UNIVERSIDADE DO ESTADO DO AMAZONAS

ERICK LEONARDO DE SOUSA MONTEIRO

DESENVOLVIMENTO DE UM SISTEMA GENERALIZADO DE CÓDIGO ABERTO DE PROCESSAMENTO E MANIPULAÇÃO DE SINAIS DE ELETROCARDIOGRAMA

Manaus/AM
2017

ERICK LEONARDO DE SOUSA MONTEIRO

DESENVOLVIMENTO DE UM SISTEMA GENERALIZADO DE CÓDIGO ABERTO DE PROCESSAMENTO E MANIPULAÇÃO DE SINAIS DE ELETROCARDIOGRAMA

Pesquisa desenvolvida durante a disciplina de Trabalho de Conclusão de Curso II e apresentada à banca avaliadora do Curso de Engenharia Elétrica da Escola Superior de Tecnologia da Universidade do Estado do Amazonas, como pré-requisito para a obtenção do título de Bacharel em Engenharia Elétrica.

Orientador: Dr. Jozias Parente de Oliveira

Manaus/AM
2017

Universidade do Estado do Amazonas – UEA
Escola Superior de Tecnologia - EST

Reitor:
Cleinaldo de Almeida Costa
Vice-Reitor:
Mario Augusto Bessa de Figueiredo
Diretor da Escola Superior de Tecnologia:
Roberto Higino Pereira da Silva
Coordenador do Curso de Engenharia Elétrica:
Cláudio Gonçalves

Banca Avaliadora composta por: *Data da defesa: <18/12/2017>.*
Prof. Jozias Parente de Oliveira *(Orientador)*
Prof. André Luiz Printes
Prof. Fábio de Souza Cardoso

CIP – Catalogação na Publicação

Monteiro, Erick Leonardo de Sousa

Desenvolvimento de um sistema generalizado de código aberto de processamento e manipulação de sinais de eletrocardiograma / Erick Leonardo de Sousa Monteiro; [orientado por] Jozias Parente de Oliveira – Manaus: 2017.
72 p.: il.

Trabalho de Conclusão de Curso (Graduação em Engenharia Elétrica). Universidade do Estado do Amazonas, 2017.

1. Código Aberto. 2. Eletrocardiograma. 3. Adaptável. I. Oliveira, Jozias Parente.

ERICK LEONARDO DE SOUSA MONTEIRO

DESENVOLVIMENTO DE UM SISTEMA GENERALIZADO DE CÓDIGO ABERTO DE PROCESSAMENTO E MANIPULAÇÃO DE SINAIS DE ELETROCARDIOGRAMA

Pesquisa desenvolvida durante a disciplina de Trabalho de Conclusão de Curso II e apresentada à banca avaliadora do Curso de Engenharia Elétrica da Escola Superior de Tecnologia da Universidade do Estado do Amazonas, como pré-requisito para a obtenção do título de Engenheiro Eletricista.

Nota obtida: _____ (_____)

Aprovada em _____/_____/_____.

Área de concentração: Processamento Digital de Sinais

BANCA EXAMINADORA

_____

Orientador: *Jozias Parente de Oliveira, Dr.*

_____

Avaliador: André Luiz Printes, M. Sc.

_____

Avaliador: Fábio de Souza Cardoso, Dr.

Manaus 2017

À minha mãe, por ter dedicado toda a sua vida para que eu pudesse chegar a este ponto. Agradeço por todo o apoio e o carinho dados.

AGRADECIMENTOS

Agradeço ao meu orientador, Jozias, pela sua grande contribuição com a minha formação e todo o seu empenho na orientação do meu trabalho.

Agradeço aos meus amigos do curso, especialmente ao Marcus e ao Giovanni por durante grande parte do curso terem enfrentado os desafios junto a mim.

Agradeço aos meus amigos Joaci, Luiz e Nilson por todos os ensinamentos no decorrer do curso.

Agradeço a todos os professores que fizeram parte da minha formação, especialmente aos professores Roberto Higino e André Printes que possibilitaram diversas oportunidades para o meu desenvolvimento técnico.

