NOVA School of Science and Technology

Sistema Eletrocardiográfico

Daniel Levi (59809)
Diogo Amaro (59781)

Docente: Dr.José Luís Ferreira
Docente: Dr.André Wemans

Instrumentação Digital

Mestrado Integrado em Engenharia Biomédica

17 de junho de 2024

Conteúdo

Lista de Figuras

Lista de Tabelas

Introdução

Nos últimos tempos, os mecanismos de interpretação de sinais vitais têm vindo a ter um papel cada vez mais crucial na monitorização e no controlo do estado médico de um paciente, na medida em que permitem detetar, em tempo real, alterações súbitas e/ou anormais de certos parâmetros fisiológicos. No entanto, e apesar de eficazes, alguns destes métodos de análise carecem de certos parâmetros funcionais, tais como sistemas de alerta automáticos e/ou interfaces rudimentares, que podem atrasar a prontidão com que se responde a uma alteração fisiológica.

1.1 Objetivo

Assim, este projeto visa desenvolver um sistema de monitorização e alerta em tempo real do sinal cardíaco capaz de detetar rapidamente uma variação anormal no mesmo. Para tal, foi integrada uma placa de Arduino como fonte principal de deteção e respetiva interpretação do sinal cardíaco, juntamente com LabVIEW que fornece a componente visual do projeto.

1.1.1 Sinal de teste

Numa fase inicial, foi utilizado o gerador de sinais de bancada para produzir um sinal pulsado que serviu para testar toda a cadeia de instrumentação, desde a entrada do conversor analógico-digital da placa Arduino até à interface visual do computador. Tem como objetivo simular um sinal cardíaco de forma a garantir que o sistema está funcional durante toda a sua execução.

1.1.2 Sinal real

Depois de garantir o funcionamento correto de todo o sistema digital, o sinal cardíaco real é obtido por via do sensor ecgPlux, que é ligado diretamente à placa de Arduino e que permite a aquisição de um sinal eletrocardiográfico real.

1.1.3 Análise de dados

Depois de registado, o sinal é exportado para uma folha de cálculo com o objetivo de determinar as suas principais características.

1.2 Visão Geral

Devido à complexidado do projeto, foi necessário segmentá-lo em várias etapas mais simples de forma conseguir controlar e minimizar os erros. Esta segmentação de etapas permite também a segmentação de erros, de maneira que a identificação de erros e a respetiva resolução tornam-se mais expeditos.

1.2.1 Integração de Hardware

Esta etapa foca-se unicamente na funcionalidade da placa de Arduino. Envolve desenvolver o código que permite detetar e interpretar o sinal cardíaco e, de seguida, enviar uma série de instruções e de medidas específicas para a interface do computador (LabVIEW), como por exemplo os batimentos por minuto (BPM). Além disto, incluiu-se também um circuito eletrónico numa *breadboard* onde se integraram os LEDs que, numa fase posterior, permitem a visualização dos vários estados do sinal.

1.2.2 Desenvolvimento de Software

Aqui, o objetivo passa por desenvolver uma interface que permita ao utilizador visualizar as instruções e os dados provenientes do Hardware e do sinal, respetivamente. Permite também a interação do utilizador com o sistema, nomeadamente através da forma de aquisição de dados ou das instruções dadas ao sistema. Esta interface foi desenvolvida em LabVIEW.

1.2.3 Integração conjunta

Esta etapa mostrou ser a mais importante durante todo o desenvolvimento do projeto. Aqui, o objetivo é garantir que, mais importante do que estarem funcionais em separado, tanto o Hardware como o Software estão funcionais em conjunto. Coordenar de forma fluída o envio de instruções e de medidas por parte do primeiro com a receção destas mesmas instruções e medidas por parte do segundo, garante não só uma aquisição eficaz de dados como também uma representação fidedigna do sinal que está a ser lido.

1.2.4 Teste

Esta etapa surge com o objetivo de simular uma situação real de leitura do sinal cardíaco de forma a garantir que todo o sistema se encontra operacional e capaz de recolher um sinal real. Como referido na secção **1.1.1**, esta fase utiliza um gerador de sinais que produz um sinal semelhante ao sinal cardíaco real e segue uma série de características previamente estabelecidas que serão explicadas em detalhe nas secções seguintes.

Metodologia

2.1 Fluxograma

As etapas do projeto encontram-se sequencialmente representadas na figura seguinte.

