Pacemaker Sensível à Frequência Cardíaca para o Tratamento de B. AV. *Mobitz* Tipo II

Duarte Rodrigues, Ricardo Brioso, Mariana Xavier

Resumo — O objetivo deste projeto é a criação de um *pacemaker* simulado que envie um impulso elétrico para o coração sempre que falhe um batimento num sinal de ECG real. Mais concretamente, são tratados sinais que apresentem patologia B. AV. *Mobitz* Tipo II. O projeto é realizado em *LabView* em cooperação com *Multisim* e ao longo deste relatório é explicado o seu funcionamento.

Palavras-chave – *pacemaker*, impulsos, eletrocardiograma, arritmia, *LabView*, *Multisim*, *cosimulação*.

I. Introdução

O elemento do coração responsável por gerar o impulso elétrico normal, que permite que o miocárdio contraia e relaxe de forma iniciar o batimento cardíaco, é o nó sinoatrial. Este pode ser chamado de *pacemaker* inerente ao coração. No entanto, quando este falha, é implantado um pacemaker artificial para controlar o batimento e não-comprometer a circulação sanguínea [1]. Um *pacemaker* artificial é um dispositivo eletrónico, implantado na região torácica ou abdominal. capaz de fornecer um estímulo elétrico ao coração, de forma a este conseguir manter um ritmo estável e ininterrupto [2]. Um *pacemaker* tem também a capacidade de sensor, no qual este é capaz de detetar a atividade nas diferentes regiões do coração e condicionar a sua atividade perante estas, localizando o estímulo nas zonas de falha.

A arritmia cardíaca é a patologia relacionada com a anormalidade no ritmo cardíaco [2] e esta pode ser dividida em dois tipos diferentes: taquicardia, no caso em que o ritmo cardíaco está demasiado rápido, ou braquicardia, no caso em que o ritmo cardíaco está demasiado lento [3]. Neste trabalho vamos focar-nos na patologia B. AV *Mobitz* Tipo II (arritmia taquicardíaca do nó sinusal) que consiste numa falha da condução ao nível do feixe de *His*, em que o coração toma um comportamento "*tudo ou nada"*, isto é, após a passagem pelo nó atrioventricular, a atividade elétrica do coração cessa por completo [4]. Uma agravante deste problema é a aleatoriedade do padrão de falha, visto que, certas vezes falha 1 só impulso, contudo, o mais comum, é a falha ocorrer em blocos 2:1, ou seja, em cada 2 pulsos 1 falha, prolongando-se durante vários segundos, tornando-se assim este o foco de correção no nosso trabalho.

Figura 1 – Sinal ECG arrítmico representando B. AV *Mobitz* Tipo II.

Para aplicação nesta patologia, pretendeu-se desenvolver a simulação de um *pacemaker* do tipo VDI. Neste modo, o sinal é adquirido na aurícula direita e no ventrículo esquerdo, de forma a obter as ondas P e R, e, após um *delay*, é enviado um impulso ao ventrículo direito, caso o estímulo elétrico não tenha despoletado o complexo QRS. Este *delay* corresponde ao potencial de ativação fisiológico e encontra-se entre os 120-200 ms [5]. O estímulo elétrico fornecido ao coração, aquando da ativação do pacemaker teria a duração de 2 ms. O *pacemaker* em causa torna-se *rate responsive* pelo facto de se basear no intervalo de tempo entre a onda P e o complexo QRS, que é independente do batimento cardíaco do paciente.

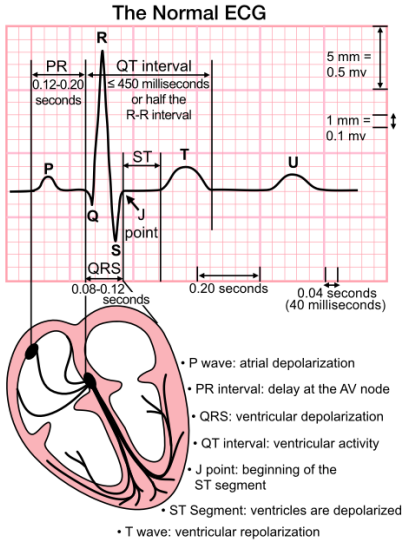


Figura 2 – Onda ECG normal, com respetivos intervalos de tempo.

Na implementação em si, iremos recorrer a um sinal ECG da plataforma *PhysioNet*, da base de dados *MIT-BIH Arrhythmia Database* – excerto do sinal 231, que já se encontra devidamente amplificado [6]. Nesta base de dados os sinais foram adquiridos com uma frequência de amostragem de 360 Hz. Este será lido em *Labview* e processado em *Multisim* de forma a remover algum *baseline wander* característico destes sinais. Por fim, em *Labview*, identificar-se-á os picos da onda P e, entre 120-200 ms após estes (intervalo PR), caso o complexo QRS não se apresente, um impulso será enviado para permitir o constante ritmo cardíaco.

II. DESENVOLVIMENTO

Primeira Abordagem

No decorrer da elaboração do projeto, o processamento descrito a seguir não foi a primeira técnica utilizada para tentar criar o *pacemaker*. A primeira abordagem consistia na utilização de um circuito de *Multisim*, baseado num circuito de um artigo de implementação de um *pacemaker* computacional [5], que seria capaz de amplificar as ondas T do sinal de ECG, atenuando as restantes.

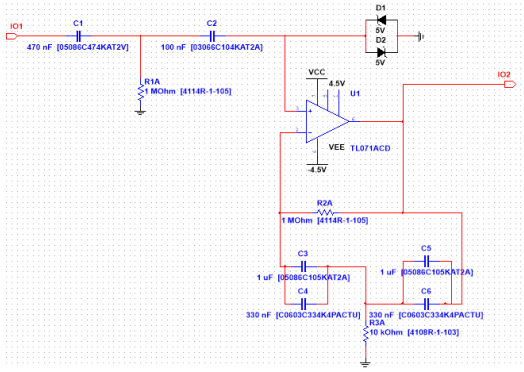


Figura 3 – Circuito de amplificação das ondas T.

