.000244596779823655,
54 -0.000221790331446695,-0.000202338951392229,
55 -0.000186313716983139,-0.000173771824085955,
56 -0.000164756372485308,-0.000159296208787861,
57 -0.000157405827432331 };

```

Listagem 4.7: Filtro FIR passa-baixo no STM32

Média Móvel

O filtro de média móvel, é um tipo de filtro simples, que calcula um novo valor filtrado baseado na média dos estados das amostras anteriores. Tem diversas vantagens como, a simplicidade de implementação, a não necessidade de coeficientes, estabilidade devido a ausência de pólos na função transferência e eficácia contra ruídos de amplitudes aleatórias [24]. Assim permite suavizar a onda e retirar todo o ruído inerente. Este filtro tem o seguinte comportamento relativo a uma onda, representado na Figura 4.6.

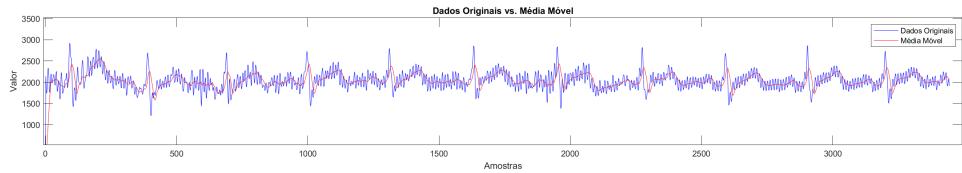


Figura 4.6: Filtro média móvel aplicado

Na Figura 4.7 está representada graficamente a resposta ao filtro de média móvel e a sua resposta em amplitude e fase, respetivamente.

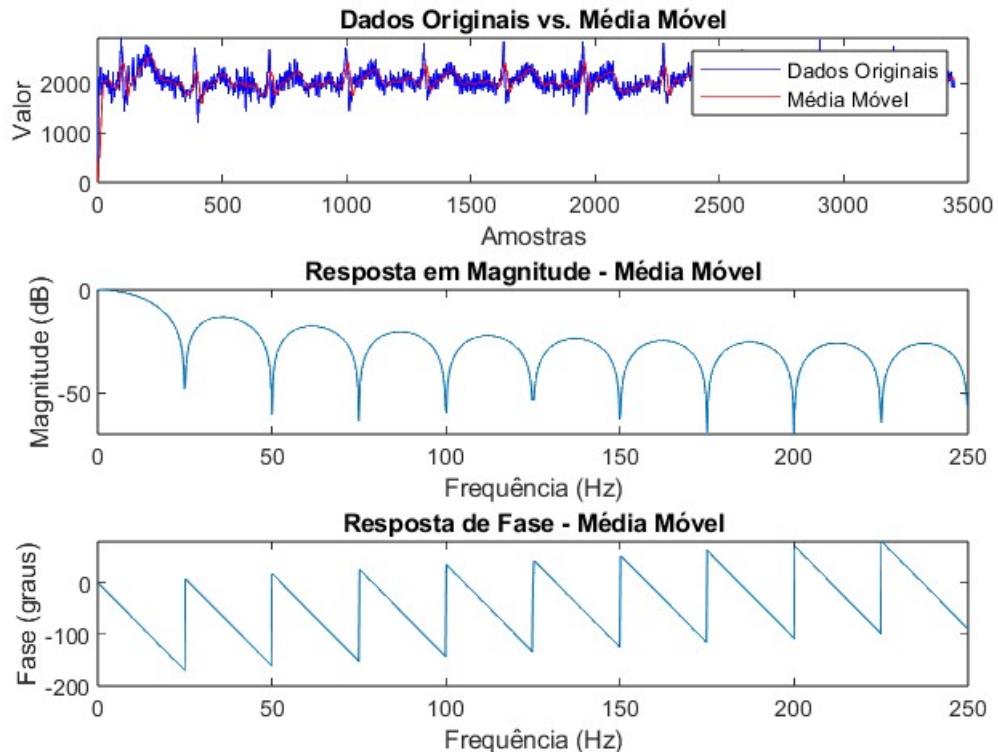


Figura 4.7: Filtro média móvel aplicado

```

1
2 % Filtro de Media Movel
3 windowSize = 20;                                % Define o tamanho da janela
4 para a media movel
5 b_ma = ones(1, windowSize) / windowSize;
6 moving_avg_filtered = filter(b_ma, 1, dataStore);
7 moving_avg_filtered_2 = filter(b_ma, 1, filteredData_2);
8 fprintf('Filtro de Media Movel\nTamanho da Janela: %d\
nCoeficientes: %s\n\n', windowSize, mat2str(b_ma));
9 [h_ma, w_ma] = freqz(b_ma, 1, 1024, fs);
10 figure;
11 subplot(3,1,1); % Adiciona um terceiro grafico
12 plot(dataStore, 'b');
13 hold on;
14 plot(moving_avg_filtered, 'r');
15 title('Dados Originais vs. Media Movel');
16 xlabel('Amostras');
17 ylabel('Valor');
18 legend('Dados Originais', 'Media Movel');
19 subplot(3,1,2);
20 plot(w_ma, 20*log10(abs(h_ma)));
21 title('Resposta em Magnitude - Media Movel');
22 xlabel('Frequencia (Hz)');
23 ylabel('Magnitude (dB)');
24 subplot(3,1,3);
25 plot(w_ma, angle(h_ma)*180/pi);
26 title('Resposta de Fase - Media Movel');
27 xlabel('Frequencia (Hz)');
28 ylabel('Fase (graus)');

```

Listagem 4.8: Filtro Média Móvel

Da mesma forma que foram implementados os filtros anteriores, também o filtro de média móvel implementado no Matlab retorna coeficientes que serão utilizados no STM32 afim de aplicar o mesmo ao sinal obtido em tempo real. O código referente a esta implementação está apresentado na Listagem 4.9.

```

1 #define BLOCK_SIZE           1      // Processa um dado por vez na
2                         interruptao
2 #define NUM_TAPS_MA          20     // Numero de taps no filtro de
3                         Media Movel
3 float32_t firStateMA[NUM_TAPS_MA + BLOCK_SIZE - 1];
4 float32_t firCoeffsMA[NUM_TAPS_MA] = {0.05, 0.05, 0.05, 0.05,
5                         0.05, 0.05, 0.05, 0.05, 0.05, 0.05,
6                         0.05, 0.05, 0.05, 0.05, 0.05, 0.05, 0.05,
7                         0.05, 0.05, 0.05, 0.05, 0.05};
5 arm_fir_instance_f32 S_FIR_MA;

```

```

6 arm_fir_init_f32(&S_FIR_MA, NUM_TAPS_MA, firCoeffsMA, firStateMA,
                  BLOCK_SIZE);
7 arm_fir_f32(&S_FIR_MA, &tempBuffer1, &outputSample, BLOCK_SIZE);

```

Listagem 4.9: Filtro Média Móvel

O coeficiente final da média móvel é determinado pela expressão por $1/N$, em que N representa o número total de valores presentes no *array* de coeficientes.

O número total de coeficientes foi determinado como 20 pois este possui um melhor desempenho na rejeição de ruído, e uma resposta em frequência mais nítida.

4.2.3 Filtro IIR

Os filtros IIR são normalmente utilizados para aplicações onde a fase linear não é tão importante e a memória disponível é limitada. Têm como características um baixo custo de implementação e baixa latência [23]. Este filtro terá o papel fundamental de remover ruídos derivados da corrente elétrica, que como estamos em Portugal é de 50 Hz.

Notch

- **Frequências de corte:** $f_c = 50$ Hz: para remover a interferência da corrente elétrica.
- **Fator de Qualidade:** $Q = 40$

Mais uma vez, na Listagem 4.10 está representado o código usado para a implementação do filtro IIR *notch* no MATLAB.

```

1 % Filtro IIR Notch de 50 Hz
2 f0 = 50;                                % Frequencia central do notch
3                                     % em Hz
4 Q = 40;                                 % Fator de qualidade
5 bw = f0 / Q;                            % Largura de banda
6 [b, a] = iirnotch(f0 / (fs/2), bw / (fs/2)); % Coeficientes do
7                                     % filtro notch com frequencia normalizada
8 data_notched = filter(b, a, dataStore);    % Aplicacao do filtro
9 data_notched_2 = filter(b, a, data_low_passed);
10 fprintf('Filtro IIR Notch\nCoeficientes b: %s\nCoeficientes a: %s\
           \n', mat2str(b), mat2str(a));
11 [h_notch, w_notch] = freqz(b, a, 1024, fs);
12 figure;
13 subplot(3,1,1); % Adiciona um terceiro grafico
14 plot(dataStore, 'b');
15 hold on;
16 plot(data_notched, 'r');

```

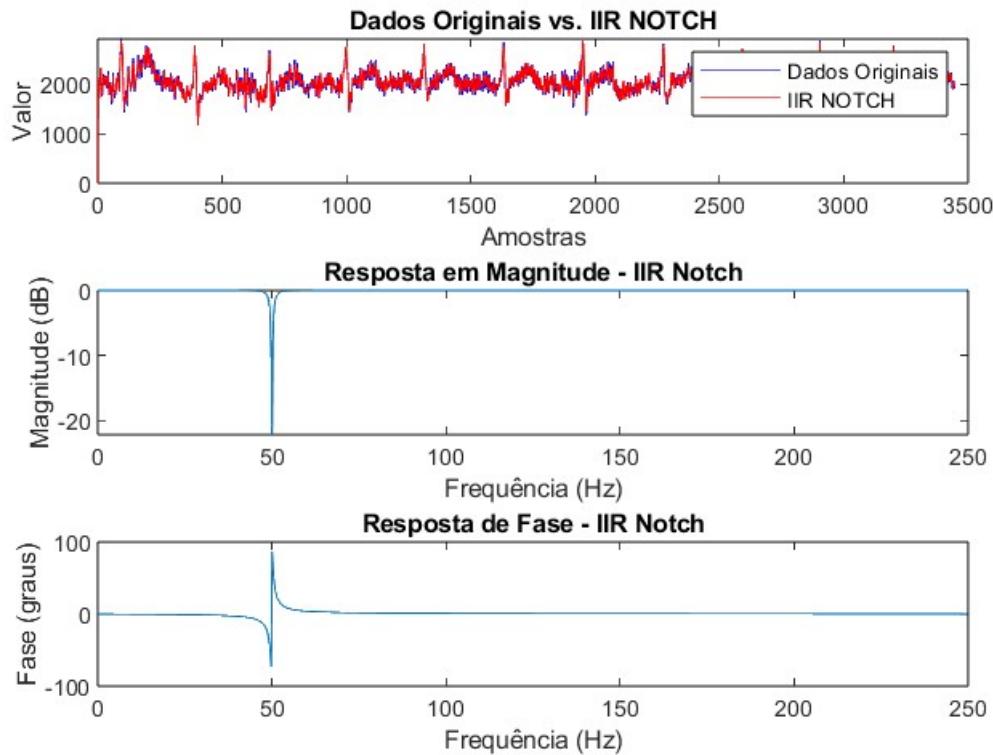
```

15 title('Dados Originais vs. IIR NOTCH');
16 xlabel('Amostras');
17 ylabel('Valor');
18 legend('Dados Originais', 'IIR NOTCH');
19 subplot(3,1,2);
20 plot(w_notch, 20*log10(abs(h_notch)));
21 title('Resposta em Magnitude - IIR Notch');
22 xlabel('Frequencia (Hz)');
23 ylabel('Magnitude (dB)');
24 subplot(3,1,3);
25 plot(w_notch, angle(h_notch)*180/pi);
26 title('Resposta de Fase - IIR Notch');
27 xlabel('Frequencia (Hz)');
28 ylabel('Fase (graus)');

```

Listagem 4.10: Filtro IIR *notch*

Nas Figuras 4.8 estão representadas graficamente a resposta ao filtro IIR *notch* e a sua resposta em amplitude e fase, respectivamente.

Figura 4.8: Resposta ao filtro IIR *notch*

De acordo com as informações geradas através do MATLAB, procedeu-se à respectiva implementação do filtro no STM32. Assim, a Listagem 4.11 representa os

coeficientes necessários para a aplicação do filtro em STM32.