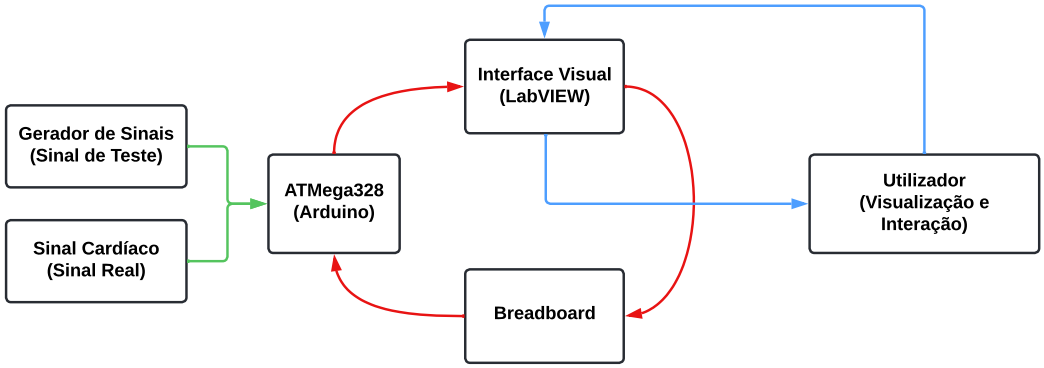

Figura 1: *Fluxograma*

Como é possível verificar, a integração conjunta entre o Hardware (Arduino+Breadboard) e o Software (LabVIEW) é a base de funcionamento do sistema. Qualquer disrupção numa destas ligações compromete automaticamente a aquisição do sinal cardíaco.

2.2 Sinais de entrada

Para efeitos de teste, foi utilizado um sinal produzido pelo gerador de sinais com características específicas de forma a não danificar o microcontrolador. Gerou-se então um sinal quadrado, com um *duty cycle* de 10%, que corresponde ao rácio entre o tempo segundo o qual o circuito está ligado comparativamente ao tempo segundo o qual está desligado, e uma frequência de 2Hz. Estabeleceu-se também um valor de tensão médio de 2.5V e uma tensão pico-a-pico de 4V. Como valores de tensão negativos danificam o microcontrolador, optou-se por um *DC offset* de 0.5V, daí o valor médio de 2.5V. Numa fase final, ou seja, com o sistema totalmente funcional, este sinal de entrada é substituído por um sinal cardíaco real.

2.3 Hardware

2.3.1 Arduino

O Hardware utilizado foi uma placa de Arduino com o microcontrolador ATMega328 integrado na placa. No IDE do Arduino, implementou-se o código que permite controlar todas as operações que o sistema deve realizar, nomeadamente o controlo da taxa de amostragem e o cálculo de BPM. Além disto, inclui-se também um modo automático e um modo manual. Isto teve como objetivo permitir ao utilizador controlar o sistema operativo: em modo automático, é o próprio program que, consoante os valores de BPM, aciona os LEDs correspondentes como sinal de alerta, enquanto que em modo manual, independentemente dos valores de BPM, é o utilizador que define qual o sinal luminoso que ativa. De notar que em modo manual, os valores e BPM continuam a ser mostrados na interface visual, apesar de não terem influência no estado dos LEDs.

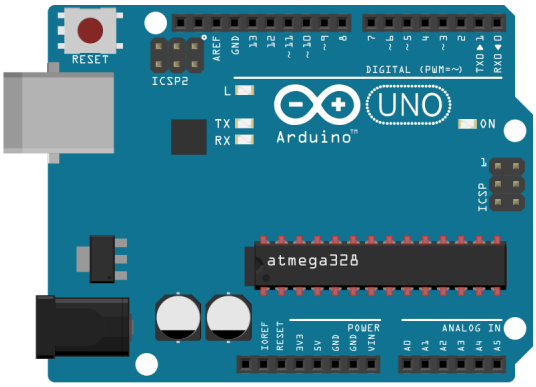

Figura 2: *Placa de Arduino com ATMega328 integrado*

A coordenação com os LEDs da *breadboard* foi efetuada através das portas digitais 10, 8, 5 e 6. Os sinais de teste e real foram introduzidos com recurso à porta analógica A0 e ao GND.

2.3.2 Breadboard

A montagem da *breadboard* foi feita com o objetivo de conseguir integrar sinais luminosos no projeto e estabelecer uma ligação quer com a placa de Arduino, quer com a interface visual. Isto deve-se à possibilidade de um modo automático ou visual, como referido na secção **2.3.1**, já que em modo automático é o Arduino a controlar o estado dos LEDs e, em modo manual, é o utilizador que realiza esta operação. Para a contrução do circuito na breadboard, foram utilizados quatro LEDS luminosos, cada um deles ligado em série com uma resistência de 220Ω.