Em *LabView* procedeu-se à leitura do sinal, da mesma forma que na segunda abordagem. De seguida procedeu-se à identificação dos picos R através do sinal original e dos picos P através do sinal filtrado por cosimulação.

De seguida, para identificação dos momentos em que o *pacemaker* deveria ser ativado, comparou-se cada localização da onda T com a onda R seguinte e, caso não se encontrasse dentro de um determinado valor por nós estabelecido, procedia-se à ativação do *pacemaker* gerando o impulso.

Esta abordagem foi desconsiderada por dois motivos:

1. Não era uma abordagem *rate responsive*, visto que o intervalo dentre a onda T de um ciclo cardíaco e a onda R do ciclo cardíaco seguinte não é inerente ao ECG em si, dependendo do batimento cardíaco do paciente. Desta forma, os *thresholds* escolhidos não podiam ser fixos, tal como estavam a ser considerados.
2. Os resultados obtidos, em termos de precisão, não eram satisfatórios.

Segunda Abordagem

Esta segunda abordagem pretendeu a implementação de um *pacemaker heart rate responsive* de uma maneira diferente, fazendo a comparação entre ondas P e R, de forma a evitar os problemas anteriores.

O programa de *LabView* encontra-se dividido em vários ficheiros, desde a leitura do sinal de ECG retirado do *PhysioNet* até à aplicação dos impulsos.

A. Leitura do Sinal ECG – Módulo "Read CSV"

Inicia-se pela leitura do sinal ECG em formato .csv (*comma separated values*) e são usados *arrays* para armazenar tanto os valores do sinal ECG como as instâncias de tempo a que esses valores foram captados de forma a calcular o tempo total dos dados. É também usado o bloco *Build Waveform* para se poder representar o sinal original num *Waveform Chart.*

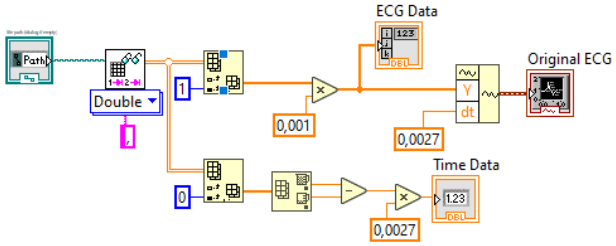


Figura 4 – Diagrama de blocos de *LabView* que efetua a leitura do sinal de ECG em formato de .csv.

B. Cosimulação e Multisim – Módulo "CoSim Filter"

De seguida, de forma a remover algum *baseline wander* e obter os picos do sinal ligeiramente mais alinhados, este é filtrado em *Multisim*. Para tal, recorreu-se ao *Control & Simulation Loop* de forma a estabelecer a comunicação entre os dois programas.

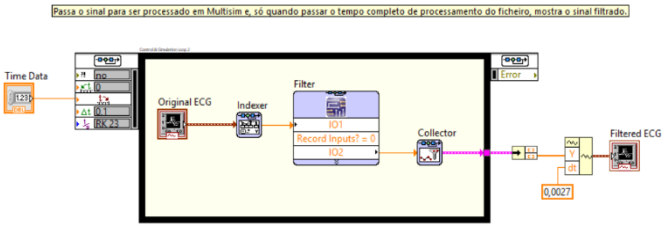


Figura 5 – Diagrama de blocos de *LabView* que envia o sinal original para o *Multisim* para ser filtrado.

O sinal original é, então, indexado pelo bloco *Indexer* e o sinal recebido, após filtragem, será armazenado pelo *Collector*. Após o tempo de funcionamento do *loop*, definido nos parâmetros deste como o tempo do sinal, este é enviado para forma do mesmo, recorrendo-se novamente à função *Build Waveform* para representar o sinal filtrado com a respetiva frequência de amostragem original.

Quanto à filtragem em *Multisim*, este é, em primeiro lugar, invertido por um amplificador operacional TL071, na configuração inversora, e com ganho unitário. De seguida, é aplicado um filtro *band pass*, com frequências de corte inferior e superior igual a 0,5Hz e 150Hz , respetivamente. Este filtro é, mais uma vez, aplicado com um amplificador operacional TL071 na configuração inversora. Desta forma, obtemos o sinal filtrado e não-invertido. O amplificador em causa foi escolhido por ser um amplificador de baixo ruído e que permite a execução da cosimulação.

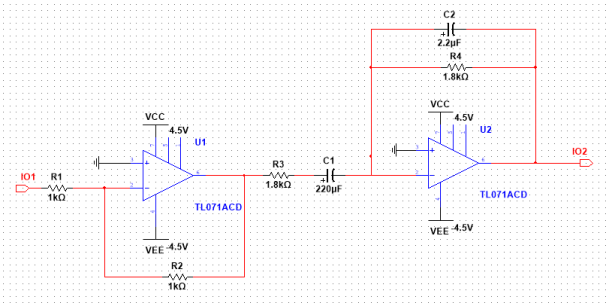


Figura 6 – Esquema *Multisim* para filtragem do sinal de ECG recebido.

C. Deteção dos Picos R – Módulo "Detect R Peaks"

A deteção de picos R é importante para calcular o batimento cardíaco a que o utilizador se encontra, visto que este pico representa a contração dos músculos ventriculares, ou seja, informa se o sangue está de facto a ser bombeado para o exterior do coração.