```

1 #define BLOCK_SIZE           1      // Processa um dado por vez na
2                           // interrupcao
3 float32_t iirStateNotch[4]={0};
4 float32_t iirCoeffsNotch[5] = { 0.992207063708048 ,
5   -1.60542475295735, 0.992207063708048, 1.60681611383654 ,
6   -0.986133944044009};
7 arm_biquad_casd_df1_inst_f32 S_IIR_Notch;
8 arm_biquad_cascade_df1_init_f32(&S_IIR_Notch, 1, iirCoeffsNotch,
9 iirStateNotch);
10 arm_biquad_cascade_df1_f32(&S_IIR_Notch, &tempBuffer1, &
11 tempBuffer2, BLOCK_SIZE);

```

Listagem 4.11: Filtro IIR *notch* no STM32

4.2.4 Considerações Finais

Aplicando todos os filtros ao sinal original, na Figura 4.9 está apresentado as implementações sucessivas dos vários filtros ao sinal original, obtida através do Matlab.

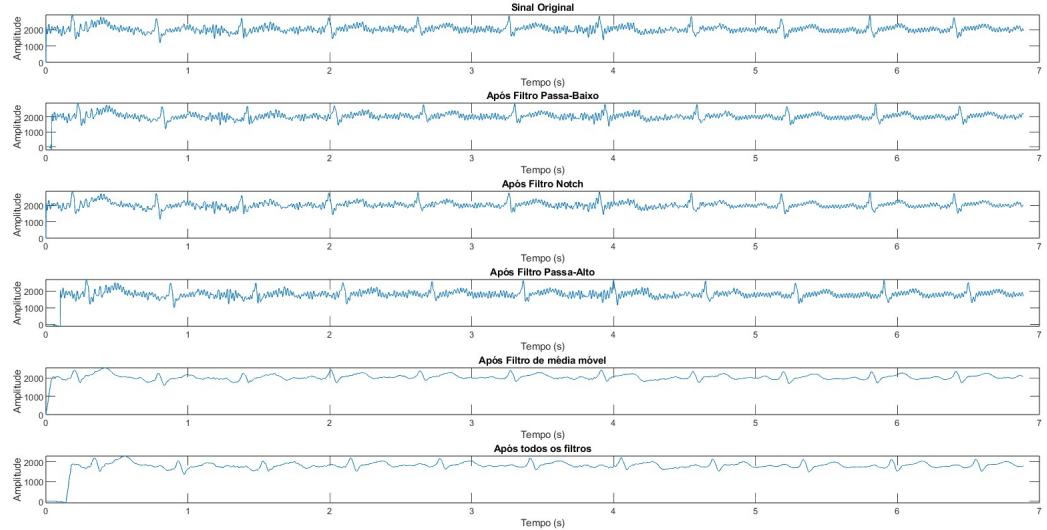


Figura 4.9: Resposta aos vários filtros

A escolha entre filtros IIR e FIR depende de vários fatores, como a resposta de fase, a complexidade computacional e os requisitos específicos da aplicação em ECG. Filtros FIR são preferidos quando a linearidade de fase é crucial, enquanto filtros IIR podem ser mais eficientes em termos de ordem do filtro. Filtros *notch* são eficazes na eliminação de interferências de frequências específicas, como a interferência de 50

Hz da rede elétrica. Por o filtro de média móvel ser linear quanto à sua estrutura, o melhor filtro para o implementar foi o FIR. O conjunto destes filtros foi essencial para remover ruídos no ECG, tornando-o numa linha mais percetível, sem tantas oscilações.

4.3 Inteligência artificial

A inteligência artificial está cada vez mais evoluída, sendo cada vez mais utilizada em diversos processos. Existem duas vertentes da inteligência artificial, o *Machine Learning*, onde os computadores aprendem o método de aprendizagem dos humanos e o *Deep learning*, sendo este um tipo da anterior, destacando-se pelo facto de o modelo aprender a executar tarefas de classificação através de imagens, texto ou som, sendo geralmente implementada usando uma arquitetura de rede neuronal.

Um diagnóstico antecipado previne fins trágicos. Desta forma, a implementação de inteligência artificial para fazer esse diagnóstico é uma ferramenta que poderá vir a tornar-se extremamente importante. Mesmo podendo não ser um resultado totalmente correto, e ainda assim necessitar de acompanhamento médico, um diagnóstico prematuro pode salvar uma vida.

O *dataset* escolhido foi o *MIT-BIH Arrhythmia*. Este conjunto de dados contém 48 excertos de meia hora de registos de ECG de ambulatório, obtidos de 47 indivíduos estudados entre 1975 e 1979 [25].

Desta forma, foi implementado um modelo baseado em inteligência artificial. Para isso foi usado o *Google Colab* [26], sendo este um serviço gratuito da Google que permite escrever e executar código Python diretamente no navegador, sem necessidade de configuração prévia. Isto permite que o código seja executado em servidores da Google, o que significa que é possível utilizar a potência de *hardware*, como Unidade de Processamento Gráfico (GPU) e Unidade de Processamento Tensor (TPU), sem ter nenhum custo adicional. Com isto estaremos a poupar as nossas máquinas, visto que o treino de um modelo desgasta bastante o *hardware*. Também é permitida a integração com o *Google Drive*, o que facilita o armazenamento do *dataset*. De modo a implementar o modelo pretendido, primeiramente foram importadas e instaladas todas as bibliotecas necessárias não existentes no *Google Collab*. Entre as mesmas destacam-se a *wfdb*, servindo esta para ler ficheiros .dat, .hea e etc.; a *numpy*, que inclui operações matemáticas eficientes; *sklearn*, ajudando na divisão dos dados e métricas de avaliação do modelo; *tensorflow* e *keras*, que servem para construir e treinar redes neurais; o *matplotlib*; o *panda* e o *seaborn*, tendo como objetivo visualizar os dados e o *imblearn*, servindo para lidar com os dados não balanceados. Na Listagem 4.12 estão apresentadas todas as bibliotecas importadas.

¹ !pip -q install wfdb imbalanced-learn

```

2
3 import os
4 import wfdb
5 from collections import Counter
6 import numpy as np
7 from sklearn.model_selection import train_test_split
8 import tensorflow as tf
9 from tensorflow.keras.models import Sequential
10 from tensorflow.keras.layers import Conv1D, Flatten, Dense,
11     MaxPooling1D, Dropout, Conv1D, MaxPooling1D, Flatten, Dense,
12     LSTM, Bidirectional, BatchNormalization
13 from tensorflow.keras.optimizers import Adam
14 import matplotlib.pyplot as plt
15 from imblearn.over_sampling import SMOTE
16 from tensorflow.keras.callbacks import EarlyStopping,
17     ReduceLROnPlateau
18 from sklearn.metrics import accuracy_score, recall_score, f1_score
19     , confusion_matrix, precision_score
20 import pandas as pd
21 import seaborn as sns
22 from tensorflow.keras.regularizers import l2
23 from tensorflow.keras.utils import to_categorical

```

Listagem 4.12: Importação de bibliotecas

De seguida, como é possível observar na Listagem 4.13, é carregado o diretório do *Google Drive*, onde se encontra o *dataset*.

```

1 try:
2     from google.colab import drive
3     drive.mount('/content/drive', force_remount=True)
4
5     project_directory = '/content/drive/MyDrive/Colab Notebooks/
6         MIT-BIH/physionet.org/files/mitdb/1.0.0/' # Change this
7         path to match your Google Drive directory
8     os.chdir(project_directory)
9
10 except:
11     pass

```

Listagem 4.13: Diretório do *dataset*

Desta forma, foi criada uma lista vazia para guardar os nomes dos registos, estando estes em números, por exemplo, 100.dat. A biblioteca *wfdb* necessita que seja retirada a extensão para ser possível carregar o ficheiro completo, com o .dat e o .hea, pertencentes ao ECG.

Assim, foi reduzido o tamanho do ECG carregado em três mil amostras. Apesar disto, foi ainda efetuada a seleção da primeira derivação, visto que o nosso sistema só tem uma derivação. É então efetuada uma amostra do sinal através de um gráfico, como apresentado na Listagem 4.14.

```
1 # Inicializar uma lista vazia para guardar os numeros dos registos
2 list_records = []
3
4 # Obter uma lista de todos os ficheiros no diretorio atual
5 file_list = os.listdir('.')
6
7 # Iterar sobre cada nome de ficheiro na lista de ficheiros
8 for file_name in file_list:
9     # Dividir o nome do ficheiro pelo ponto para separar as partes
10    parts = file_name.split(".")
11
12    # Verificar se a primeira parte do nome e um numero
13    if parts[0].isdigit():
14        # Se for um numero, adicionar a lista de registos
15        list_records.append(parts[0])
16
17        # Construir o caminho para o ficheiro de registo
18        record_path = f'./{parts[0]}'
19
20        # Ler o registo ECG do caminho especificado, limitando a
21        # leitura a 3000 amostras
22        record = wfdb.rdrecord(record_path, sampto=3000)
23
24        # Extrair o sinal ECG da primeira derivacao do registo
25        ecg_signal = record.p_signal[:,0]
26
27        # Iniciar uma nova figura para o plot
28        plt.figure()
29
30        # Imprimir o sinal ECG
31        plt.plot(ecg_signal)
32
33        # Definir o titulo do grafico com o numero do registo
34        plt.title(f'Sinal ECG para {parts[0]}')
35
36        # Definir a etiqueta do eixo X
37        plt.xlabel('Pontos de Amostra')
38
39        # Definir a etiqueta do eixo Y
40        plt.ylabel('Amplitude')
41
42        # Mostrar o grafico
43        plt.show()
```

Listagem 4.14: Carregamento do ECG

Foi então inicializada uma nova lista vazia com o intuito de armazenar as anotações dos registos, como apresentado na Listagem 4.15, sendo lido o ficheiro com a extensão .atr, de modo a obter essas anotações.

```

1 # Inicializar uma lista vazia para armazenar as anotacoes dos
2 # registo
3
4 # Iterar sobre cada nome de registo armazenado na lista ,
5 # list_records'
6
7 for record_name in list_records:
8     # Construir o caminho para o ficheiro de registo
9     record_path = f'./{record_name}'
10
11
12     # Ler o sinal ECG do registo especificado, extrair apenas a
13     # primeira derivacao
14     record = wfdb.rdrecord(record_path).p_signal[:,0]
15
16
17     # Tentativa de ler a anotacao associada ao registo
18     try:
19         # Ler o ficheiro de anotacao 'atr' associado ao registo
20         ann = wfdb.rddann(record_path, 'atr')
21
22         # Adicionar o objeto de anotacao a lista de anotacoes
23         annotations.append(ann)
24
25     # Capturar o erro caso o ficheiro de anotacao 'atr' nao seja
26     # encontrado
27 except FileNotFoundError:
28     # Imprimir uma mensagem de erro indicando que nao foi
29     # encontrada anotacao
30     print(f"Nenhuma anotacao 'atr' encontrada para o registo {record_name}")

```

Listagem 4.15: Carregamento das anotações do ECG

Foram inicializadas mais duas listas, uma para guardar o tempo entre os intervalos r-r e outra para guardar a anotação. É criado um dicionário com as anotações e as classes definidas. Neste caso, foi definido por números de 0 a 7, ou seja, 8 classes. Para obter o tempo entre os intervalos r-r, foi calculada a diferença entre os picos, onde esses intervalos foram normalizados dividindo os intervalos pelo intervalo máximo sendo assim possível obter todos os intervalos r-r.

Foi definida uma janela de dez segmentos de anotações médicas. Posteriormente, é realizada a verificação da anotação frequente e registada a mesma, ou então fica como normal se não for encontrada. De seguida, estas informações são adicionadas às listas X e Y. A Listagem 4.16 enumera os passos descritos.

```

1 # Inicializacao de listas para armazenar intervalos R-R e as
   respetivas etiquetas de classificacao
2 X = [] # Intervalos R-R
3 y = [] # Etiquetas (0: normal, 1: V, 2: A, etc.)
4
5 # Definicao do tamanho da janela de analise
6 window_size = 10
7
8 # Definir anotacoes de doenças e sua codificacao numerica
9 disease_annotations =
10    'N': 0, # Normal
11    'V': 1, # Extrassistoles Ventriculares
12    'A': 2, 'a': 2, # Extrassistoles Auriculares
13    'L': 3, 'R': 3, # Bloqueios AV
14    'j': 4, 'e': 4, 'E': 4, # Ritmos de Escape Juncionais e
      Ventriculares
15    'F': 5, # Taquicardia Ventricular
16    'f': 6, '[': 6, ']': 6, # Flutter Auricular e Fibrilacao
17    'x': 7 # Bloqueios AV de Segundo/Terceiro Grau
18 }
19
20 # Iterar sobre cada registo na lista de registos
21 for record_name in list_records:
22     # Carregar o sinal e as anotacoes correspondentes
23     annotation = wfdb.rdann(f'./{record_name}', 'atr')
24
25     # Detectar picos no sinal
26     peaks = annotation.sample
27     #Output
28     #[ 18 77 370 ... 649484 649734 649991]
29
30     # Obter os intervalos R-R a partir dos picos detetados
31     intervals = np.diff(peaks)
32     #output
33     #[ 59 293 292 ... 252 250 257]
34
35     # Normalizar os intervalos R-R pelo valor maximo
36     normalized_intervals = intervals / intervals.max()
37     #output
38     #[0.14496314 0.71990172 0.71744472 ... 0.61916462 0.61425061
      0.63144963]
39

```

```

40     # Dividir os intervalos normalizados em segmentos de tamanho
41     # definido pela 'window_size'
42     for i in range(len(intervals) - window_size + 1):
43         # Obter as anotações e intervalos para a janela atual
44         window_annotations = annotation.symbol[i:i + window_size]
45         #output
46         # ['N', 'N', 'N', 'N', 'R', 'R', 'R', 'R', 'R', 'R']
47         window_intervals = normalized_intervals[i:i + window_size]
48         #output
49         #[0.82570806 0.81263617 0.76906318 0.74291939 0.76688453
50             0.77559913 0.78867102 0.83224401 0.77559913 0.7167756 ]
51
52
53         # Determinar a anotação mais frequente dentro da janela
54         most_common_annotation = max(set(window_annotations), key=
55             window_annotations.count)
56
57         # Atribuir uma etiqueta com base na anotação mais comum
58         label = disease_annotations.get(most_common_annotation, 0)
59             # Por defeito é 0 (normal) se não encontrada
60
61         # Adicionar os dados às listas X e y
62         X.append(window_intervals)
63         y.append(label)

```

Listagem 4.16: Definir os tempos entre cada onda r-r e associar a um problema

Assim, é impresso um gráfico de dispersão onde o eixo X representa o desvio padrão, para indicar a quantidade de variação do conjunto. O eixo Y representa as classes das doenças. Este processo está descrito na Listagem 4.17.

```

1 # Converter as listas X e y para arrays numpy para facilitar
2     manipulações matemáticas e plotagem
3 X = np.array(X)
4 y = np.array(y)
5
6 # Criar um gráfico de dispersão onde o eixo X representa o desvio
7     padrão dos intervalos R-R em cada segmento
8 # e o eixo Y representa as etiquetas de doenças codificadas
9     numericamente
10 plt.scatter(np.std(X, axis=1), y, alpha=0.3)
11
12 # Definir a etiqueta do eixo X como 'std', que indica o desvio
13     padrão dos intervalos R-R
14 plt.xlabel("std")
15
16 # Definir a etiqueta do eixo Y como 'disease score', que
17     corresponde às categorias de doenças

```

```
13 plt.ylabel("disease score")
```

Listagem 4.17: Imprimir o gráfico de dispersão

De seguida, é feita uma contagem dos problemas identificados por classe, sendo impresso o gráfico de barras com o número de problemas que o ECG têm em cada classe, como é demonstrado na Listagem 4.18.

```
1
2 # Inicializar um contador para contabilizar as frequencias das
   anotacoes de doenças
3 annotation_counter = Counter()
4
5 # Atualizar o contador com as etiquetas de doença presentes no
   array y
6 annotation_counter.update(y)
7
8 # Criar um grafico de barras para visualizar a distribuicao das
   anotacoes de doença
9 plt.bar(annotation_counter.keys(), annotation_counter.values())
10
11 # Definir o titulo do grafico, enfatizando que ele mostra a
    distribuicao das anotacoes dos batimentos cardiacos
12 plt.title('Distribui o das Anotacoes dos Batimentos Cardiacos')
13
14 # Definir a etiqueta do eixo X como 'Annotation', indicando os
    diferentes tipos de anotacoes
15 plt.xlabel('Anotacao')
16
17 # Definir a etiqueta do eixo Y como 'Count', indicando a contagem
    de cada anotacao
18 plt.ylabel('Contagem')
```

Listagem 4.18: Contagem dos ECG

É também efetuada uma divisão entre 70% dos dados para treino, 15% para teste e 15% para validação. Esta divisão é efetuada de acordo com a Listagem 4.19.