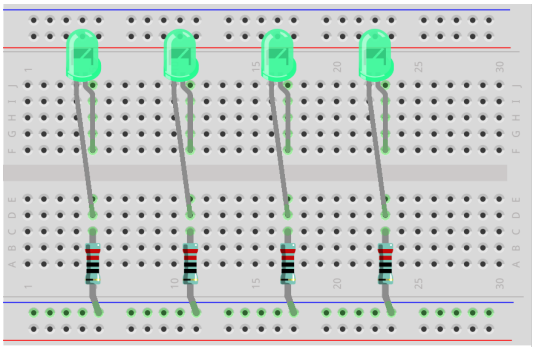

Figura 3: *Circuito com os LEDs*

Desta forma, e depois de coordenado com o Arduino, cada LED irá ligar consoante o valor de BPM calculado. As resistências tem como objetivo limitar a corrente que atravessa cada um dos LEDs. Isto porque cada *pin* no ATMega328 suporta, aproximadamente, uma corrente entre 20-40mA. Assim, limitar uma possível corrente excessiva com uma resistência de 220Ω pode evitar danos no microcontrolador.

$$R(\Omega) = \frac{V_{\text{in}} - V_{\text{LED}}}{I_{\text{LED}}} = 220\Omega$$

Onde V_{in} é a tensão de entrada (5V), V_{LED} a queda de tensão de um LED verde ($\approx$ 2V) e I_{LED} a corrente que atravessa o LED, que se considerou ser entre 10-15mA.

2.3.3 Circuito funcional

Depois de ambos os componentes programados, realizou-se a ligação entre eles. Os LEDs ligaram-se às portas digitais 10, 8, 5 e 6 da placa de Arduino e o sinal de entrada (gerador de sinais) à porta analógica A0.

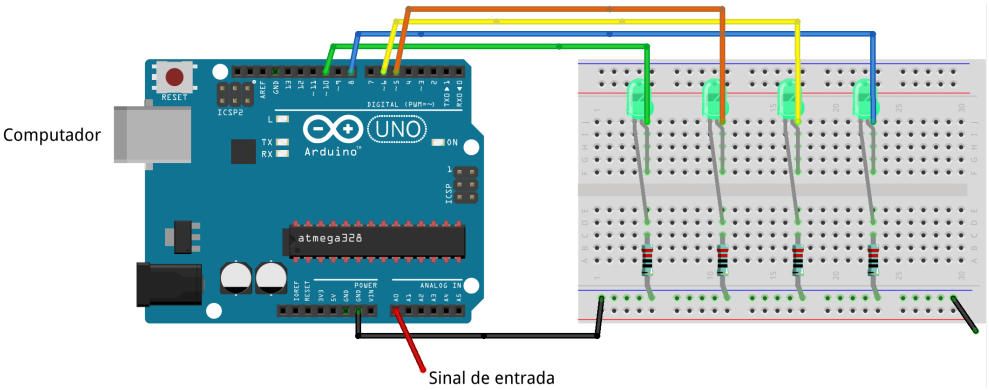

Figura 4: *Ligação da placa de Arduino com a breadboard*

Desta forma, as instruções definidas no código de Arduino já estão em condições de serem visualizadas na *breadboard*, através dos LEDs. Para um valor de BPM superior a 160, o primeiro LED, ou seja, aquele que está ligado à porta digital 10, acende. No contexto do projeto, este LED representa a indicação para uma possível injeção de propanolol no paciente. As instruções para os restantes LEDs estão representados na Tabela 1.

Tabela 1: *Critérios de ativação dos LEDs*

Valor de BPM	**LED Ativo**	**Porta Digital**
BPM $>$ 160	LED 1 (Propranolol)	10
30 $<$ BPM $\leq$ 40	LED 2 (Atropina)	5
10 $<$ BPM $\leq$ 30	LED 3 (Pacemaker)	6
BPM $=$ 0	LED 4 (Desfibrilhador)	8

De acrescentar que a ativação do LED 4 é ligeiramente diferente dos restantes. O desfribrilhador foi programado para ativar apenas se o valor de BPM for nulo durante pelo menos 10 segundos, para garantir que não foi nenhum erro de leitura ou cálculo e que o paciente está efetivamente com valores anormais. Além disto, o desfibrilhador fica ativo durante 5 segundos, independentemente de entretanto serem detetados valores de BPM normais.

2.4 Software

A interface de interação e/ou visualização foi desenvolvida em ambiente LabVIEW. Aqui, o objetivo passa por permitir ao utilizador visualizar os dados que estão a ser recolhidas que, neste caso, se tratam de leituras do sinal cardíaco e de permitir ainda que o utilizador assume o controlo do programa, na medida em que passa a ser capaz de influenciar diretamente os estados dos LEDs, ou seja, que medida/substância se deve administrar no paciente.