Sendo assim, o sinal de ECG após ser tratado é submetido ao bloco *peak detector* com um *threshold* de 0,35 V e devolve os números das amostras onde os picos se encontram. Para transformar esses números de amostra em segundos, é necessário apenas dividi-los por 360, correspondente à frequência de amostragem do sinal escolhido. De seguida os valores de tempo que indicam a localização dos picos são armazenados no *array Locations*. A combinação entre as localizações e as amplitudes dos picos P são também guardadas num *array* 2D para futuro *display* dos mesmos.

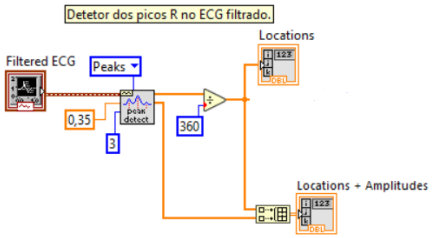
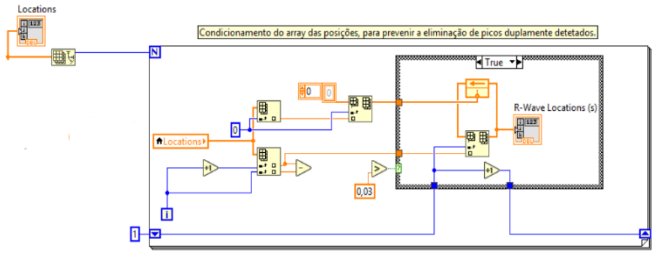


Figura 7 - Diagrama de blocos de *LabView* que faz a identificação dos picos R.

Como o sinal adquirido possui algum ruído inerente, ao fazer a deteção dos picos R, estes certas vezes surgiam rodeados por falsos picos. Para tal, tornou-se necessário processar o *array* anterior, procedendo ao armazenamento da amostra de índice i apenas se a diferença entre ela e a amostra i+1 fosse superior a 0,03 segundos. Assim, criou-se o *array R-Wave Locations*. A imagem seguinte mostra este ciclo de condicionamento tanto no caso positivo, em que há a inserção da amostra no novo *array*, quer no caso negativo, em que nem há inserção nem avanço do índice do novo *array*.

a)



b)

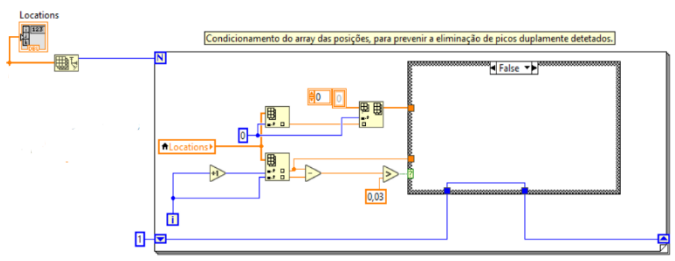


Figura 8 – Diagrama de blocos do *LabView* que faz o condicionamento do *array* das localizações (a) no caso positivo e (b) no caso negativo.

De seguida é chamado o módulo *"R Peaks Coordinates"* que vai juntar num *cluster* as localizações finais dos picos R com as suas amplitudes, sob a forma de coordenadas para permitirem o seu *display*. Este módulo será explicado posteriormente, juntamente com o módulo *"P Peaks Coordinates"*.

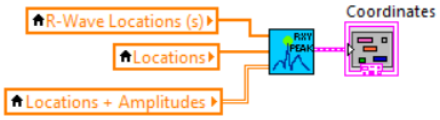


Figura 9 – Diagrama de blocos do *LabView* para chamada ao módulo *"R Peaks Coordinates"*.

D. Deteção dos Picos P – Módulo "Detect P Peaks"

Seguidamente, torna-se necessária a deteção dos picos P do mesmo sinal. Como na doença em causa verifica-se sempre a presença de pico P, independentemente da presença ou ausência do complexo QRS seguinte, a comparação entre as localizações dos picos P e R vão ser relacionados de forma a identificação do momento de geração do impulso, na secção seguinte.

Neste módulo, para o isolamento das localizações dos picos P, certos *arrays* são obrigatoriamente formados no interior de *case structures* e *for loops*, podendo somente ser acedidos por variáveis locais nos passos seguintes. O *LabView* tem como limitação a perda de sequencialidade ao usar variáveis locais, sendo por isso necessário impor esta com o uso de uma *flat sequence*.

Assim, começa-se por aplicar duas vezes o bloco *Peak Detector* com dois *thresholds* diferentes, um claramente abaixo das amplitudes das ondas P e outro claramente acima: 0,25 V e 0.075 V, respetivamente. Estas localizações são, da mesma forma que as R, passadas para segundos. A combinação entre as localizações e as amplitudes dos picos apanhados no *Lower* são também guardadas num *array* 2D para futuro processamento e *display* dos mesmos. Desta forma, com o *threshold* superior apanha-se tudo aquilo que está presente no inferior à exceção das ondas P. Fazendo a diferença entre estes dois *arrays* (exclusão dos elementos em comum), obtém-se um *array* novo que contém as localizações das ondas P, de algumas ondas T mais baixas e de algum ruído. Esta interseção é feita da seguinte maneira: no caso em que o valor de índice i em *Lower* não se encontra presente em *Upper*, este é inserido no novo *array Difference*, avançando o seu índice. No caso negativo nada é feito, tal como na secção anterior.