```
1
2 # Primeira divisao: Dividindo em 70% para treino, 30% para teste +
   validacao
3 X_train, X_temp, y_train, y_temp = train_test_split(X, y,
   test_size=0.3, random_state=42)
4
5 # Segunda divisao: Dividindo os 30% em 15% teste e 15% validacao
6 X_test, X_validation, y_test, y_validation = train_test_split(
   X_temp, y_temp, test_size=0.5, random_state=42)
```

Listagem 4.19: Divisão dos dados

De modo a haver o balanceamento das classes é usado o *Synthetic Minority Over-sampling Technique* (SMOTE) que é uma técnica de sobreamostragem que cria exemplos sintéticos da classe minoritária para corrigir o desequilíbrio entre classes num conjunto de dados de treino.

Aqui, o SMOTE é aplicado aos dados de treino (X_{train} , Y_{train}). O método `fit_resample()` ajusta o SMOTE aos dados existentes e cria novos exemplos sintéticos, resultando em X_{train_smote} e Y_{train_smote} , que têm um equilíbrio de classes melhorado.

`to_categorical` é uma função utilizada para converter vetores de classes (que são inteiros) numa matriz de classe binária, conhecida como codificação "*one-hot*". Isto é comum em modelos de classificação onde cada coluna da matriz resultante corresponde a uma classe. O número de classes (`num_classes`) é definido como 8, indicando que há 8 classes distintas.

Após isto, é feita novamente a contagem e impresso um novo gráfico de barras, neste momento já balanceado. Este processo todo é demonstrado na Listagem 4.20.

```

1
2 # Corrigir o desequilibrio de classes utilizando a tecnica SMOTE
3 smote = SMOTE()
4
5 # Ajustar o SMOTE aos dados de treino para criar exemplos
6 # sinteticos e equilibrar as classes
7 X_train_smote, y_train_smote = smote.fit_resample(X_train, y_train)
8
9 # Inicializar um contador para contabilizar as frequencias das
10 # anotacoes apos o reamostragem com \ac{smote}
11 annotation_smote_counter = Counter()
12 annotation_smote_counter.update(y_train_smote)
13
14 # Definir o numero de classes existentes nos dados
15 num_classes = 8
16
17 # Converter as etiquetas y para o formato categorico, necessario
18 # para modelos de classifica o multiclasse
19 y_train_smote = to_categorical(y_train_smote, num_classes=
20     num_classes)
21 y_validation = to_categorical(y_validation, num_classes=
22     num_classes)
23 y_test = to_categorical(y_test, num_classes=num_classes)
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
579
580
581
582
583
584
585
586
587
588
589
589
590
591
592
593
594
595
596
597
598
599
599
600
601
602
603
604
605
606
607
608
609
609
610
611
612
613
614
615
616
617
618
619
619
620
621
622
623
624
625
626
627
628
629
629
630
631
632
633
634
635
636
637
638
639
639
640
641
642
643
644
645
646
647
648
649
649
650
651
652
653
654
655
656
657
658
659
659
660
661
662
663
664
665
666
667
668
669
669
670
671
672
673
674
675
676
677
678
679
679
680
681
682
683
684
685
686
687
687
688
689
689
690
691
692
693
694
695
696
697
697
698
699
699
700
701
702
703
704
705
706
707
708
709
709
710
711
712
713
714
715
716
717
717
718
719
719
720
721
722
723
724
725
726
727
727
728
729
729
730
731
732
733
734
735
736
737
737
738
739
739
740
741
742
743
744
745
745
746
747
747
748
749
749
750
751
752
753
754
755
755
756
757
757
758
759
759
760
761
762
763
764
765
765
766
767
767
768
769
769
770
771
772
773
774
775
775
776
777
777
778
779
779
780
781
782
783
784
784
785
786
786
787
788
788
789
789
790
791
792
793
794
794
795
796
796
797
798
798
799
799
800
801
802
803
803
804
805
805
806
807
807
808
809
809
810
811
811
812
813
813
814
815
815
816
817
817
818
819
819
820
821
821
822
823
823
824
825
825
826
827
827
828
829
829
830
831
831
832
833
833
834
835
835
836
837
837
838
839
839
840
841
841
842
843
843
844
845
845
846
847
847
848
849
849
850
851
851
852
853
853
854
855
855
856
857
857
858
859
859
860
861
861
862
863
863
864
865
865
866
867
867
868
869
869
870
871
871
872
873
873
874
875
875
876
877
877
878
879
879
880
881
881
882
883
883
884
885
885
886
887
887
888
889
889
890
891
891
892
893
893
894
895
895
896
897
897
898
899
899
900
901
901
902
903
903
904
905
905
906
907
907
908
909
909
910
911
911
912
913
913
914
915
915
916
917
917
918
919
919
920
921
921
922
923
923
924
925
925
926
927
927
928
929
929
930
931
931
932
933
933
934
935
935
936
937
937
938
939
939
940
941
941
942
943
943
944
945
945
946
947
947
948
949
949
950
951
951
952
953
953
954
955
955
956
957
957
958
959
959
960
961
961
962
963
963
964
965
965
966
967
967
968
969
969
970
971
971
972
973
973
974
975
975
976
977
977
978
979
979
980
981
981
982
983
983
984
985
985
986
987
987
988
989
989
990
991
991
992
993
993
994
995
995
996
997
997
998
999
999
1000
1000
1001
1001
1002
1002
1003
1003
1004
1004
1005
1005
1006
1006
1007
1007
1008
1008
1009
1009
1010
1010
1011
1011
1012
1012
1013
1013
1014
1014
1015
1015
1016
1016
1017
1017
1018
1018
1019
1019
1020
1020
1021
1021
1022
1022
1023
1023
1024
1024
1025
1025
1026
1026
1027
1027
1028
1028
1029
1029
1030
1030
1031
1031
1032
1032
1033
1033
1034
1034
1035
1035
1036
1036
1037
1037
1038
1038
1039
1039
1040
1040
1041
1041
1042
1042
1043
1043
1044
1044
1045
1045
1046
1046
1047
1047
1048
1048
1049
1049
1050
1050
1051
1051
1052
1052
1053
1053
1054
1054
1055
1055
1056
1056
1057
1057
1058
1058
1059
1059
1060
1060
1061
1061
1062
1062
1063
1063
1064
1064
1065
1065
1066
1066
1067
1067
1068
1068
1069
1069
1070
1070
1071
1071
1072
1072
1073
1073
1074
1074
1075
1075
1076
1076
1077
1077
1078
1078
1079
1079
1080
1080
1081
1081
1082
1082
1083
1083
1084
1084
1085
1085
1086
1086
1087
1087
1088
1088
1089
1089
1090
1090
1091
1091
1092
1092
1093
1093
1094
1094
1095
1095
1096
1096
1097
1097
1098
1098
1099
1099
1100
1100
1101
1101
1102
1102
1103
1103
1104
1104
1105
1105
1106
1106
1107
1107
1108
1108
1109
1109
1110
1110
1111
1111
1112
1112
1113
1113
1114
1114
1115
1115
1116
1116
1117
1117
1118
1118
1119
1119
1120
1120
1121
1121
1122
1122
1123
1123
1124
1124
1125
1125
1126
1126
1127
1127
1128
1128
1129
1129
1130
1130
1131
1131
1132
1132
1133
1133
1134
1134
1135
1135
1136
1136
1137
1137
1138
1138
1139
1139
1140
1140
1141
1141
1142
1142
1143
1143
1144
1144
1145
1145
1146
1146
1147
1147
1148
1148
1149
1149
1150
1150
1151
1151
1152
1152
1153
1153
1154
1154
1155
1155
1156
1156
1157
1157
1158
1158
1159
1159
1160
1160
1161
1161
1162
1162
1163
1163
1164
1164
1165
1165
1166
1166
1167
1167
1168
1168
1169
1169
1170
1170
1171
1171
1172
1172
1173
1173
1174
1174
1175
1175
1176
1176
1177
1177
1178
1178
1179
1179
1180
1180
1181
1181
1182
1182
1183
1183
1184
1184
1185
1185
1186
1186
1187
1187
1188
1188
1189
1189
1190
1190
1191
1191
1192
1192
1193
1193
1194
1194
1195
1195
1196
1196
1197
1197
1198
1198
1199
1199
1200
1200
1201
1201
1202
1202
1203
1203
1204
1204
1205
1205
1206
1206
1207
1207
1208
1208
1209
1209
1210
1210
1211
1211
1212
1212
1213
1213
1214
1214
1215
1215
1216
1216
1217
1217
1218
1218
1219
1219
1220
1220
1221
1221
1222
1222
1223
1223
1224
1224
1225
1225
1226
1226
1227
1227
1228
1228
1229
1229
1230
1230
1231
1231
1232
1232
1233
1233
1234
1234
1235
1235
1236
1236
1237
1237
1238
1238
1239
1239
1240
1240
1241
1241
1242
1242
1243
1243
1244
1244
1245
1245
1246
1246
1247
1247
1248
1248
1249
1249
1250
1250
1251
1251
1252
1252
1253
1253
1254
1254
1255
1255
1256
1256
1257
1257
1258
1258
1259
1259
1260
1260
1261
1261
1262
1262
1263
1263
1264
1264
1265
1265
1266
1266
1267
1267
1268
1268
1269
1269
1270
1270
1271
1271
1272
1272
1273
1273
1274
1274
1275
1275
1276
1276
1277
1277
1278
1278
1279
1279
1280
1280
1281
1281
1282
1282
1283
1283
1284
1284
1285
1285
1286
1286
1287
1287
1288
1288
1289
1289
1290
1290
1291
1291
1292
1292
1293
1293
1294
1294
1295
1295
1296
1296
1297
1297
1298
1298
1299
1299
1300
1300
1301
1301
1302
1302
1303
1303
1304
1304
1305
1305
1306
1306
1307
1307
1308
1308
1309
1309
1310
1310
1311
1311
1312
1312
1313
1313
1314
1314
1315
1315
1316
1316
1317
1317
1318
1318
1319
1319
1320
1320
1321
1321
1322
1322
1323
1323
1324
1324
1325
1325
1326
1326
1327
1327
1328
1328
1329
1329
1330
1330
1331
1331
1332
1332
1333
1333
1334
1334
1335
1335
1336
1336
1337
1337
1338
1338
1339
1339
1340
1340
1341
1341
1342
1342
1343
1343
1344
1344
1345
1345
1346
1346
1347
1347
1348
1348
1349
1349
1350
1350
1351
1351
1352
1352
1353
1353
1354
1354
1355
1355
1356
1356
1357
1357
1358
1358
1359
1359
1360
1360
1361
1361
1362
1362
1363
1363
1364
1364
1365
1365
1366
1366
1367
1367
1368
1368
1369
1369
1370
1370
1371
1371
1372
1372
1373
1373
1374
1374
1375
1375
1376
1376
1377
1377
1378
1378
1379
1379
1380
1380
1381
1381
1382
1382
1383
1383
1384
1384
1385
1385
1386
1386
1387
1387
1388
1388
1389
1389
1390
1390
1391
1391
1392
1392
1393
1393
1394
1394
1395
1395
1396
1396
1397
1397
1398
1398
1399
1399
1400
1400
1401
1401
1402
1402
1403
1403
1404
1404
1405
1405
1406
1406
1407
1407
1408
1408
1409
1409
1410
1410
1411
1411
1412
1412
1413
1413
1414
1414
1415
1415
1416
1416
1417
1417
1418
1418
1419
1419
1420
1420
1421
1421
1422
1422
1423
1423
1424
1424
1425
1425
1426
1426
1427
1427
1428
1428
1429
1429
1430
1430
1431
1431
1432
1432
1433
1433
1434
1434
1435
1435
1436
1436
1437
1437
1438
1438
1439
1439
1440
1440
1441
1441
1442
1442
1443
1443
1444
1444
1445
1445
1446
1446
1447
1447
1448
1448
1449
1449
1450
1450
1451
1451
1452
1452
1453
1453
1454
1454
1455
1455
1456
1456
1457
1457
1458
1458
1459
1459
1460
1460
1461
1461
1462
1462
1463
1463
1464
1464
1465
1465
1466
1466
1467
1467
1468
1468
1469
1469
1470
1470
1471
1471
1472
1472
1473
1473
1474
1474
1475
1475
1476
1476
1477
1477
1478
1478
1479
1479
1480
1480
1481
1481
1482
1482
1483
1483
1484
1484
1485
1485
1486
1486
1487
1487
1488
1488
1489
1489
1490
1490
1491
1491
1492
1492
1493
1493
1494
1494
1495
1495
1496
1496
1497
1497
1498
1498
1499
1499
1500
1500
1501
1501
1502
1502
1503
1503
1504
1504
1505
1505
1506
1506
1507
1507
1508
1508
1509
1509
1510
1510
1511
1511
1512
1512
1513
1513
1514
1514
1515
1515
1516
1516
1517
1517
1518
1518
1519
1519
1520
1520
1521
1521
1522
1522
1523
1523
1524
1524
1525
1525
1526
1526
1527
1527
1528
1528
1529
1529
1530
1530
1531
1531
1532
1532
1533
1533
1534
1534
1535
1535
1536
1536
1537
1537
1538
1538
1539
1539
1540
1540
1541
1541
1542
1542
1543
1543
1544
1544
1545
1545
1546
1546
1547
1547
1548
1548
1549
1549
1550
1550
1551
1551
1552
1552
1553
1553
1554
1554
1555
1555
1556
1556
1557
1557
1558
1558
1559
1559
1560
1560
1561
1561
1562
1562
1563
1563
1564
1564
1565
1565
1566
1566
1567
1567
1568
1568
1569
1569
1570
1570
1571
1571
1572
1572
1573
1573
1574
1574
1575
1575
1576
1576
1577
1577
1578
1578
1579
1579
1580
1580
1581
1581
1582
1582
1583
1583
1584
1584
1585
1585
1586
1586
1587
1587
1588
1588
1589
1589
1590
1590
1591
1591
1592
1592
1593
1593
```