2.4.1 Painel de ligação ao Arduino

Primeiramente, foi construída uma página que permite carregar o programa de Arduino para o ambiente LabVIEW. Isto é feito através do *VISA Resource Name* que funciona como um identificador único para uma sessão de I/O num dispositivo. Quer isto dizer que, atribuir um valor a um *VISA Resource Name*, significa dizer que o LabVIEW está conectado a um dispositivo externo (Arduino) através desse valor. Neste caso, trata-se de uma porta COM, atribuída automaticamente pelo computador aquando da ligação do Arduino. Adicionalmente, utiliza-se também a função *VISA Read*, que permite que o LabVIEW consiga enviar pedidos para o Arduino, receba a resposta a esses pedidos e interprete essa resposta para poder ser executada em LabVIEW.

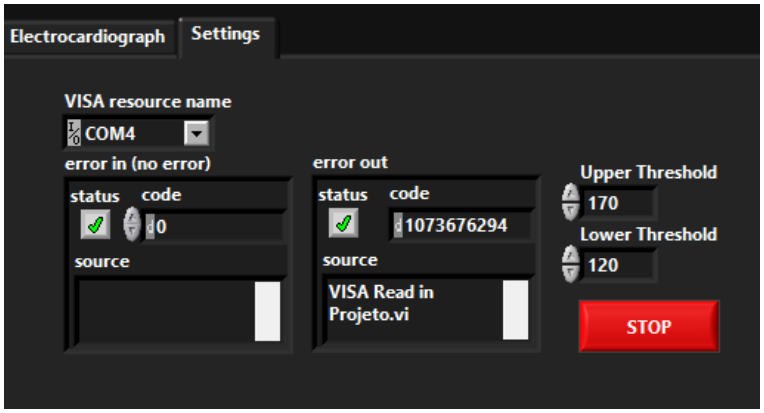

Figura 5: *Página Settings*

Foram também integrados dois botões ajustáveis para definir os limites superior e inferior para deteção de uma onda R do sinal cardíaco, que é explicado com maior detalhe na secção **2.4.2**

2.4.2 Painel Principal

A maior parte da interface está representada no painel principal. É aqui que acontece todo o processo de visualização e interação por parte do utilizador.

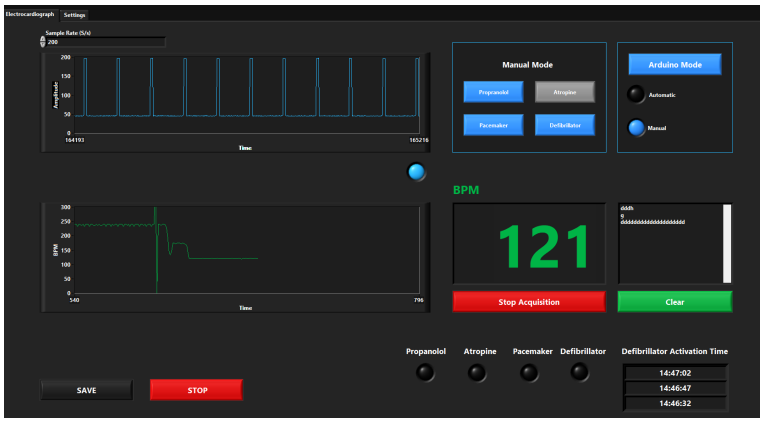

Figura 6: *Vista geral do painel principal*

Taxa de amostragem

Inicialmente, e antes de iniciar a aquisição do sinal cardíaco, o utilizador tem a opção de definir qual a taxa segundo a qual vai amostrar o sinal. Esta ação é controlada pelo microcontrolador que, consoante o caracter recebido, devolve a taxa de amostragem correspondente.

Tabela 2: *Definição da taxa de amostragem*

Comando enviado (PC)	Código ASCII	Taxa de amostragem (S/s)
'a'	97	25
'b'	98	50
'c'	99	100
'd'	100	200

De seguida, o utilizador fica em condições de visualizar, num gráfico Amplitude x Tempo, o sinal em tempo real.

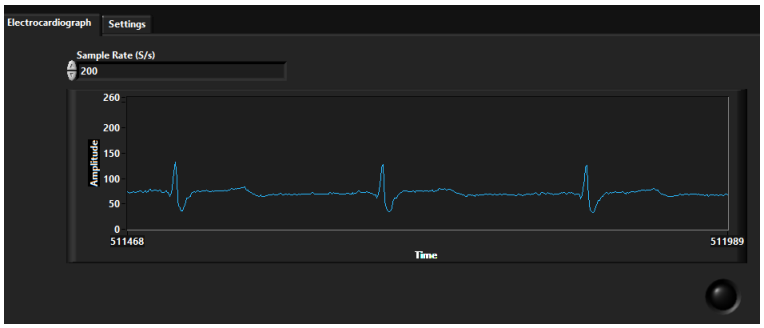

Figura 7: *Gráfico Amplitude x Tempo com a taxa de amostragem escolhida*

Uma funcionalidade extra deste gráfico prende-se com a presença de um botão no canto inferior direito, que ativa um sinal luminoso sempre que é detetada uma onda R. Os botões referentes aos limites, representados na figura 4 da secção **2.4.1**, servem para esse efeito. Devem escolher-se dois valores de limite superior e inferior que atravessem completamente o pico R.