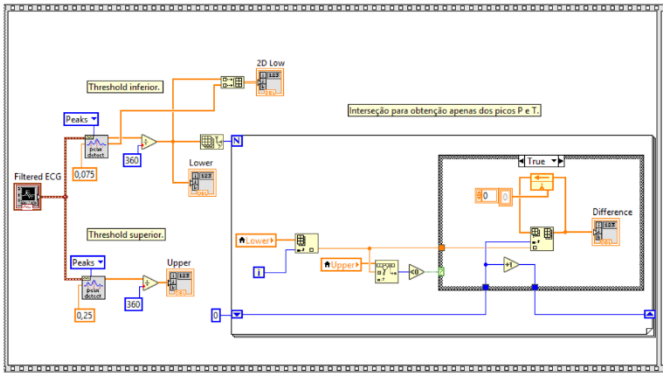


Figura 10 – Diagrama de blocos do *LabView* que aplica os dois *thresholds* e a diferença entre os dois *arrays* gerados.

Posto isto, procedeu-se de igual maneira à secção anterior no que toca à eliminação de picos duplamente detetados mas, desta vez, com um *threshold* igual a 0.25, criando o *array Clean Difference*.

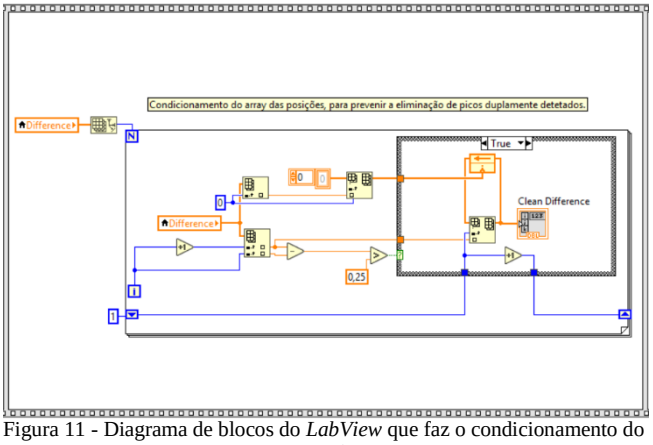


Figura 11 - Diagrama de blocos do *LabView* que faz o condicionamento do *array* das diferenças.

De seguida o objetivo é eliminar as localizações dos picos T, para que restem somente as P. O raciocínio parte do princípio que a primeira localização encontrada é sempre P e processa-se da seguinte forma: ao correr o *array* anterior, caso a diferença entre o elemento de índice i e o elemento de índice i+1 fosse inferior a 0.5 segundos (isto é, foi encontrado um pico dentro do mesmo ciclo cardíaco) este corresponde a uma onda T. Assim, no *array P-waves* é inserido o elemento i e, de seguida, realiza-se uma dupla incrementação no *array*, de forma a que o elemento i+1, correspondente ao pico T, não seja avaliado - figura 12a. No caso do intervalo entre índices ser superior a 0.5 segundos (isto é, os picos localizam-se em ciclos cardíacos diferentes), realiza-se a inserção do elemento de índice i, correspondente a uma onda P, no entanto só se faz uma incrementação, de forma a que o elemento seguinte seja avaliado, visto que este corresponderá à onda P do batimento seguinte - figura 12b.

a)

b)

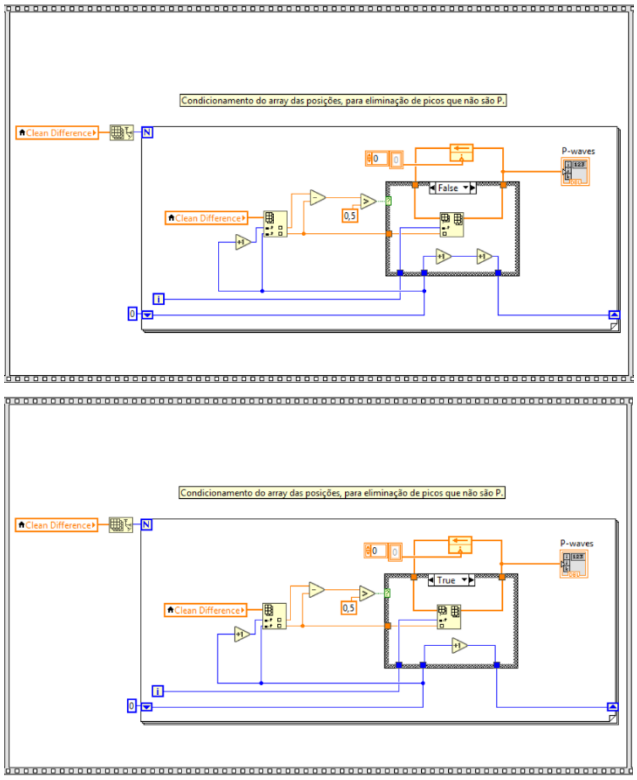


Figura 12 – Diagrama de blocos do *LabView* que mantém apenas as localizações respetivas às ondas P, caso a localização seguinte detetada seja uma onda (a) T ou (b) P.

Devido ao facto de haver a dupla incrementação, o ciclo anterior vai colocar zeros no final do *array* de saída, visto que este atinge o elemento final do *array* de entrada antes de acabar o ciclo. Assim, criaram-se outros dois ficheiros, um com a intenção de eliminar elementos repetidos do *array* e outro de eliminar o único 0 que nele fica presente.

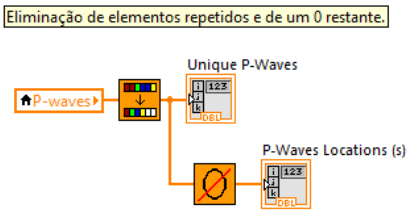


Figura 13 – Diagrama de blocos do *LabView* para interligação entre a eliminação de elementos repetidos e a eliminação do 0 restante.

No que toca à eliminação de elementos repetidos criou-se um módulo de eliminação – Módulo *"Unique Elements"* – o qual foi baseado num exemplo extraído de um fórum da NI [7]. Este consiste na ordenação do *array* de forma a que todos os pontos repetidos fiquem juntos, convertendo, de seguida, o *array* para *variant data*. Este tipo especial de *container* não vai registar elementos com o mesmo nome e valor. Ao reconverter para *array*, garantimos assim que todos os elementos são diferentes e ordenados.