```

20 # Criar um grafico de barras para visualizar a distribuicao das
21 # anotacoes apos a reamostragem
22 plt.bar(annotation_smote_counter.keys(), annotation_smote_counter.
23         values())
24 plt.title('Distribuicao das Anotacoes dos Batimentos Cardiacos')
25 plt.xlabel('Anotacao')
26 plt.ylabel('Contagem')
27 plt.show() # Mostrar o grafico
28
29 # Definir o caminho para salvar os modelos treinados dentro do
30 # diretorio do projeto
31 output_path = os.path.join(project_directory, 'models')

```

Listagem 4.20: Balanceamento dos dados

Neste momento, o modelo LSTM é criado. Como já referido anteriormente, é destinado a dados temporais, a principal razão para a sua escolha. O número de camadas, de épocas e neurónios são determinados através da experimentação e validação cruzada, que será explicado mais à frente. A ideia é encontrar um equilíbrio entre capacidade de aprendizagem e risco de *overfitting*.

Foi implementado um *callback EarlyStopping*, que ajuda a parar o treino quando uma métrica acompanhada para de melhorar, evitando assim o *overfitting* e reduzindo o tempo de treino, melhorando o sistema para não aprender ruído.

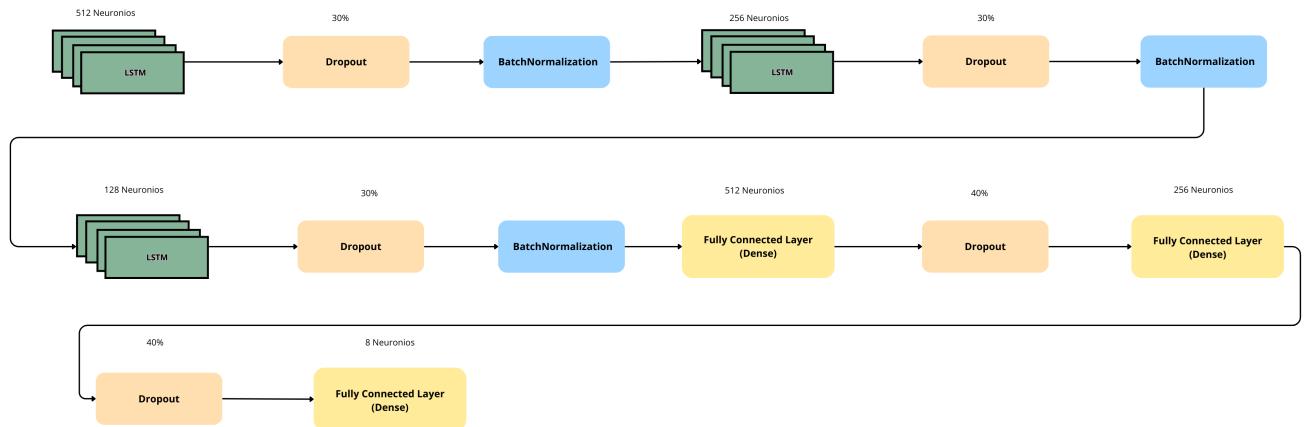


Figura 4.10: Arquitetura da AI

Foi criado um modelo LSTM, como é possível observar na figura 4.10, com três camadas onde, a primeira contém 512 neurónios e retorna sequências, ou seja, a respetiva saída é enviada à próxima camada. Relativamente à segunda camada, tem um comportamento idêntico à primeira camada contendo apenas 256 neurónios. Por fim, a terceira camada contém 128 neurónios, não retorna sequências, retornando apenas o último estado da saída, que será o valor passado à próxima camada.

O modelo apresenta vários parâmetros, como o número de unidades na camada (neurónios), que determina a dimensão no espaço de saída; o `return_sequences=True` que faz com que cada instante de tempo na sequência de entrada produza uma saída correspondente para a sequência, necessário quando se empilha camadas; o `input_shape=(window_size,1)` define a forma da entrada esperada pelo modelo, onde `window_size` é o tamanho da janela de tempo considerada e 1 é o número de características por instante de tempo (por exemplo, um único canal de ECG) e o `kernel_regularizer=l2(0.01)` que aplica a regularização L2 às matrizes dos pesos da camada, penalizando pesos grandes com um fator de 0.01 para evitar *overfitting*.

Entre estas camadas existem *dropout layers* de modo a prevenirem o *overfitting* ao aleatoriamente ignorarem uma fração de *outputs* das camadas durante o treino. Após esta ação, existem *Batch Normalization Layers* que normaliza as ativações da camada anterior, reescalando e centrando-as em torno de zero e reduzindo a *covariate shift* entre as camadas do modelo. Isto geralmente resulta num treino mais rápido e estável.

No final destas camadas existem outras 3 camadas de *Dense (Fully Connected) Layers*, onde a primeira contém 512 neurónios e a sua ativação *Rectified Linear Unit* (ReLU); a segunda possui um comportamento semelhante à primeira, executando a mesma ativação e por fim, a terceira camada é a camada de saída, onde existem oito neurónios correspondentes às oito classes existentes.

Estas camadas possuem parâmetros como o número de unidades (neurónios) e a ativação `activation='relu'` que introduz não-linearidades no modelo. A função de ativação *sigmoid* é usada para que a rede possa produzir um valor entre 0 e 1 para cada classe, interpretável como a probabilidade da classe correspondente.

A função de ativação numa rede neural tem um papel fundamental na modelação das relações complexas nos dados.

Introdução de Não-linearidade

- **Linearidade Limitada:** Sem uma função de ativação não-linear (ou seja, usando apenas transformações lineares), uma rede neuronal, independentemente do número de camadas (profundidade) ou da complexidade, equivaleria a uma função linear simples. Isso limitaria severamente a capacidade do modelo de capturar padrões complexos nos dados, especialmente útil em problemas como classificação, onde as decisões de separação entre as classes geralmente não são lineares.
- **Não-linearidade:** As funções de ativação não-lineares permitem que as redes neuronais criem fronteiras de decisão complexas e aprendam a modelagem variada e intrincada de padrões nos dados.

Transformação dos Dados de Entrada

- Cada função de ativação transforma os dados de entrada de uma maneira que pode ajudar o neurónio a decidir melhor o que deve ser passado adiante. Ao modificar os dados de entrada acumulados (combinação linear dos *inputs* e seus pesos), a função de ativação define o *output* do neurónio, que será então usado como entrada para a próxima camada.

As funções de ativação usadas têm as seguintes características:

1. Sigmoid

- **Forma:**

$$f(x) = \frac{1}{1 + e^{-x}} \quad (4.8)$$

- **Características:** Converte valores de entrada numa escala entre 0 e 1, tornando-a útil para problemas de classificação binária onde precisamos de probabilidades como saída.
- **Uso:** Camadas de saída em classificação binária.

2. ReLU

- **Forma:**

$$f(x) = \max(0, x) \quad (4.9)$$

- **Características:** Permite apenas valores positivos, sendo zero para qualquer valor negativo de entrada. É muito utilizada pela sua eficiência computacional e porque permite uma rápida convergência da rede durante o treino, além de evitar problemas comuns em gradientes em redes profundas.
- **Uso:** Camadas ocultas para promover aprendizado rápido e eficaz.

Na compilação do modelo é utilizado o *Adam* com uma taxa de aprendizagem de 0.001 de modo a otimizar. O *Adam* é eficaz em termos de computação e tem pouca memória exigida, sendo adequado para problemas com grandes quantidades de dados e/ou muitos parâmetros. Existe também o *loss='binary_crossentropy'*: como o modelo termina com uma ativação *sigmoid* e cada saída é tratada como uma classificação binária independente, a função de perda usada é a entropia cruzada binária. O *metrics=['accuracy']*, ou seja, a precisão é usada como métrica para avaliar o desempenho do modelo. Este processo todo é demonstrado na Listagem 4.21.

```
1
2 # Definicao do modelo LSTM para analise de series temporais, como
3 # sinais de ECG
4 model_lstm = Sequential([
5     # Adicionar uma camada LSTM com 512 unidades, retornando
6     # sequencias para a proxima camada LSTM,
7     # com regularizacao L2 para reduzir o overfitting
8     LSTM(512, return_sequences=True, input_shape=(window_size
9         ,1), kernel_regularizer=l2(0.01)),
10
11    # Adicionar dropout para reduzir o overfitting ao
12    # descartar aleatoriamente conexoes durante o treinamento
13    Dropout(0.3),
14
15    # Normalizacao de batch para acelerar o treinamento e
16    # melhorar a performance do modelo
17    BatchNormalization(),
18
19    # Segunda camada LSTM com 256 unidades, tambem com retorno
20    # de sequencias
21    LSTM(256, return_sequences=True, kernel_regularizer=l2
22        (0.01)),
23    Dropout(0.3),
24    BatchNormalization(),
25
26    # Terceira camada LSTM com 128 unidades, sem retorno de
27    # sequencias
28    LSTM(128, kernel_regularizer=l2(0.01)),
29    Dropout(0.3),
30    BatchNormalization(),
31
32    # Camada densa com 512 unidades e ativacao ReLU
33    Dense(512, activation='relu', kernel_regularizer=l2(0.01))
34        ,
35    Dropout(0.4),
36
37    # Outra camada densa com 256 unidades e ativacao ReLU
38    Dense(256, activation='relu', kernel_regularizer=l2(0.01))
39        ,
40    Dropout(0.4),
41
42    # Camada de saida com 8 unidades, uma para cada classe,
43    # usando a funcao de ativacao sigmoid
44    Dense(8, activation='sigmoid')
45])
46
47 # Compilar o modelo com o otimizador Adam, usando uma taxa de
48 # aprendizagem de 0.001,
```

```

37 # e configurando a função de perda como 'binary_crossentropy' para
    classificação binária multiclasse,
38 # e medindo a precisão do modelo
39 model_lstm.compile(optimizer=Adam(learning_rate=0.001), loss='
    binary_crossentropy', metrics=['accuracy'])

```

Listagem 4.21: Modelo de AI

Para a verificação das métricas de avaliação do modelo, foi criado um dicionário que representa a descrição das classes criadas acima. Cada classe tem uma possível designação descrita na Tabela 4.1 e na Listagem 4.22.

Tabela 4.1: Correspondência entre classes numéricas e anotações de doenças cardíacas

Classe	Anotação	Descrição
0	N	Normal
1	V	Extrassístoles Ventriculares
2	A, a	Extrassístoles Auriculares
3	L, R	Bloqueios AV
4	j, e, E	Ritmos de Escape Juncionais e Ventriculares
5	F	Taquicardia Ventricular
6	f, [,]	Flutter Auricular e Fibrilação
7	x	Bloqueios AV de Segundo/Terceiro Grau

```

1 class_descriptions = {
2     0: 'N',
3     1: 'E',
4     2: 'A',
5     3: 'B',
6     4: 'R',
7     5: 'T',
8     6: 'F',
9     7: 'V'
10 }

```

Listagem 4.22: Dicionário com descrição das classes

Na utilização do *callback EarlyStopping*, foi usado o parâmetro *patience=5*, onde o treino continua por 5 épocas após a deteção de que a melhoria parou. O *restore_best_weights=True* restaura os pesos do modelo ao estado onde a métrica monitorada teve o melhor valor.

É também utilizado outro *callback*, o *ReduceLROnPlateau*, que permite reduzir a taxa de aprendizagem quando a métrica monitorada parou de melhorar, o que pode ajudar a refinar o treino do modelo para alcançar um desempenho melhor, tendo como parâmetros, o *factor=0.1*, que multiplica a taxa de aprendizagem por

0.1 e o *patience*=3, que aguarda 3 épocas sem melhoria antes de reduzir a taxa de aprendizagem.

O método `model.fit` é usado para treinar o modelo:

- **Dados de Entrada:**

- `X_train_smote` e `Y_train_smote`: Dados de treino e etiquetas balanceadas usando SMOTE para lidar com desequilíbrios de classe.

- **Validação:**

- `validation_data=(X_validation, Y_validation)`: Conjunto de dados usados para validar o desempenho do modelo após cada época, não sendo usados para treinar os pesos.

- **Configurações de treino:**

- `epochs=40`: Número total de passagens pelo conjunto de treino.
- `batch_size=32`: Número de amostras por atualização do gradiente.
- `verbose=1`: Mostra a barra de progresso e outras informações durante o treino.
- `callbacks=[callbacks]`: Lista de callbacks definidos anteriormente para controlar o treino.

Avaliação e previsões

- **Avaliação:** `model_lstm.evaluate` calcula a perda e a precisão no conjunto de teste.
- **Previsão:** `model_lstm.predict` gera previsões para o conjunto de teste.
- **Conversão de Previsões:** As previsões são convertidas de formatos probabilísticos para etiquetas de classe.
- **Métricas por Classe:** Calcula precisão, *recall* e *F1-score* para cada classe usando as funções de métricas do *scikit-learn*.

Visualização de Resultados

- **Matriz de Confusão:** Visualiza o desempenho do modelo, mostrando as contagens de verdadeiros positivos, falsos positivos, verdadeiros negativos e falsos negativos para cada classe.
- **Precisão e Perda ao Longo das Épocas:** Mostra gráficos de linha para precisão e perda no treino e na validação, oferecendo uma visão de como o modelo aprendeu ao longo do tempo.

Na Listagem 4.23 são apresentados os passos anteriormente descritos.