Visualização da evolução dos valores de BPM

Um segundo gráfico foi inserido no painel principal com objetivo de visualizar a evolução dos valores de BPM ao longo do tempo.

Figura 8: *Gráfico BPM x Tempo*

O gráfico pode ajudar o utilizador na deteção de algum tipo de variação anormal do ritmo cardíaco bem como o instante em que essa variação ocorreu.

Modo Automático e Manual

Enquanto sistema de visualização e interação, o utilizador tem a opção de definir se os alertas luminosos dados pelos LEDs derivam da decisão do Arduino, ou se derivam dele próprio. No modo manual, é o utilizador que, de acordo com o valor de BPM que visualizar no painel principal, toma a decisão final relativamente à substância a administrar no paciente. Isto porque os limites de BPM definidos para o Arduino podem não servir como regra para todos os pacientes. Considere-se, por exemplo, que um paciente se encontra com um valor de BPM de 158. Segundo o código estabelecido para o Arduino, este paciente encontra-se em pefeito estado, já que o sistema não ativou nenhum dos LEDs. No entanto, um valor de BPM de 158 em repouso é altamente anormal, pelo que o médico, enquanto utilizador, pode optar por administrar uma injeção de propanolol, acionando manualmente o botão 'Propanolol' do *Manual Mode* que ativa o LED 1.

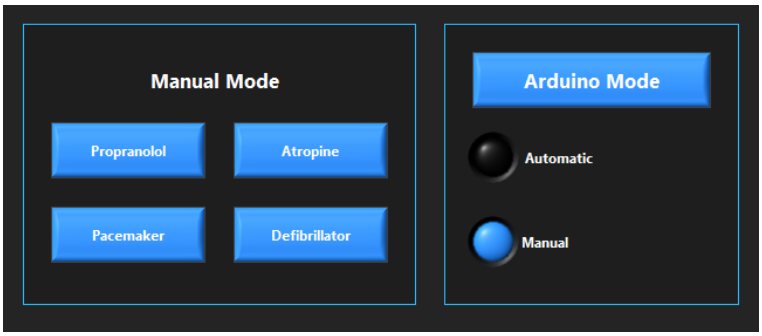

Figura 9: *Possibilidade de transição entre modo automático e manual*

A escolha entre o modo manual e o modo automático depende do quão díspar é a análise crítica do médico à situação relativamente às instruções estabelecidas para o Arduino para essa mesma situação.

Visualização dos valores de BPM

Bem destacado no painel principal encontra-se uma zona que exibe o valor de BPM por cada instante de tempo. É segundo este valor que o Arduino, em modo automático, controla o estado dos LEDs.

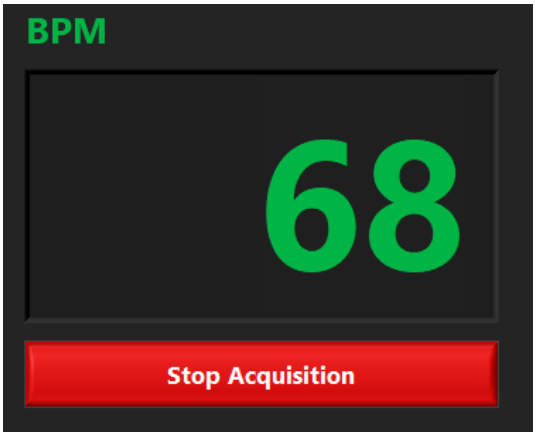

Figura 10: *Visualização do valor de BPM em cada instante*

Por baixo dessa zona, existe ainda o botão que permite ao utilizador iniciar ou interromper a aquisição do sinal. De notar que, sempre que o utilizador interrompe a aquisição do sinal e volta a iniciá-la, é necessário definir manualmente a taxa de amostragem, já que a taxa *default* do programa é de 0 S/s, ou seja, não adquire amostra nenhuma.

Visualização dos caracteres enviados

Esta região do painel principal tem uma função meramente observacional e tem como objetivo mostrar os caracteres que estão a ser enviados para a *Serial Port* do Arduino através do *Serial Monitor*. Isto porque o microcontrolador está programado para terminar imediatamente a aquisição e a comunicação das medidas ao computador se não for enviado nenhum dos caracteres 'a', 'b', 'c' ou 'd' num intervalo de 2 segundos.