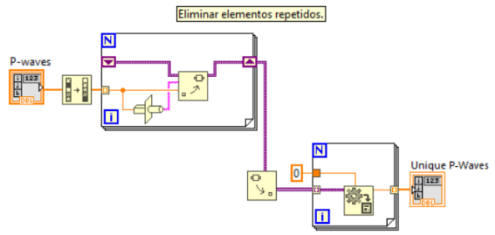


Figura 14 – Diagrama de blocos do *LabView* para a eliminação de elementos repetidos.

Para finalizar o processamento, para a eliminação do 0 que se encontra na primeira posição criou-se um módulo *"Delete Zero"*. O processamento neste módulo começa por inverter o *array* anterior e eliminar o seu último elemento. Por fim, inverte-se de novo o *array*, obtendo o *array* final das *P-Waves Locations*.

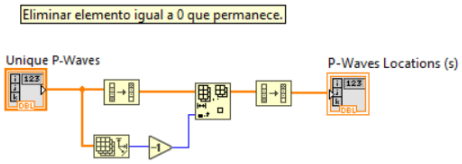


Figura 15 – Diagrama de blocos do *LabView* para eliminação do 0 na primeira posição.

Por fim, à semelhança da secção anterior é chamado o módulo *"P Peaks Coordinates"* que vai juntar num *cluster* as localizações finais dos picos P com as suas amplitudes, sob a forma de coordenadas para permitirem o seu *display*.

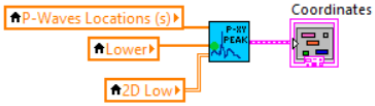


Figura 16 – Diagrama de blocos do *LabView* para chamada ao módulo *"P Peaks Coordinates"*.

E. Representação das Localizações-Amplitudes em Coordenadas – Módulos "R Peaks Coordinates" e "P Peaks Coordinates"

De forma a ser possível a identificação dos picos na visualização do gráfico final, foi então necessária a obtenção das coordenadas XY destes. Para tal, seguindo um raciocínio análogo para as ondas R e P foram criados os módulos *"Peak Coordinates"*.

O primeiro passo foi descobrir as coordenadas dos picos finais, sendo necessário pegar nas localizações inicias, vindas diretamente do *peak detector*, e descobrir os índices destas que correspondem aos pontos das localizações finais (dado que houve um processamento que intermédio que alterou a ordem). Assim, sempre que se verificava que um ponto final se encontrava no *array* inicial, o índice deste era guardado no *array Indexes*.

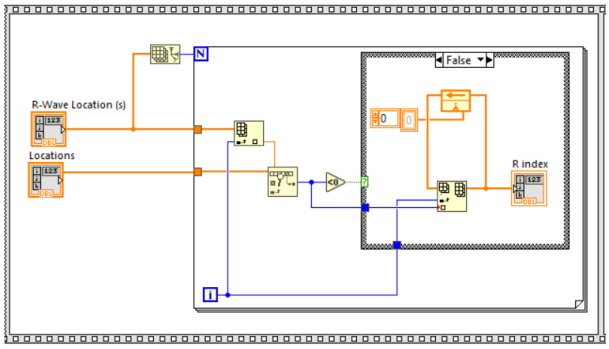


Figura 17 – Diagrama de blocos do *LabView* para identificação dos índices correspondentes aos picos R no *array* inicial.

O segundo passo, visto que as coordenadas se baseiam na localização temporal (X) e na amplitude (Y), baseou-se em pegar no *array* 2D, e como agora se sabe os índices iniciais, foi então possível selecionar deste o par de coordenadas correspondentes aos picos finais. Estes foram guardados num *cluster* sendo possível, no fim, ser visualizados como pontos sobrepostos ao sinal.

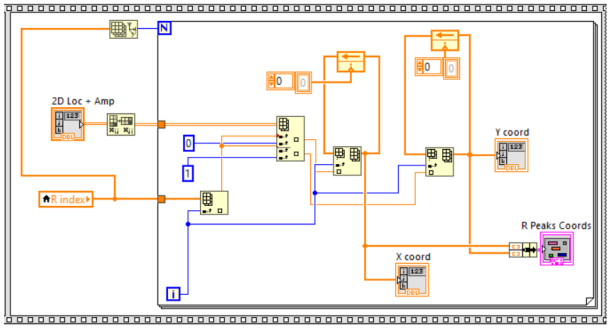


Figura 18 – Diagrama de blocos do *LabView* para obtenção do *cluster* com os pares localizações-amplitudes correspondentes aos picos R.

F. Geração dos Impulsos – Módulo "Pace Impulse"

Para a geração dos impulsos, correu-se o *array* das *P-Waves Locations* e, caso a localização do pico R seguinte fosse superior e inferior à localização da onda P mais 110 ms ou mais 210 ms, respetivamente, significava que houve a contração dos músculos ventriculares e houve bombeamento do sangue pelo coração. Neste caso, avança-se para a localização seguinte do *array* das ondas R. Caso a condição anterior fosse negativa, haveria necessidade de enviar um impulso através do *pacemaker*.

Idealmente seriam usados os valores de 120-200 ms, mas, neste caso, verificou-se que estes não eram suficientes para garantir a condição dos intervalos, sendo que utilizámos 110-210 ms para a margem ser ligeiramente superior.

O impulso foi gerado através do bloco *Impulse Pattern*. Este bloco cria um *array* do tamanho do sinal original, em amostras, cujo valor correspondente à amostra em que é necessário enviar o impulso é 1, sendo tudo o resto igual a 0. O 1 corresponde ao estado binário ativo do *pacemaker* e o 0 ao seu estado inativo. Os *arrays* dos vários impulsos são somados e passados a uma forma de onda, de forma a fazer o seu *display*.