```
1 # Treinar o modelo com callbacks para EarlyStopping e
  ReduceLROnPlateau
2 callbacks = [
3     EarlyStopping(patience=5, restore_best_weights=True), # Parar
      cedo se nao houver melhoria
4     ReduceLROnPlateau(factor=0.1, patience=3) # Reduzir a taxa de
      aprendizagem se estagnado
5 ]
6 history = model_lstm.fit(
7     X_train_smote, # Dados de treino balanceados por SMOTE
8     y_train_smote, # Etiquetas de treino balanceadas
9     validation_data=(X_validation, y_validation), # Dados de
      validacao
10    epochs=40, # Numero total de epocas
11    batch_size=32, # Tamanho do lote
12    verbose=1, # Mostrar progresso
13    callbacks=callbacks # Callbacks definidos anteriormente
14 )
15
16 # Avaliar o modelo nos dados de teste e imprimir a precisao e
  perda
17 loss, accuracy = model_lstm.evaluate(X_test, y_test, verbose=0)
18 print(f'Test accuracy: {accuracy}')
19 print(f'Loss: {loss}')
20
21 # Prever no conjunto de teste
22 y_pred = model_lstm.predict(X_test)
23 # Converter as previsoes para etiquetas categoricas
24 y_pred_labels = np.argmax(y_pred, axis=1) if y_pred.shape[1] > 1
  else (y_pred > 0.5).astype(int)
25 # A linha seguinte foi alterada para resolver o erro
26 y_true_labels = np.argmax(y_test, axis=1) # Converter as
  verdadeiras etiquetas para formato categorico
27 # Calcular a precisao geral
28 accuracy = accuracy_score(y_true_labels, y_pred_labels)
29
30 # Calcular metricas por classe
31 precision_per_class = precision_score(y_true_labels, y_pred_labels,
  , average=None) # Precisao por classe
32 recall_per_class = recall_score(y_true_labels, y_pred_labels,
  , average=None) # Recall por classe
33 f1_per_class = f1_score(y_true_labels, y_pred_labels, average=None
  ) # F1-Score por classe
34
35 # Exibir resultados para cada classe
36 print(f"Model: model_lstm")
37 print(f"Overall Accuracy: {accuracy:.2f}")
```

```
38 for idx, (prec, rec, f1) in enumerate(zip(precision_per_class,
39     recall_per_class, f1_per_class)):
40     print(f"Class {class_descriptions[idx]}: Precision: {prec:.2f}
41         }, Recall: {rec:.2f}, F1-Score: {f1:.2f}")
42
43 # Opcionalmente, armazenar resultados de forma estruturada em um
44 # DataFrame
45 results = []
46 for idx, (prec, rec, f1) in enumerate(zip(precision_per_class,
47     recall_per_class, f1_per_class)):
48     results.append({
49         'Model': "model_lstm",
50         'Class': class_descriptions[idx],
51         'Accuracy': accuracy, # Precisao geral, igual para cada
52             classe
53         'Precision': prec,
54         'Recall': rec,
55         'F1-Score': f1
56     })
57 df_results = pd.DataFrame(results)
58
59 # Reformatar DataFrame para o formato desejado: uma linha por
60 # classe, colunas por modelo
61 pivot_table = df_results.pivot_table(index=['Class'], columns=[

62     'Model'], values=['Precision', 'Recall', 'F1-Score', 'Accuracy',
63 ])
64 print(pivot_table)
65
66 # Calcular a matriz de confusao
67 cm = confusion_matrix(y_true_labels, y_pred_labels) # Matriz de
68     confusao entre etiquetas verdadeiras e previstas
69
70 # 'classes_listed' e uma lista com os nomes das classes
71 plt.figure(figsize=(10, 7))
72 sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=
73     class_descriptions[idx], yticklabels=class_descriptions[idx])
74 plt.title('Matriz de Confusao')
75 plt.xlabel('Etiqueta Prevista')
76 plt.ylabel('Etiqueta Verdadeira')
77 plt.show()
78
79 # Imprimir a precisao de treino e validacao ao longo das epocas
80 plt.plot(history.history['accuracy'], label='Precisao de Treino')
81 plt.plot(history.history['val_accuracy'], label='Precisao de
82     Validacao')
83 plt.title('Precisao de Treino e Validacao')
84 plt.xlabel('Epocas')
85 plt.ylabel('Precisao')
86 plt.legend()
```

```

76 plt.show()
77
78 # Imprimir a perda de treino e validacao ao longo das epocas
79 plt.plot(history.history['loss'], label='Perda de Treino')
80 plt.plot(history.history['val_loss'], label='Perda de Validacao')
81 plt.title('Perda de Treino e Validacao')
82 plt.xlabel('Epocas')
83 plt.ylabel('Perda')
84 plt.legend()
85 plt.show()
86
87 # Guardar o modelo treinado no caminho especificado
88 save_path = os.path.join(output_path, 'model_lstm.h5')
89 model_lstm.save(save_path)

```

Listagem 4.23: Treino do modelo

4.4 Interface gráfica

Foi criada uma interface gráfica em *python* de modo a visualizar o eletrocardiograma de uma forma simples. Nesta interface é utilizada uma comunicação através da USART do STM32, recebendo assim os vários valores que compõem o sinal. Desta forma, é possível configurar a porta e o *baud rate*, bem como ativar o uso da inteligência artificial. Está também implementado um algoritmo capaz de calcular o número de batimentos cardíacos por minuto.

Caso esteja ativado o uso de inteligência artificial e a mesma detete alguma anomalia, é enviado um *e-mail* com todas as informações relevantes para o contacto de emergência do paciente.

Diversos métodos foram desenvolvidos pelo que serão enumerados e explicados cada um deles a seguir.

O método representado na Listagem 4.24 serve para verificar se o input do email está preenchido.

```

1
2 def check_email_input(self):
3     """Verifica se o campo de email esta preenchido."""
4     if self.email_input.text().strip():
5         self.submit_button.setEnabled(True)
6     else:
7         self.submit_button.setEnabled(False)

```

Listagem 4.24: Método de verificação do input email

Este método, presente na listagem 4.25, calcula os bpm com base nos intervalos R-R usando uma média. Como há 60.000 milissegundos em um minuto (1.000 milissegundos por segundo e 60 segundos por minuto), é dividido 60.000 pelo número de milissegundos num intervalo RR para descobrir quantos desses intervalos cabem em um minuto.

```

1
2     def calculateBPM(self):
3         """Calcula BPM com base nos intervalos R-R."""
4         if len(self.rr_intervals) > 0:
5             avg_rr_interval = np.mean(self.rr_intervals)
6             self.bpm = 60000 / avg_rr_interval # Converter ms
7                         para BPM
8             self.bpm_label.setText(f"{int(self.bpm)} bpm")

```

Listagem 4.25: M\'etodo para calcular os BPM

Também foi implementado outro método que calcula os batimentos por minuto através de uma média dos últimos batimentos, como está apresentado na Listagem 4.26.

```

1     def calculate_average_bpm(self):
2         """Calcula a media de BPM com base nos ultimos 500
3             batimentos."""
4         beat_new = QDateTime.currentMSecsSinceEpoch() # Obtém o
5                         milissegundo atual
6         diff = beat_new - self.beat_old # Encontra o tempo entre
7                         os dois ultimos batimentos
8         currentBPM = 60000 / diff # Converte para batimentos por
9                         minuto
10        self.beats[self.beatIndex] = currentBPM # Armazena no
11                         array para calcular a media
12        total = np.sum(self.beats) # Soma todos os valores do
13                         array
14        self.bpm = int(total / 500) # Calcula a media
15        self.beat_old = beat_new
16        self.beatIndex = (self.beatIndex + 1) % 500 # Cicla pelo
17                         array em vez de usar uma fila FIFO
18        self.bpm_label.setText(f"{int(self.bpm)} bpm")

```

Listagem 4.26: Método para calcular os BPM por média

Na Listagem 4.27 está representado o método que é utilizado para serem detetadas as portas do computador.

```

2 def update_ports(self):
3     """Atualiza a lista de portas COM disponíveis."""
4     ports = [port.device for port in serial.tools.list_ports.
5               comports()]
6     self.port_combo.clear()
7     self.port_combo.addItems(ports)

```

Listagem 4.27: Método para obter as portas do computador

Através do método descrito na Listagem 4.28 é possível ativar ou desativar a AI.

```

1
2 def toggle_ia(self):
3     """Ativa ou desativa o botão de IA."""
4     self.ia_button.setText("Ativado" if self.ia_button.
5                           isChecked() else "Desativado")

```

Listagem 4.28: Método para ativar a inteligencia artificial e desativar

É também possível ativar a versão *raw* do ECG através do método descrito na Listagem 4.29.

```

1
2 def toggle_second_line(self):
3     """Ativa ou desativa a segunda linha do gráfico."""
4     self.show_second_line = not self.show_second_line

```

Listagem 4.29: Método para ativar a linha do raw

A Listagem 4.30 representa o momento em que é iniciado o ECG.

```

1
2 def start_monitoring(self):
3     """Inicia a monitorização do ECG."""
4     if not self.email_input.text().strip():
5         return
6     if not self.serial or not self.serial.is_open:
7         try:
8             port = self.port_combo.currentText()
9             baud = int(self.baud_input.text())
10            self.serial = serial.Serial(port, baud)
11            self.timer.start(2) # Verificar se com 1 não fica
12                muito rápido
13            self.submit_button.setText("Parar")
14            self.port_combo.setEnabled(False)
15            self.baud_input.setEnabled(False)
16            self.email_input.setEnabled(False)

```

```

16         except Exception as e:
17             print(f"Erro ao conectar: {e}")
18     else:
19         self.stop_monitoring()

```

Listagem 4.30: Método para iniciar o ECG

De forma a interromper o eletrocardiograma é utilizado o método apresentado na Listagem 4.31.

```

1
2     def stop_monitoring(self):
3         """Para a monitoriza o do ECG."""
4         if self.serial and self.serial.is_open:
5             self.timer.stop()
6             self.serial.close()
7             self.serial = None
8             self.submit_button.setText("Continuar")
9             self.port_combo.setEnabled(True)
10            self.baud_input.setEnabled(True)
11            self.email_input.setEnabled(True)

```

Listagem 4.31: Método para parar o ECG

Na Listagem 4.32 é descrito o método utilizado para detetar os picos r, recebendo o valor e o tempo como parâmetros O *thereshold* foi definido acima das menores oscilações, mas abaixo dos picos dos complexos QRS, garantindo que os principais eventos cardíacos sejam detetados sem capturar detalhes irrelevantes. Serve para detetar os complexos QRS e atualizar os batimentos cardíacos.

```

1
2     def detect_r_peak(self, value, timestamp):
3         """Deteta pico R e calcula intervalo R-R."""
4         if value > self.threshold and self.belowThreshold:
5             #self.calculate_average_bpm() # Atualiza BPM com base
6             nos ultimos 500 batimentos
6             self.belowThreshold = False
7             self.r_peaks.append(timestamp)
8             if len(self.r_peaks) > 1:
9                 rr_interval = self.r_peaks[-1] - self.r_peaks[-2]
10                self.rr_intervals.append(rr_interval)
11                if len(self.rr_intervals) >= WINDOW_SIZE:
12                    self.rr_intervals = self.rr_intervals[-
12                      WINDOW_SIZE:]
13                    self.calculateBPM() # Atualiza BPM com base
13                    nos intervalos R-R
14                    #self.calculate_average_bpm()

```

```

15             # Processamento IA
16     if self.ia_button.isChecked():
17         if self.ia_thread is None or not self.
18             ia_thread.is_alive():
19                 self.ia_thread = threading.Thread(
20                     target=self.process_rr_intervals)
21                 self.ia_thread.start()
22
23     elif value < self.threshold:
24         self.belowThreshold = True

```

Listagem 4.32: Método para detetar pico r

De forma a processar o intervalo r-r é utilizado o método da Listagem 4.33, sendo com essa informação possível proceder a um diagnóstico caso a AI esteja ativada.

```

1
2     def process_rr_intervals(self):
3         """Processa intervalos R-R com o modelo."""
4         rr_data = np.array(self.rr_intervals)
5         rr_data = rr_data.reshape(1, len(rr_data), 1)
6         prediction = model.predict(rr_data)
7         predicted_class_index = np.argmax(prediction)
8         self.diagnosis = class_descriptions[predicted_class_index]
9         self.diagnostic_text.setText(self.diagnosis)
10        if self.diagnosis != "Normal":
11            recipient = self.email_input.text()
12            subject = "Alerta de Doença Detectada"
13            body = f"A IA detectou uma poss vel doença no
14                eletrocardiograma. Diagnóstico: {self.diagnosis}"
15            send_email(recipient, subject, body)
16            # Desativar IA apos enviar email
17            self.ia_button.setChecked(False)
18            self.ia_button.setText("Desativado")
19            self.ia_thread = None # Desassociar a thread de IA
20            apos enviar o email

```

Listagem 4.33: Método para processar o intervalo r-r

Usado para estar sempre a atualizar o gráfico e alguns valores, o método presente na Listagem 4.34 foi criado para proceder à verificação se o valor passado por USART é válido. Caso não seja, é pedido logo outro valor no mesmo instante para não haja perda de dados. Existem também verificações para os valores estarem entre 0 e 4095 por causa do ADC do STM32 ($2^{12} = 4096$, ou seja, 4096 valores, intervalo do 0 ao 4095).

```

1
2     def update_plot(self):

```

```

44             self.second_curve.setData(self.
45                     second_line_data)
46         else:
47             self.second_curve.clear()
48
49             timestamp = QDateTime.currentMsecsSinceEpoch()
50             self.detect_r_peak(value1, timestamp)
51
52
53
54
55         except Exception as e:
56             print(f"Erro na atualizacao do grafico: {e}")

```

Listagem 4.34: Método atualizar o gráfico e a interface global

Foi também implementado um método que envia um email com as informações obtidas, sendo necessário uma configuração do mesmo, com a configuração do servidor do email utilizado, password e o próprio email, como demonstrado na Listagem 4.35.