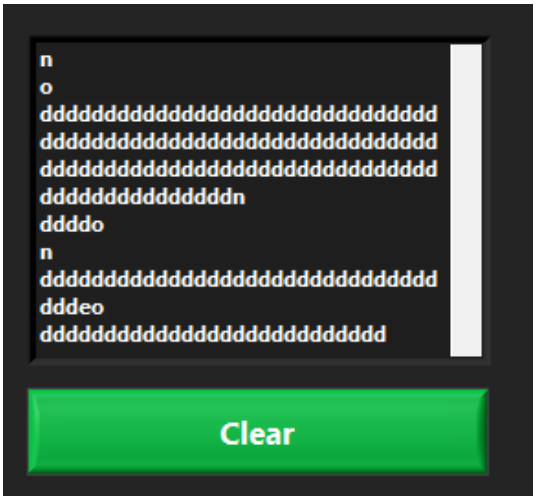

Figura 11: *Visualização dos caracteres enviados à Serial Port*

Assim, para descartar um possível erro derivado dessa funcionalidade, incluiu-se uma

região do painel principal para a visualização desses mesmos envios.

Visualização do estado dos LEDs

Na zona inferior do painel principal estão representados os indicadores que mostram ao utilizador o estado de um determinado LED nesse momento: se uma variação dos valores de BPM acionar um LED na *breadboard*, então esse fenómeno acontece simultaneamente no painel principal, para que o utilizador consiga percecionar, em tempo real, como varia o ritmo cardíaco do paciente e como deve atuar perante essas variações.

Figura 12: *Visualização do estado dos LEDs*

O caso específico representado na figura acima serve para comprovar a observação feita na secção **2.3.3**, que estabelece que o desfibrilhador fica ativo durante 5 segundos independentemente da deteção de valores de BPM nesse intervalo. Esta funcionalidade é importante porque prova que existe uma anormalidade no ritmo cardíaco do paciente, dado que o desfibrilhador só é ativado se forem detetados valores de BPM próximos de zero durante 10 segundos, ou seja, descarta a hipótese de ter sido uma má leitura por parte do sistema. É muito improvável que o sistema efetue leituras erradas durante 10 segundos e, além disso, que essas más leituras sejam todas de valor 0.

Terminar e Guardar

Por último, o programa inclui também dois botões que permitem ao utilizador guardar as leituras efetuados sob a forma de um ficheiro **.xlsx** e terminar a execução do ambiente LabVIEW.

Figura 13: *Terminar e Guardar*

2.4.3 Funcionalidades Extra

Indicador luminoso do modo atual do programa

O primeiro extra implementado prende-se com um sinal luminoso referente ao modo do programa num determinado instante. Se estiver a operar em modo manual, o indicador

Manual estará ativo. Caso contrário, é o indicador **Automatic** que se encontrará ligado.

Figura 14: *Visualização do modo operacional do programa (Extra 1)*

Desativação de botões no modo automático

Para impedir que o utilizador seja capaz de influenciar o funcionamento do sistema mesmo em modo automático, integrou-se a desativação dos botões referentes aos LEDs quando o sistema se encontra no modo automático. Para tal, foi criada uma **case structure** com uma condição que, sempre que verdadeira, o botão desativa. A condição garante que o modo automático está ligado e a desativação do botão fez-se através da criação de uma **Local Variable** com o parâmetro **Property Node** que, enquanto o modo automático está ligado, assume a especificação **Disabled and Greyed Out**.

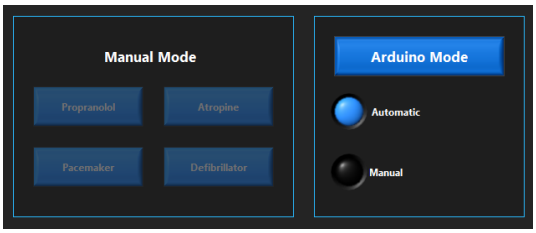

Figura 15: *Estado dos botões em modo automático (Extra 2)*

Quando a condição deixa de se verificar, ou seja, quando o modo manual é ativado, os botões voltam ao estado normal.

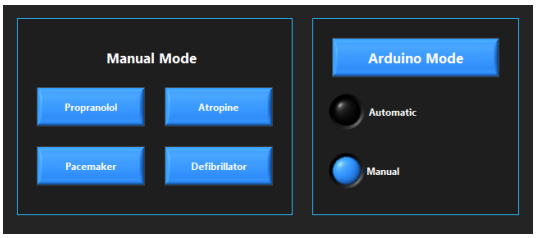

Figura 16: *Estado dos botões em modo manual (Extra 2)*

Ativação de um único botão no modo manual

Ao contrário do que acontece no modo automático, o modo manual não permite que o utilizador acione dois ou mais LEDs em simultâneo. Esta funcionalidade impede que estejam a ser dadas instruções contraditórias, como por exemplo LED do propanolol e do desfibrilhador ligados em simultâneo.