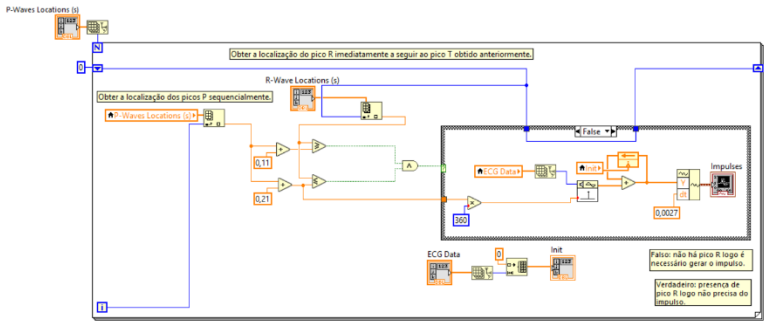


Figura 19 – Diagrama de blocos do *LabView* para geração dos impulsos do *pacemaker* quando necessário.

G. Passagem dos Sinais para Cluster – Módulo "Waveform to Cluster"

No final, devido a ser necessária a representação no gráfico de pontos soltos, para evidenciar a determinação correta dos picos P e R, decidiu-se optar por XY *Graph*, que recebe como *input clusters* de pontos. Como ao longo do processo trabalhamos com *waveforms*, foi então necessário a criação de um módulo, designado de *"Waveform to Cluster"*, para a conversão do formato. Este módulo começa por decompor a *waveform* nas suas componentes, obtendo assim diretamente as ordenadas Y. Como sabemos que para cada ordenada existe um avanço *dt*, utilizou-se um ciclo *for* com um incremento com este intervalo, criando assim todos pares de coordenadas XY, armazenados num *cluster*. Este *cluster* torna-se, assim, viável de ser adicionado/visualizado no XY *Graph*.

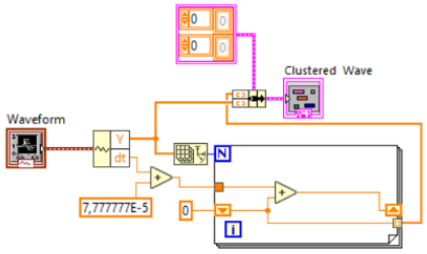


Figura 20 – Diagrama de blocos do *LabView* para passagem de uma *waveform* para *cluster*.

H. Pacemaker Final

Interligando os módulos com os diversos processamentos anteriores, obtém-se o seguinte diagrama de blocos:

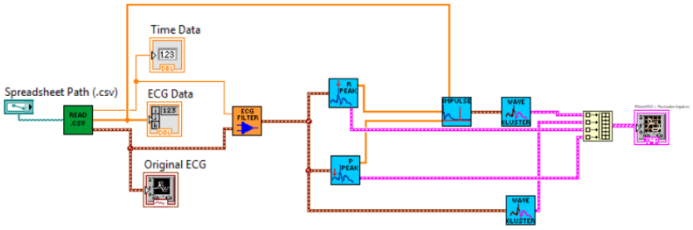


Figura 21 – Diagrama de blocos do *LabView* para implementação geral do *pacemaker* pretendido.

Este começa, então, por fazer a leitura do sinal, recorrendo à cosimulação para a filtragem analógica, passando por ambos os *peak detector* que vão servir de base para a geração dos impulsos, sobrepondo toda esta informação num gráfico final.

III. Resultados e Discussão

Primeira Abordagem

Relativamente à primeira abordagem, verificou-se que as ondas T eram devidamente amplificadas. Por observação do gráfico verificou-se que, caso existisse onda T no sinal original, esta era representada no sinal filtrado com uma amplitude superior. Caso não existisse, esta era representada na mesma, mas com uma amplitude inferior.

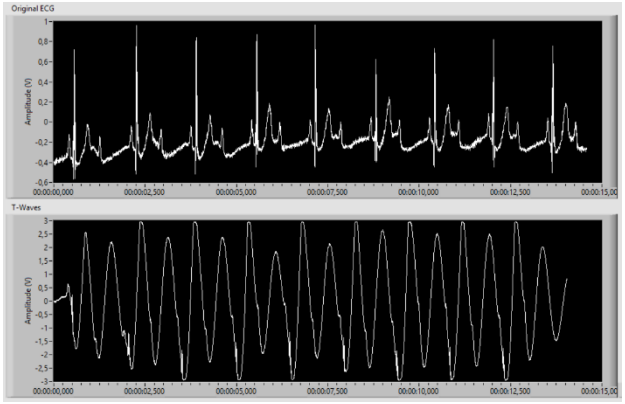


Figura 22 – Sinal original (cima) e sinal filtrado em *Multisim* para amplificação das ondas T (baixo).

Após a identificação dos picos e a geração dos impulsos, observou-se o comportamento presente na figura 18. Verifica-se, então, que apesar de alguns impulsos estarem corretamente colocados, outros encontram-se em localizações não desejadas, daí esta abordagem ter sido desconsiderada.

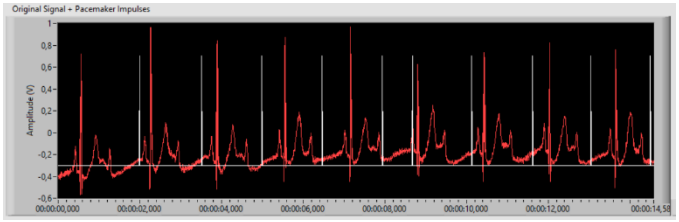


Figura 23 – Gráfico representante do sinal de ECG filtrado e dos impulsos gerados na primeira abordagem.