```

1
2     def send_mail(recipient, subject, body):
3         sender_email = "geral@universeextrememind.com"
4         sender_password = ""
5
6         msg = MIME_Multipart()
7         msg['From'] = sender_email
8         msg['To'] = recipient
9         msg['Subject'] = subject
10
11        msg.attach(MIMEText(body, 'plain'))
12
13    try:
14        server = smtplib.SMTP_SSL('smtp.titan.email', 465) # Use
15                    SMTP_SSL for port 465
16        server.login(sender_email, sender_password)
17        text = msg.as_string()
18        server.sendmail(sender_email, recipient, text)
19        server.quit()
20        print("Email sent successfully")
21    except smtplib.SMTPAuthenticationError as e:
22        print(f"Authentication error: {e.smtp_code} - {e.
23                     smtp_error.decode()}")
24    except Exception as e:
25        print(f"Error occurred while sending email: {e}")

```

Listagem 4.35: Método para enviar email

Na Figura 4.11 está representada a interface gráfica desenvolvida para este efeito.

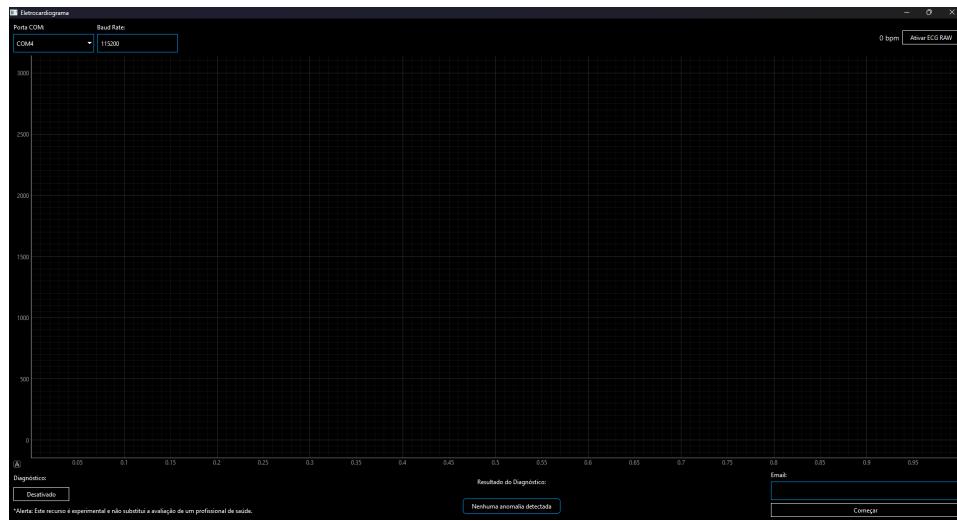


Figura 4.11: Interface gráfica

4.5 Testes e resultados

Depois dos componentes interligados, o aspetto da montagem final do circuito está apresentado na Figura 4.12.

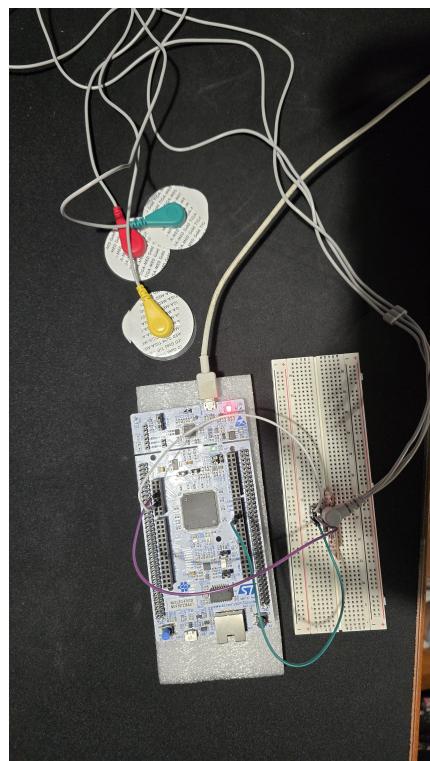


Figura 4.12: Montagem do Protótipo

Após as montagens, e de forma a perceber se o sinal que estava a ser transmitido na interface gráfica era o adequado, foi utilizado um osciloscópio de forma a visualizar o sinal de saída do Sen-12650, obtendo-se o sinal apresentado na Figura 4.13.

É possível constatar a considerável existência de ruído, mas é perceptível as características de uma onda de ECG.

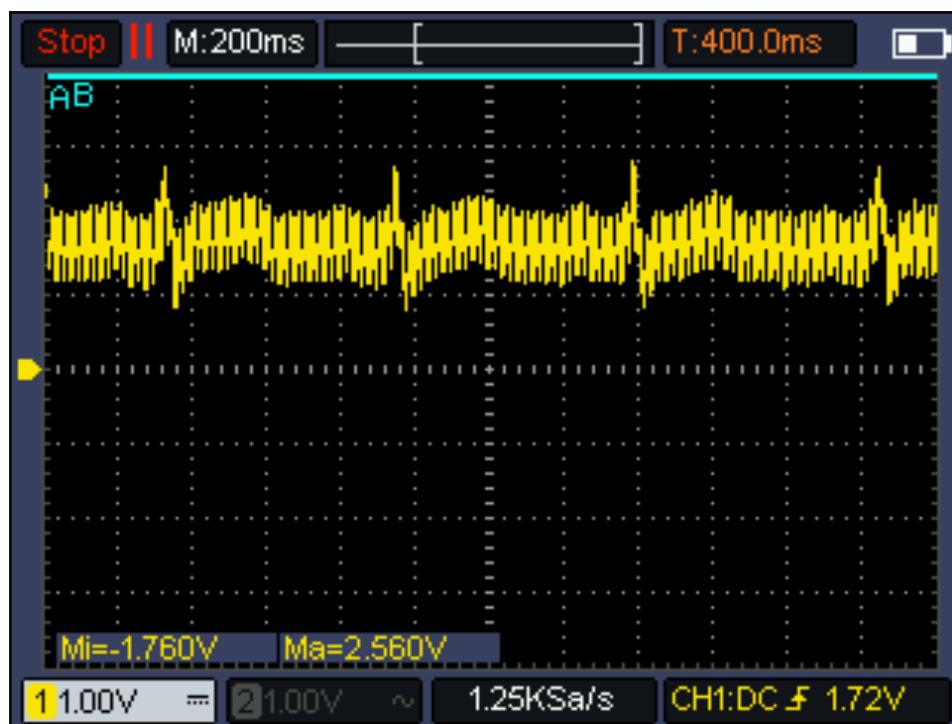


Figura 4.13: Sinal de saída do Sen-12650

4.5.1 Aplicação dos Filtros no STM22

Depois de determinados os coeficientes necessários através do Matlab, as Figuras 4.14, 4.15, 4.16 e 4.17 demonstram, respetivamente, a aplicação do filtro FIR passa-baixo, passa-alto e média móvel, bem como o filtro IIR *Notch*, resultando na aplicação simultânea dos mesmos demonstrada na Figura 4.18.



Figura 4.14: Filtro FIR passa alto

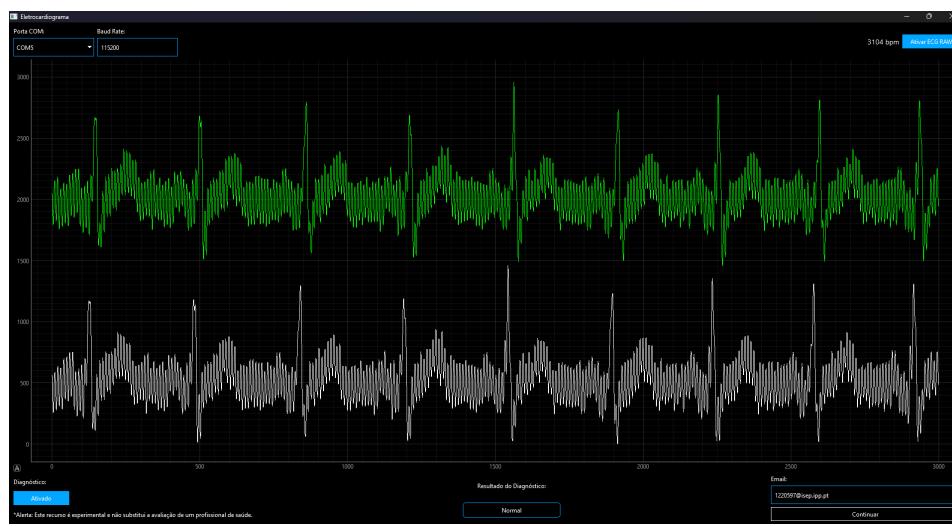


Figura 4.15: Filtro FIR passa baixo

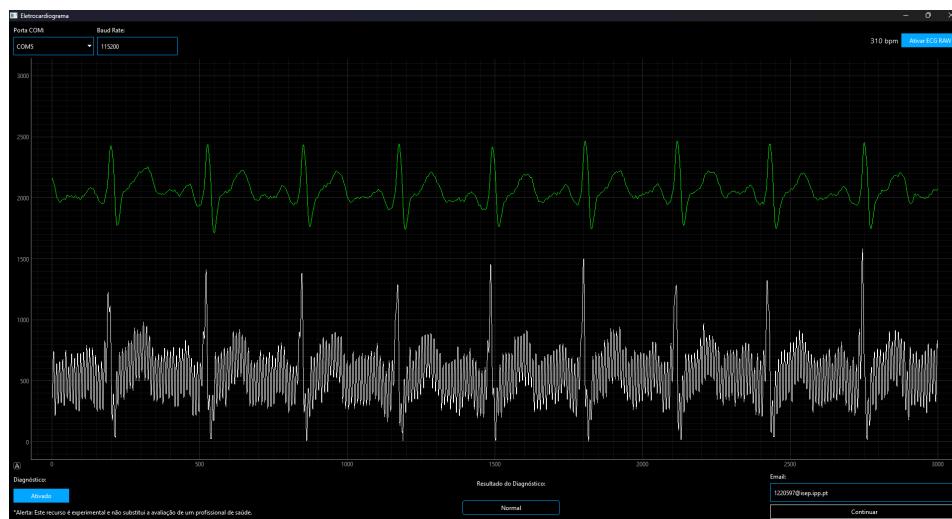


Figura 4.16: Filtro FIR média móvel

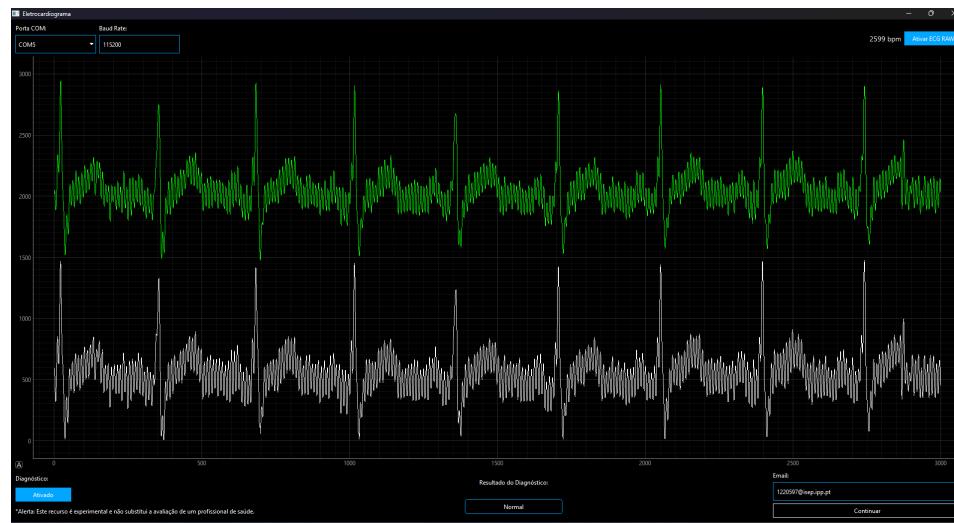


Figura 4.17: Filtro IIR Notch

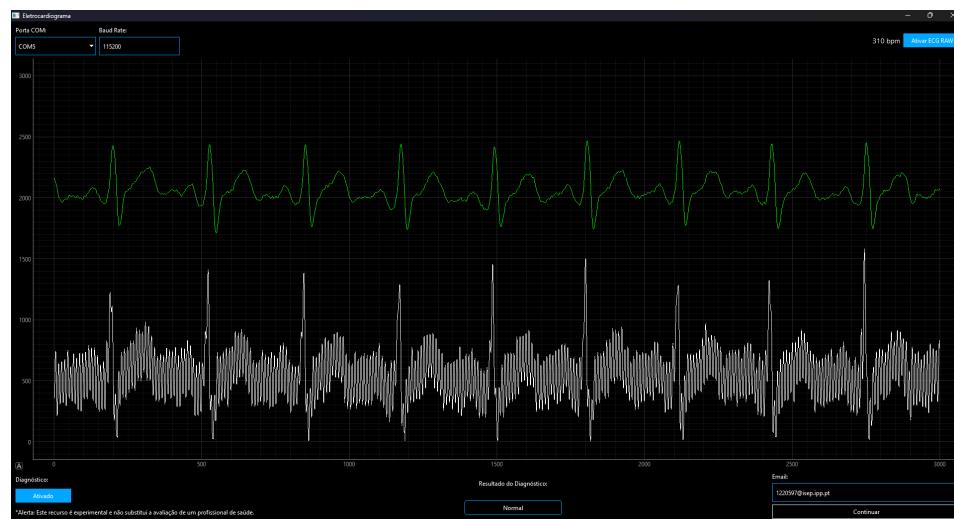


Figura 4.18: Resultado Final