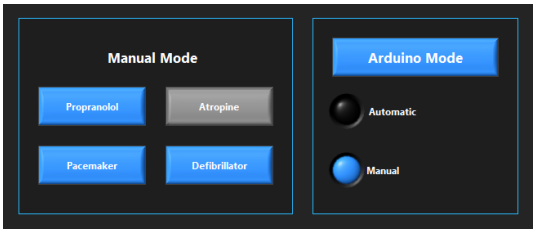

Figura 17: *Ativação de um único botão de cada vez (Extra 3)*

Para obter esta funcionalidade em LabVIEW, o objetivo é forçar, por exemplo, a desativação dos LEDs referentes à atropina, pacemaker e desfibrilhador quando o LED referente ao propanolol está ativado. Replicou-se esta lógica para todos os botões.

Instantes de ativação do desfibrilhador

Por último, considerou-se relevante a adição de uma tabela que mostre ao utilizador os 3 últimos instantes segundo os quais o LED do desfibrilhador ligou. A tabela regista estes instantes sob o formato **hh:mm:ss**. Esta informação é importante na medida em que permite

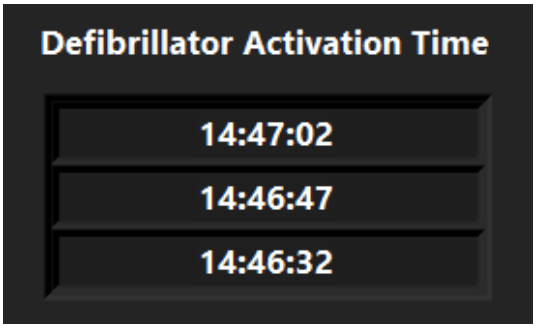

Figura 18: *Instantes de ativação do desfibrilhador (Extra 4)*

ao utilizador ver se o desfibrilhador esteve ligado em algum instante, pois este só está ligado durante 5 segundos e o médico pode não ter detetado a ativação deste LED nesse período de tempo. Permite ainda que o médico possa reconhecer certos padrões ou tendências no estado de saúde do paciente.

Resultados

3.1 Sinal de Teste

Numa primeira fase, o sistema foi testado com um sinal produzido pelo gerador de sinais, cujas características se encontram enumeradas na secção **2.2**. Após efetuar a leitura deste sinal de teste, utilizou-se o botão **Save** do painel principal e as leituras foram exportadas para uma folha de cálculo.

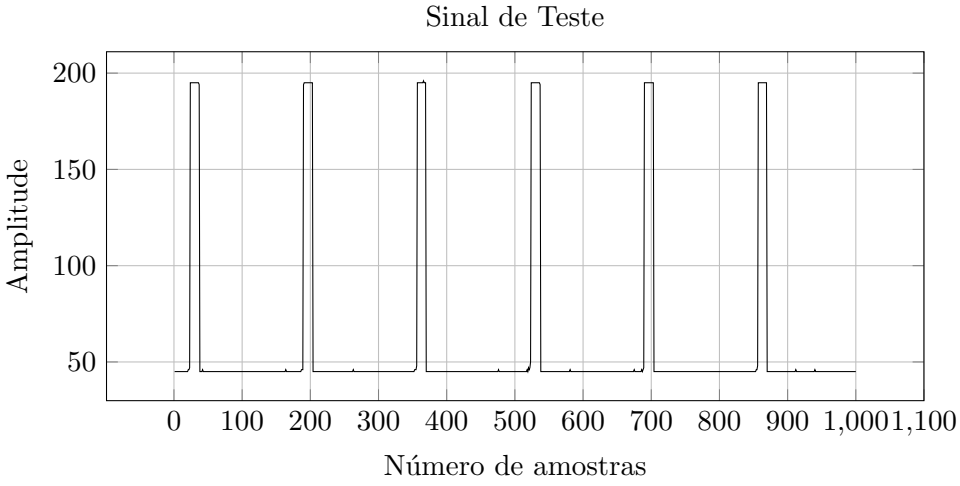

Figura 19: *ECG obtido com o sinal de teste*

Como é possível verificar, as características estabelecidas para o sinal de teste estão todas presentes no gráfico. Com efeito, verifica-se um *duty cycle* de aproximadamente 10% e um valor de *DC Offset*.

3.2 Sinal Real

Depois de realizar o teste ao sistema com o sinal de teste, repetiu-se o processo com um sinal cardíaco real. Com recurso a três elétrodos *ecgPlux*, o docente ligou-se ao sistema através da porta analógica A0 e procedeu-se então à aquisição do sinal.