Segunda Abordagem

A implementação do *pacemaker* foi testada com uma porção do sinal arrítmico B. AV *Mobitz* Tipo II, com duração de 15 segundos, na qual o paciente sofre de um típico bloco 2:1; sendo este um dos casos mais cruciais em que o *pacemaker* é necessário.

Ao correr o programa uma vez, verificou-se que todos os processos são concluídos de uma forma eficiente, mostrando o resultado final com a localização dos impulsos do *pacemaker* corretamente aplicados sob o sinal de ECG. As diversas fases do processo vão ser agora exploradas.

Após a leitura do sinal e a representação da sua onda este é enviado para o *Multisim* para a remoção do *baseline wander*. Como se trata de um sinal arrítmico inconsistente, a resposta à aplicação da filtragem entre 0,5 Hz e 150 Hz não é ideal, havendo uma certa distorção no sinal, especialmente nas zonas em que o complexo QRS não aparece. Esta distorção, em parte, dificulta a nivelação das ondas P, estando estas em patamares diferentes, dependendo se há um batimento normal ou não.

Contudo, numa visão geral da totalidade do sinal, este encontra-se muito menos oscilatório e fica notavelmente alinhado. Este passo de processamento é importante para que a definição dos *thresholds*, quer para deteção do pico R e pico P, seja possível.

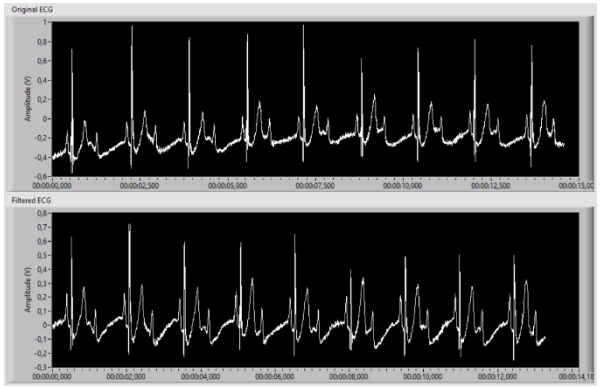


Figura 24 – Sinal original (cima) e respetivo sinal filtrado após a passagem por *Multisim* (baixo).

Depois da obtenção do sinal filtrado, foi então possível definir os limites para os quais se detetariam os picos da onda R. Apesar do nosso sinal não possuir oscilação na sua base, este possui um ligeiro ruído que afeta o isolamento dos picos, surgindo mais do que suposto no mesmo complexo. Após o processamento deste obteve-se então a localização dos picos R, posteriormente necessária para avaliar a necessidade de impulso elétrico por parte do pacemaker.

Figura 25 – Localizações obtidas para os picos R, em segundos.

No que toca à determinação dos picos da onda P, os resultados ao longo de todo estes processos encontram-se na figura 26.

O processamento por detrás deste começa por envolver uma metodologia de diferença, obtendo, em primeiro lugar, os picos mais altos (R e T) com um *threshold* superior e, em segundo lugar, os 3 tipos de picos (P, R e T) com um *threshold* mais baixo. Ao construir um *array* (*Difference*) em que somente são guardados os pontos do segundo que não aparecem no primeiro, assinalados a verde.

No entanto, nem todas as localizações nesta fase obtidas pertencem a picos P, havendo uma interferência por parte do ruído, criando picos duplicados com distâncias bastante curtas entre si. Para além disso, certos picos T não são totalmente eliminados, devido ao facto de o *threshold* que isola somente ondas R e T ter de ser definido num valor obrigatoriamente acima do maior pico P. Devido à ligeira distorção do sinal isto leva a que certos picos T sejam inferiores a esse *threshold*. O tratamento de dados seguinte resolve estes 2 pontos de forma a possuirmos um *array* com somente com as localizações dos picos P. Em primeiro lugar elimina então os picos duplamente detetados, assinalados a vermelho, e de seguida mantém apenas aqueles correspondentes a picos P, assinalados a azul.

Este *array*, contudo ainda não se encontra perfeito, dado que, por haver uma falha nas iterações dos ciclos do *LabView*, há repetição de zeros no final do *array* obtido. Assim, foi utilizado um método de remoção de duplicados, assinalados a roxo e, no fim, do zero que se encontra a mais, assinalado a laranja, acabando por ter as localizações finais das ondas P.

Neste momento, tendo as noções finais das localizações no sinal dos picos P, correspondentes ao início do batimento, e das ondas R, correspondentes à propagação ideal do batimento, então torna-se assim possível executar a avaliação do sinal.

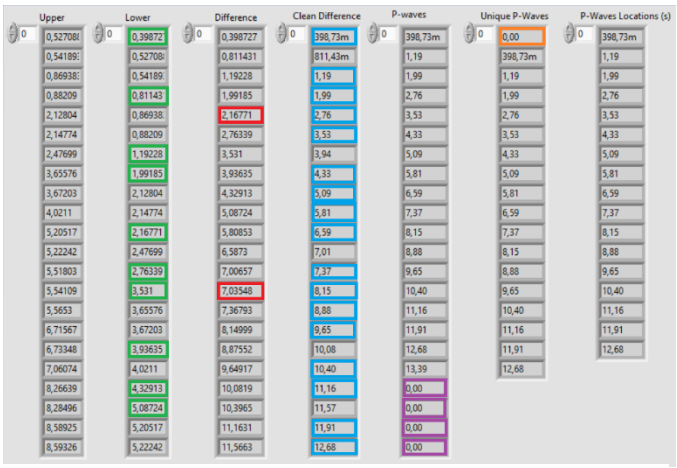


Figura 26 – Resultados das diversas fases do processamento de obtenção das localizações dos picos P, em segundos.