Assim, comparando um ECG realizado em ambiente hospitalar, representado na Figura 4.19, com a Figura 4.18, é possível perceber que as características da onda II do ECG e do sinal obtido através do STM32 são as mesmas, e que a visualização do sinal está simplificada devido à baixa presença de ruído.

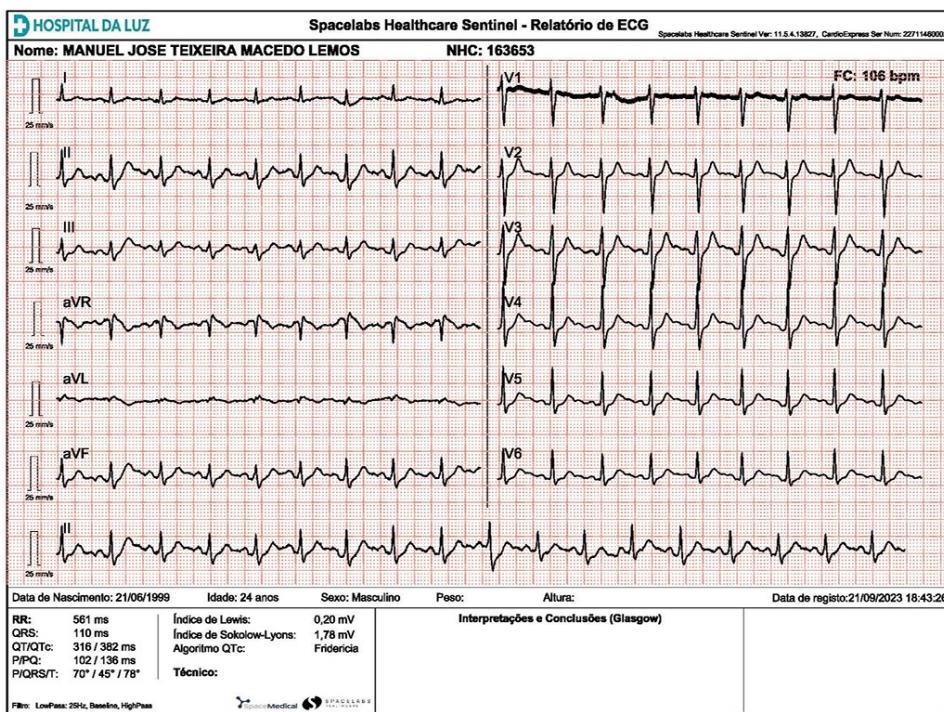


Figura 4.19: ECG realizado em ambiente hospitalar

Desta forma, podemos constatar que a aplicação dos filtros foi conseguida, na medida em que atenuou os ruídos indesejados visíveis na Figura 4.13.

4.5.2 Aplicação da AI

De modo a testar e a validar a inteligência artificial treinada, foram usadas métricas de validação. Quando a reta do *training accuracy* começa a entrar em regime permanente, significa que o treino acabou e por mais que se treine o modelo, o mesmo só irá aprender ruído, por isso é que é bom implementar o *callback EarlyStopping* já anteriormente referido. Esta maneira é a mais utilizada para saber se o treino está a ter bons resultados, ou se é necessário alterar o número de neurónios e camadas utilizadas, bem como épocas. Na Figura 4.20 está apresentada a *training accuracy* em termos de épocas. Na Tabela 4.2 são apresentados os resultados das métricas utilizadas para categorizar o treino.

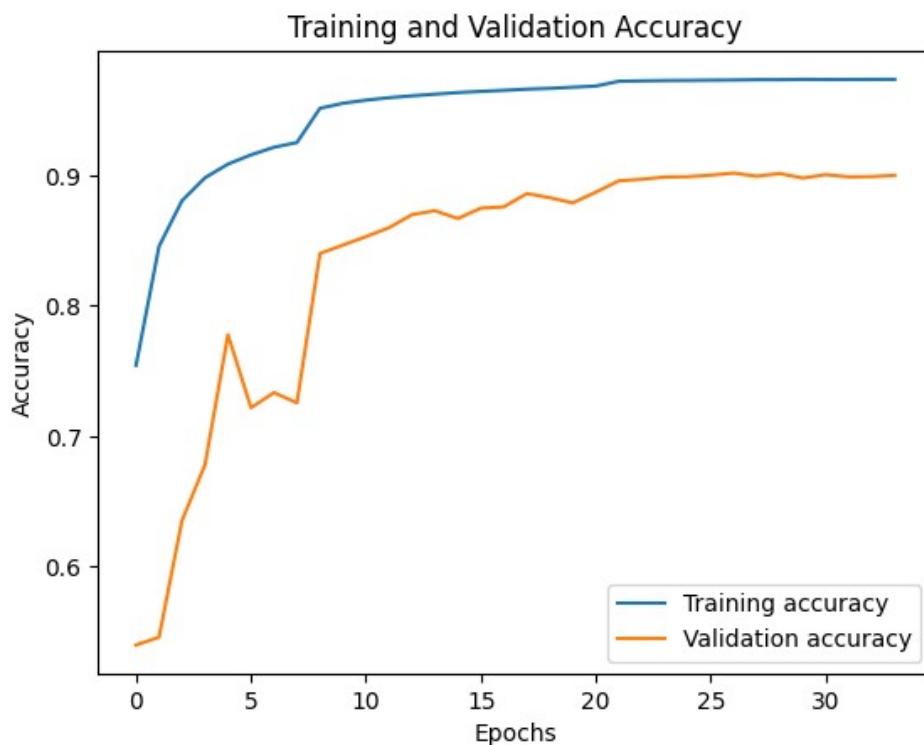


Figura 4.20: Gráfico de precisão de treino

Tabela 4.2: Métricas de desempenho do Modelo LSTM

Class	Precision	Recall	F1-Score	Accuracy
0	0.99	0.90	0.94	
1	0.93	0.99	0.96	
2	0.94	1.00	0.97	
3	0.64	0.93	0.76	
4	0.96	0.97	0.96	0.91
5	0.50	0.97	0.66	
6	0.73	0.96	0.83	
7	0.62	1.00	0.76	

Também é possível observar o comportamento do modelo através das perdas do mesmo, apresentadas na Figura 4.21 .

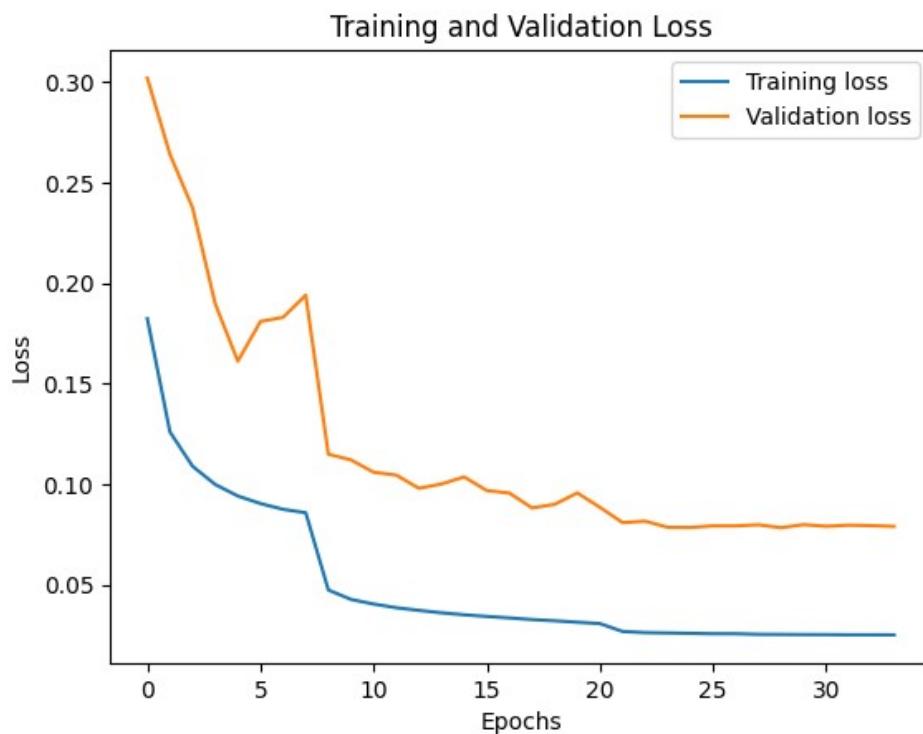


Figura 4.21: Gráfico da relação de perdas por época

Para além disto foi construída uma matriz de confusão, o que permite a validação do modelo, visto que é construído com base nos dados de teste, dados que o modelo ainda não teve contacto, onde a diagonal principal é os acertos do nosso modelo. A matriz de confusão referente ao modelo treinado está presente na Figura 4.22.

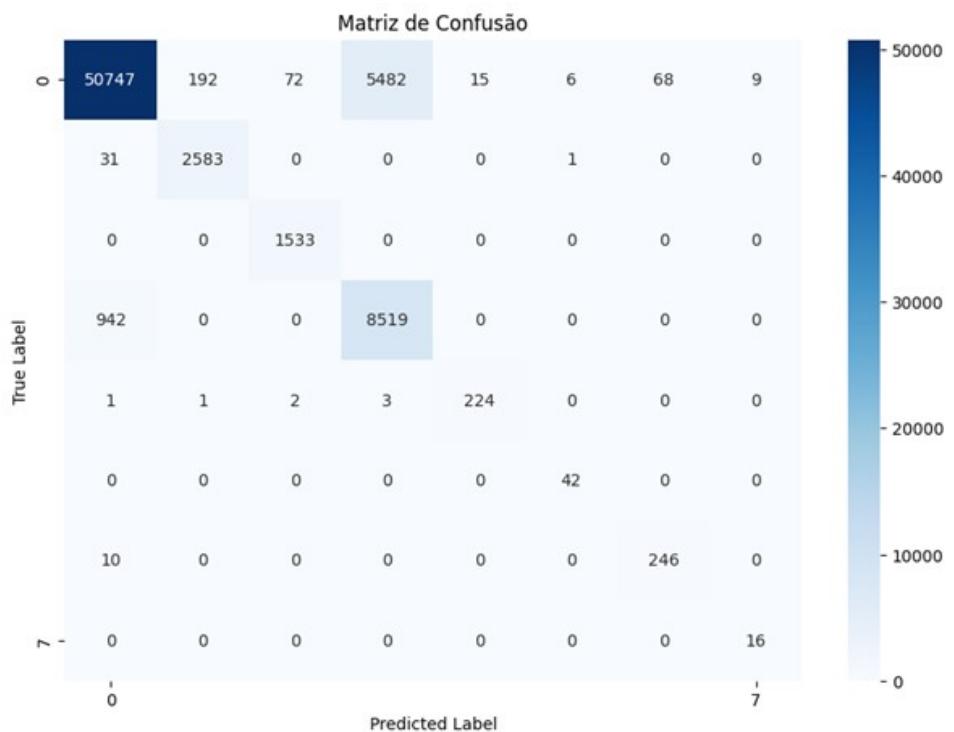


Figura 4.22: Matriz de Confusão

De modo a também testar o envio do email na nossa aplicação, foi alterado o método correspondente de modo a proceder uma verificação de enviar o email com o diagnóstico correto. Na Listagem 4.36 está presente a alteração efetuada e na Figura 4.23 é mostrado o resultado dessa mesma alteração.