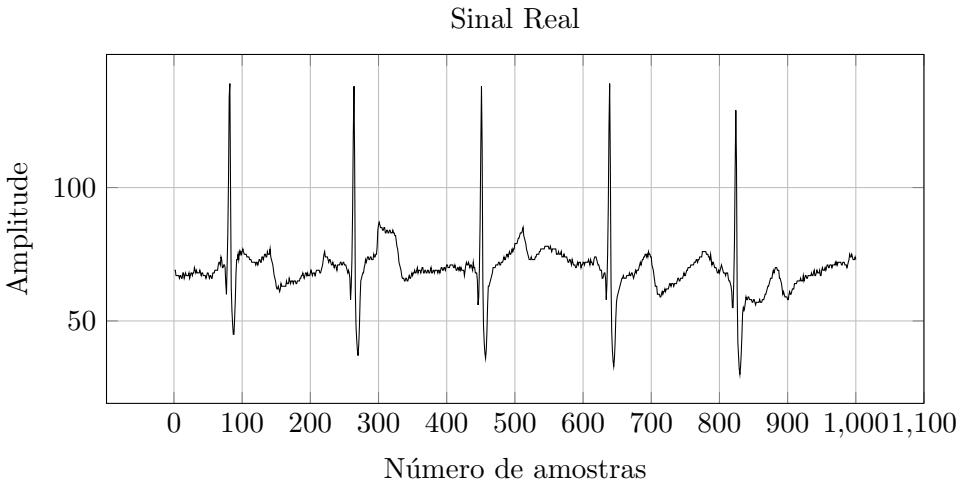

Figura 20: *ECG obtido com o sinal real*

Tal como era esperado, o sinal foi recolhido na sua plenitude e sem erros, nomeadamente descontinuidades ou más leituras. Assim, gerou-se um sinal típico de ECG, onde as suas principais características são facilmente detetadas, nomeadamente o complexo QRS e as respetivas ondas R.

Discussão

Apesar do desenvolvimento do projeto ter decorrido sem grandes percalços, surgiram, no entanto, alguns obstáculos cuja resolução exigiu mais tempo. A primeira surgiu quando o grupo tentou instalar a funcionalidade do desfibrilhador desligar ao fim de 5 segundos. Durante este intervalo de tempo, se o sistema não detetasse nenhum valor de BPM, então o desfibrilhador desligava no fim dos 5 segundos e o programa não aparentava ter erros. No entanto, caso o sistema reconhecesse algum valor de bpm no intervalo de tempo em que o desfibrilhador estava ligado, então este ficava ligado infinitamente durante o restante tempo de execução do programa. Este problema resolveu-se com a atualização do código de Arduino, através da criação de uma variável de estado capaz de isolar o funcionamento de desfibrilhador das restantes variáveis, funcionando assim de forma independente.

O segundo obstáculo prendeu-se com a atualização dos valores de BPM. Um teste típico realizado ao longo do projeto foi o de desligar o gerador de sinais para simular um BPM de 0, ou seja, simular a morte do paciente. Isto era feito repetidamente para testar o funcionamento do desfibrilhador. No entanto, e apesar do desfibrilhador responder de forma correta a este teste (esperar 10 segundos e ligar durante 5 segundos), acontece que ao voltar a ligar o gerador de sinais, o sistema não estava a atualizar os valores de BPM com as novas medições, mas sim a mostrar medições antigas, ou seja, medidas correspondentes às leituras efetuadas enquanto o gerador de sinais esteve desligado. Isto impediu o sistema de ler corretamente o sinal e, consequentemente, de dar as instruções corretas ao nível dos LEDs. Mais uma vez, este problema foi resolvido com uma alteração no código. Ao forçar um *reset* na variável BPM (variável que guarda os valores de BPM) sempre que o sistema ativava o desfibrilhador, foi possível fazer com que valores de BPM antigos fossem substituídos pelos valores atuais, permitindo que o sistema efetuasse uma leitura em tempo real, mesmo depois de desligado o gerador de sinais.

Conclusão

De forma geral, o objetivo fundamental do trabalho foi alcançado. Conseguiu-se desenvolver todo um sistema eletrónico do início ao fim, desde a montagem do circuito na *breadboard* até à interface visual em LabVIEW. Conseguiram-se também integrar dois modos: automático e manual, que permitem ao utilizador selecionar a forma de operação do sistema. Os LEDs da *breadboard* funcionam na sua totalidade, isto é, ligam apenas para certos valores de BPM, tal como especificado no código de Arduino. Os indicadores luminosos de LabVIEW (figura 11 da secção **2.4.2**) também funcionam corretamente na medida em que estão ativos exatamente nos mesmos instantes em que o LED correspondente na *breadboard* está ativo. Por último, consegui-se ainda integrar algumas funcionalidades extra que se consideraram vantajosas para o contexto do projeto.