Como descrito na literatura e visualizável na figura 2, o intervalo PR é algo fixo, variando pouco com o aumento do batimento cardíaco. Assim, de forma a testar se o *pacemaker* deveria proceder à estimulação ventricular, verificamos se entre os 110 ms e 210 ms seguintes à localização da onda P existiu um pico R. Caso não tenha então este vai "disparar" o *pacemaker* ligando-o, aparecendo, no nosso caso, um impulso binário.

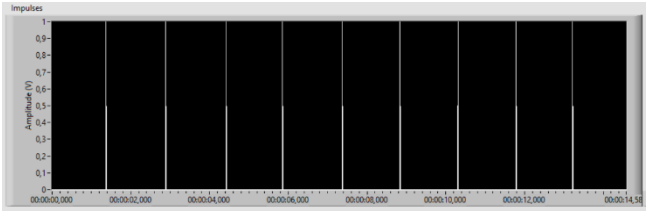


Figura 27 – Gráfico representante dos impulsos binários correspondentes à atividade do *pacemaker*.

Verificamos que em todos os casos em que falha o pico no nosso sinal o *pacemaker* é devidamente ligado na localização temporal indicada, isto é, imediatamente após a janela de verificação do pico R, de forma a substituir este, respeitando, contudo, o ciclo eletrofisiológico.

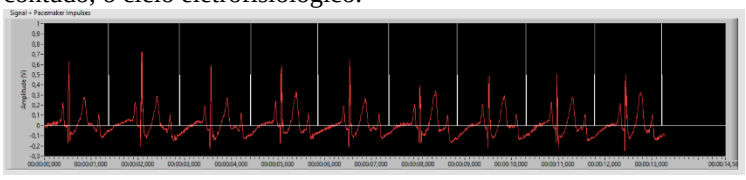


Figura 28 – Gráfico representante do sinal de ECG filtrado e dos impulsos gerados.

Assim, obteve-se a seguinte interface final, com o diretório correspondente ao sinal original, a representação do mesmo e a representação do sinal filtrado sobreposto com a atividade do *pacemaker* e os respetivos picos identificados, R a verde e P a azul:

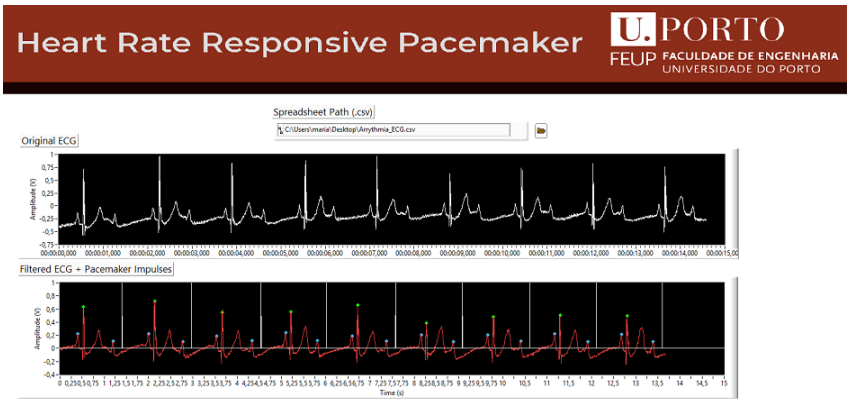


Figura 28 – Interface final da implementação do *pacemaker*.

IV. Trabalhos Futuros

Este trabalho foi feito considerando exclusivamente os 15 segundos de sinal escolhidos. De forma a aplicar o processamento a todo o comprimento do sinal e, talvez, a outros sinais com a mesma patologia, os *thresholds* usados não poderiam ser constantes específicas. Futuramente, seria de interesse arranjar uma maneira de ir adaptando os *thresholds* ao longo do sinal de forma a tornar o processamento mais geral.

V. Conclusão

Em suma, o objetivo geral do projeto foi cumprido visto que o *pacemaker* computacional projetado consegue eficientemente analisar e responder à falha do ritmo cardíaco típico da doença em questão. Este é não só capaz de sentir as diferentes correntes elétricas que atravessam o coração, isto é detetar os picos P e R; mas também de agir consoante estas, gerando impulsos elétricos quando algum dos picos R falham. Para além disso, como o ciclo eletrofisiológico do coração apresenta um intervalo PR fixo, independente da frequência cardíaca, e como a verificação do pico R é baseada neste intervalo, então concluímos assim que o nosso *pacemaker* é *heart rate responsive*, fornecendo um estímulo suficientemente rápido e eficaz independentemente da frequência cardíaca.

Referências

[1] Ananya Mandal. 2018. “Que é um pacemaker?”. News-Medical.
[2] Experiment 5-Biomedical and Signal Processing Lab, http://vlab.amrita.edu/userfiles/5/file/Experiment%205-Biomedical%20and%20Signal%20Processing%20Lab.pdf, (visitado em 1 de maio de 2020)
[3] Debra Sullivan. 2018. “What is a pacemaker?”. Healthline.
[4] AV Block: 2nd degree, Mobitz II (Hay block), https://litfl.com/av-block-2nd-degree-mobitz-ii-hay-block/ (visitado em 1 de maio de 2020)
[5] M. Penhaker, M. Heczko. 2015. “Computer Based Controlled Pacemaker Implementation”.
[6] MIT-BIH Arrhythmia Database, https://physionet.org/content/mitdb/1.0.0/?fbclid=IwAR1DJyknGBcXVvdfsngkcTozb2N-GJLmY2b-8VfJQQIvkmvxfGfl_2XCreE, (visitado em 1 de maio de 2020)
[7] Remove duplicate element & Sort the Array, http://labviewclub.blogspot.com/2014/10/remove-duplicate-element-sort-array.html, (visitado em 1 de maio de 2020)