```

1 if self.diagnosis == "Normal":
2     recipient = self.email_input.text()
3     subject = "Alerta de Doença Detectada"
4     body = f"A IA detectou uma possível doença no
5         eletrocardiograma. Diagnóstico: {self.diagnosis}"
6     send_email(recipient, subject, body)
7     # Desativar IA após enviar email
8     self.ia_button.setChecked(False)
9     self.ia_button.setText("Desativado")
10    self.ia_thread = None # Desassociar a thread de IA
11        após enviar o email

```

Listagem 4.36: Verificação do Diagnóstico

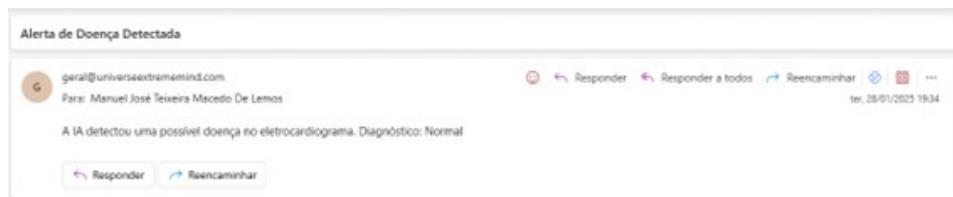


Figura 4.23: Confirmação da receção de email

Capítulo 5

Conclusão

Neste trabalho, foi desenvolvido e implementado um sistema completo para aquisição, processamento e análise de sinais de ECG, utilizando o módulo SEN-12650 (AD8232) e o microcontrolador STM32F767ZI. Além disso, uma aplicação em Python foi desenvolvida para demonstrar os sinais capturados, realizar a análise e enviar alertas automáticos com o auxílio de inteligência artificial.

Os objetivos propostos foram alcançados, conforme descrito a seguir:

- **Aquisição de Dados:** O módulo SEN-12650 foi integrado com sucesso ao microcontrolador STM32F767ZI, permitindo a colheita confiável dos sinais de ECG. O tratamento inicial dos dados incluiu a aplicação de filtros digitais para melhorar a qualidade do sinal:
 - **Filtro passa-alto:** Utilizado para remover componentes de baixa frequência, como ruídos de movimento e interferências da linha de base.
 - **Filtro passa-baixo:** Utilizado para eliminar ruídos de alta frequência, garantindo que apenas os componentes relevantes do ECG fossem preservados.
 - **Filtro de média móvel:** Aplicado para suavizar a amostra eliminando o ruído presente.
 - **Filtro notch:** Aplicado para mitigar interferências na frequência de 50 Hz, devido à presença de ruídos da rede elétrica.

Estas etapas asseguraram a integridade e a precisão dos dados analisados.

- **Visualização e Análise:** Uma aplicação desenvolvida em Python foi utilizada para exibir os sinais em tempo real, oferecendo uma interface intuitiva para visualização e interação. Esta ferramenta permitiu a análise detalhada do ECG e a identificação de padrões relevantes.
- **Inteligência Artificial:** Um modelo de inteligência artificial foi integrado no sistema, possibilitando a deteção automática de possíveis anomalias nos sinais cardíacos. Quando identificados padrões que indicam possíveis problemas de saúde, o sistema enviou notificações via email, demonstrando um fluxo completo de monitoramento e alerta.
- **Validação:** O sistema foi testado em diferentes condições, e a integração dos componentes demonstrou robustez e confiabilidade, desde a aquisição até a comunicação com os utilizadores.

Com este projeto, foi possível implementar uma solução acessível, eficiente e inteligente para monitoramento de ECG. O uso combinado dos filtros passa-alto, passa-baixo, média móvel e *notch* desempenhou um papel essencial no tratamento do sinal, garantindo dados de alta qualidade para a análise subsequente. Além disso, a integração do microcontrolador STM32F767ZI com inteligência artificial destacou-se como uma abordagem robusta e eficaz para sistemas biomédicos. Os resultados obtidos mostram a viabilidade técnica e funcional do sistema, oferecendo uma base sólida para futuras melhorias e aplicações em contextos reais de monitoramento de saúde.

5.1 Problemas Encontrados

Durante o desenvolvimento deste trabalho, vários desafios foram enfrentados que impactaram o progresso e a implementação do sistema. Um dos problemas mais significativos foi um defeito no módulo integrado SEN-12650. O módulo apresentou falhas intermitentes no sinal de saída, o que dificultou a análise precisa dos dados de ECG. Após investigações, foi identificado um problema na ligação do GND do módulo, sendo isto um defeito na soldagem do mesmo, o que atrasou parte do processo de desenvolvimento.

Outro desafio importante foi a aplicação dos filtros no sinal de ECG. Embora a teoria dos filtros passa-alto, passa-baixo e *notch* fosse clara, a implementação prática desses filtros no sistema foi complexa, devido à necessidade de ajustar os parâmetros de cada filtro para obter o melhor desempenho possível. Além disso, a configuração de filtros digitais em tempo real exigiu otimizações específicas no código, o que aumentou a complexidade da implementação.

A utilização de um modelo de AI para deteção de anomalias no ECG também apresentou desafios consideráveis. O treino do modelo foi uma tarefa difícil, principalmente porque os dados de ECG eram limitados e não estavam perfeitamente ajustados para os algoritmos utilizados. Conseguir que o modelo de AI treinasse corretamente para detetar anomalias específicas do nosso projeto foi um processo que exigiu várias tentativas e ajustes no conjunto de dados, bem como a modificação dos parâmetros do modelo (*EarlyStopping*, épocas, neurónios, etc.).

Além disso, um dos maiores obstáculos técnicos foi a implementação do DMA para otimização do processamento dos dados. A configuração correta do DMA foi complicada devido à necessidade de sincronizar a transferência de dados entre o microcontrolador e os *buffers* de memória.

Esses problemas foram superados através de testes contínuos, ajustes e otimizações no sistema, mas representaram desafios consideráveis que impactaram o desenvolvimento e exigiram uma abordagem sistemática para a resolução de cada um deles.

Perante isso houve duas tentativas de implementação da visualização dos batimentos cardíacos, através da deteção dos complexos QRS, o que não foi possível obter o resultado esperado.

5.2 Desenvolvimentos Futuros

Para futuras extensões deste projeto, existem várias melhorias e funcionalidades que podem ser implementadas para ampliar a aplicabilidade e a eficiência do sistema. Em primeiro lugar, seria interessante integrar conectividade sem fios, como WiFi ou Bluetooth, ao microcontrolador STM32F767ZI. Isso permitiria transmitir os dados do ECG em tempo real para dispositivos móveis ou para uma plataforma na nuvem, facilitando o acesso remoto e a análise dos dados.

Aplicar um método mais eficaz nos batimentos cardíacos de modo a ser o mais aproximado ao real possível.

Adicionalmente, o modelo de inteligência artificial pode ser aprimorado com a utilização de algoritmos mais avançados, como redes neuronais convolucionais ou modelos baseados em aprendizagem profunda, para melhorar a precisão na deteção de anomalias. Estes modelos poderiam ser treinados com um volume maior de dados para identificar não apenas irregularidades gerais, mas também condições cardíacas específicas, como fibrilação arterial ou taquicardia ventricular.

Outra direção possível seria o desenvolvimento de uma interface gráfica mais avançada para a aplicação em Python, incluindo a capacidade de armazenar e organizar os sinais capturados numa base de dados. Isso permitiria a criação de relatórios médicos personalizados e históricos detalhados de cada utilizador, facilitando a análise longitudinal do estado de saúde.

Por fim, o sistema poderia ser adaptado para monitoramento contínuo, com o uso de uma bateria de longa duração e sensores mais confortáveis para o utilizador, como elétrodos reutilizáveis ou dispositivos vestíveis. Isto tornaria o sistema viável para aplicações em ambiente clínico ou doméstico, contribuindo para a monitorização preventiva de pacientes com doenças cardiovasculares.

Estes desenvolvimentos não apenas ampliariam a funcionalidade do sistema, mas também fortaleceriam sua aplicabilidade em cenários reais, consolidando-o como uma ferramenta eficiente e acessível para monitorização de saúde.

Referências

- [1] “Statistics | eurostat.” [Citado na página 2]
- [2] “Statistics | eurostat.” [Citado nas páginas ix e 2]
- [3] R. Morgado, “Eletrocardiograma (ecg) - exame do coração.” Available at <https://www.saudebemestar.pt/pt/medicina/cardiologia/eletrocardiograma/>, Aug. 2021. (Last accessed in 11/10/2024). [Citado na página 3]
- [4] A. G. M. de Sousa, R. da Silva, W. de Andrade Langa, A. L. Rossetto, B. D. Rodrigues, J. C. S. Carvalho, A. K. de Oliveira Gonçalves, G. J. Lopes, H. Z. A. Mota, L. G. de Souza Jordão Paulino Santos, V. D. Araujo, A. G. A. Westry, L. F. Clementele, and A. V. M. de Lima, “Abordagem sobre a leitura do eletrocardiograma revisão de literatura,” *Brazilian Journal of Implantology and Health Sciences*, vol. 6, pp. 810–831, 4 2024. [Citado na página 3]
- [5] A. Haller, “Irritabilidade e sensibilidade : fisiologia e filosofia de,” 2002. [Citado na página 8]
- [6] W. B. Fye, “Journal of cardiology a history of the origin, evolution, and impact of electrocardiography.” [Citado nas páginas ix, 9, 10, 11 e 13]
- [7] L. M. de Caldas and M. J. de Araújo Faccin, “Universidades lusíada.” [Citado na página 13]
- [8] J. P. do Vale Madeiro, P. C. Cortez, J. L. Salinet, R. C. Pedrosa, J. M. da Silva Monteiro Filho, and A. R. A. Brayner, *Classical and modern features for interpretation of ECG signal*, pp. 1–193. Elsevier, 12 2018. [Citado nas páginas ix, x, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 25, 26, 27, 28, 29, 30, 31, 33, 34, 35, 36, 37, 38, 39 e 45]
- [9] P. D. Eletrocardiograma, D. Alexandre, and B. Rodrigues, “Desenvolvimento de um dispositivo,” 2013. [Citado nas páginas ix, x, 23, 24, 32, 33, 35 e 36]
- [10] A. J. C. Mitnacht, M. London, and D. L. Reich, *Electrocardiography*, p. 36–44. Cambridge University Press, 2011. [Citado nas páginas x e 31]
- [11] S. do eletrocardiograma na hipertrofia ventricular de acordo com gênero e massa cardíaca, “Sensibilidade do eletrocardiograma na hipertrofia ventricular de

- acordo com gênero e massa cardíaca,” *Arquivos Brasileiros de Cardiologia*, vol. 97, no. 3, pp. 225–232, 2011. Acesso em: 12 jan. 2025. [Citado nas páginas x e 39]
- [12] R. Wanzeler, “Como reconhecer o infarto no ecg,” 2019. Acesso em: 12 jan. 2025. [Citado nas páginas x e 40]
- [13] “Ecg200+ - cardioline.” [Citado nas páginas x, 40 e 41]
- [14] “Ecg 200+ 12 lead cord 12 channel machine by cardioline.” [Citado na página 42]
- [15] A. S. Prasad and N. Kavanashree, “Ecg monitoring system using ad8232 sensor,” in *2019 International Conference on Communication and Electronics Systems (ICCES)*, pp. 976–980, 2019. [Citado na página 43]
- [16] STMicroelectronics, “Stm32f765xx stm32f767xx stm32f768ax stm32f769xx: Arm® cortex®-m7 32-bit mcu+fpu, 462 dmips, up to 2 mb flash, 512+16+4 kb ram, usb otg hs/fs, 28 comm. int., lcd, dsi,” 2023. Acessado em 18 de novembro de 2024. [Citado nas páginas xiii e 49]
- [17] SparkFun Electronics, *Single-Lead, Heart Rate Monitor Front End*, Oct. 2022. Last accessed in 22/01/2025. [Citado nas páginas x, 52, 53 e 54]
- [18] A. Devices, “Ad8232: Single-lead, heart rate monitor front end data sheet,” 2012. Acessado em 18 de novembro de 2024. [Citado nas páginas xiii e 54]
- [19] F. Chollet, *Deep Learning with Python*. Shelter Island, NY: Manning Publications, 2021. [Citado na página 58]
- [20] “O que é a inteligência artificial e como funciona? | temas | parlamento europeu.” [Citado na página 58]
- [21] D. Sarkar, R. Bali, and T. Ghosh, *Hands-On Transfer Learning with Python: Implement Advanced Deep Learning and Neural Network Models Using TensorFlow and Keras*. Packt Publishing, 2018. [Citado nas páginas 58 e 59]
- [22] ARM, “Cmsis dsp software library.” Available at https://arm-software.github.io/CMSIS_5/DSP/html/index.html. Last accessed in 22/01/2025. [Citado nas páginas 67, 68 e 71]
- [23] “Difference between iir and fir filters: a practical design guide - asn home.” Available at <https://www.advsolned.com/difference-between-iir-and-fir-filters-a-practical-design-guide/>. Last accessed in 22/01/2025. [Citado nas páginas 71 e 80]
- [24] “Algoritmos dsp com a placa stm32f4discovery - parte ii.” [Citado na página 78]

- [25] “Mit-bih arrhythmia database v1.0.0.” Available at <https://physionet.org/content/mitdb/1.0.0/mitdbdir/#files-panel>. (Last accessed in 22/01/2025). [Citado na página 83]
- [26] “colab.google.” [Citado na página 83]

Anexo A

Esquemático Analógico

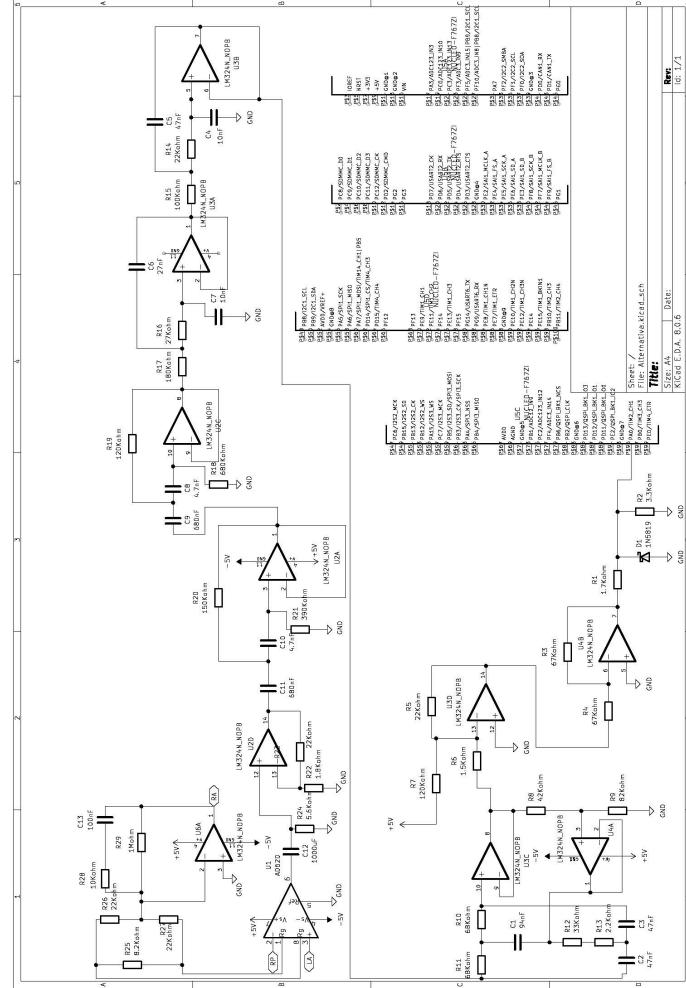


Figura A.1: Implementação analógica

Anexo B

Esquemático Digital

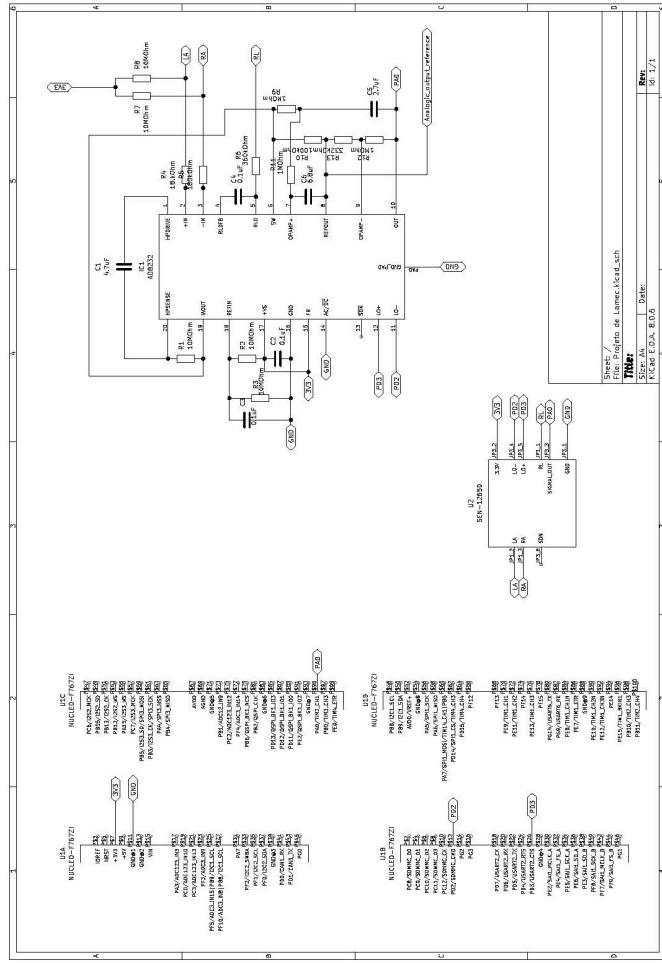


Figura B.1: Implementação digital

Anexo C

Circuito integrado do SEN-12650

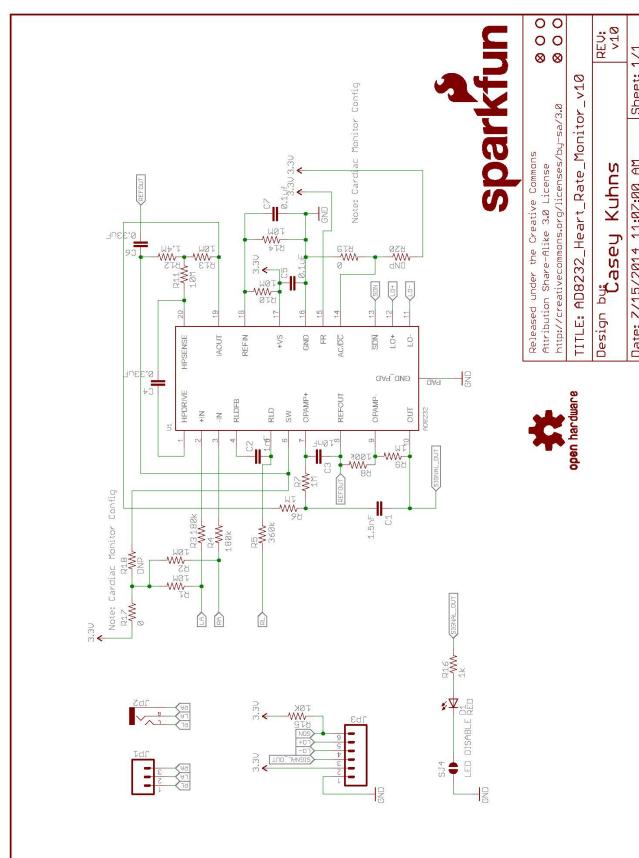


Figura C.1: Circuito integrado do SEN-12650