RESUMO

O presente trabalho tem como objetivo o desenvolvimento de um sistema de código aberto para processamento de sinais de eletrocardiograma adaptável à diferentes sistemas de aquisição e exames. O sistema consiste em uma interface gráfica executada em uma placa com microprocessador e sistema operacional embarcado (Raspberry Pi 3) capaz de realizar diferentes tipos de exames baseados em sinais de eletrocardiogramas, também podendo realizar a captura de dados de diferentes sistemas de aquisição. O sistema também foi projetado para adicionar novos exames ou sistemas de aquisição sem a necessidade de refatorar o código principal. Foram realizados testes do sistema utilizando uma placa de aquisição de biossinais (ADS1298ECG-FE) como sistema de aquisição e um simulador de sinais cardíacos (HS-15) como fonte de sinais de eletrocardiograma para realizar o exame de Eletrocardiograma de Alta Resolução. E o resultado foi que o sinal processado pelas etapas do exame possui formas de onda muito próximas ao esperado de um exame de Eletrocardiograma de Alta Resolução, validando dessa forma a integração do sistema ao exame e ao sistema de aquisição utilizados.

Palavras-chave: Código Aberto. Eletrocardiograma. Adaptável.

ABSTRACT

The present work aims to develop a open-source system for processing electrocardiogram signals that is adaptable to different acquisition systems and exams. The system consists of a graphical interface executed in a microprocessor board with embedded operational system (Raspberry Pi 3) capable of carrying different electrocardiogram based exams by doing the data acquisition with different acquisition systems. The system was also designed to add new exams or acquisition systems without the need to recode the main code. Tests were performed using a biosignal acquisition board (ADS1298ECG-FE) as acquisition system and a electrocardiogram signals simulator (HS-15) as electrocardiogram signals source to carry the high-resolution electrocardiogram exam. As a result, the signals processed by each step of the exam have waveforms close to the expected from a high-resolution electrocardiogram, thus validating the integration between the system to the exam and acquisition system used.

Keywords: Open-source. Electrocardiogram. Adaptable.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE ABREVIATURAS E SIGLAS

ABNT:	Associação Brasileira de Normas Técnicas
ADC:	Conversor Analógico-Digital
API:	Interface de Programação de Aplicação
BIBO:	Entrada Limitada -> Saída Limitada
BPM:	Batimentos por minuto
CDHR:	*Center for Devices and Radiological Health*
DTFT:	Transformada de Fourier de Tempo Discreto
ECG:	Eletrocardiograma
ECGAR:	Eletrocardiograma de Alta Resolução
EHR:	Registro Eletrônico de Saúde
FIR:	Resposta ao Impulso Finita
GPIO:	Portas de entrada e saída de propósito geral
Hz:	Hertz (Unidade de frequência)
I2C:	*Inter-Integrated Circuit*
IDE:	Ambiente de Desenvolvimento Integrado
IEC:	Comissão Internacional de Eletrotécnica
IIR:	Resposta ao Impulso Infinita
JVM:	Máquina Virtual Java
NBR:	Norma Técnica Brasileira
PDK:	Kit de Demonstração de Performance
PGA:	Amplificador de Ganho Programável
RLD:	*Right Leg Drive*
SPI:	*Serial Peripheral Interface*
SPS:	Amostras por segundo
VI:	Instrumento Virtual

SUMÁRIO

INTRODUÇÃO

O eletrocardiograma (ECG) é um exame no qual se checa se há problemas na atividade elétrica do coração do paciente, sendo capaz de ser utilizado para diagnosticar desde condições banais até outras muito graves como infartos, crescimento de cavidades, arritmias, entre outros. O eletrocardiograma representa a atividade elétrica do coração por meio de linhas traçadas em um papel quadriculado, nele os picos e as ondulações são chamados de ondas. Por ser um exame simples de ser realizado, barato e livre de riscos, pois é um exame não invasivo que pode ser utilizado tanto no cotidiano médico como na área de pesquisa clínica e por esses motivos ele é bastante difundido no meio da cardiologia (WEBMD, 2017).

O ECG é realizado utilizando um aparelho chamado eletrocardiógrafo, estes aparelhos, que já estão bastante consolidados no mercado, têm como funcionalidades essenciais a aquisição dos sinais elétricos do coração, impressão do eletrocardiograma e, caso haja suporte, exportar os dados dos sinais em um arquivo (*CENTER FOR DEVICES AND RADIOLOGICAL HEALTH,* 1998).

O acesso aos dados pelos *softwares* utilizados pelos cardiologistas é bastante restrito e incompatível entre equipamentos devido ao uso de codificações diferentes para cada equipamento eletrocardiógrafo, estes sendo diferentes conforme o hospital em que o cardiologista realiza os exames (WANN, 2006).

Porém, o custo associado aos aparelhos eletrocardiógrafos é bastante elevado e; quando os mesmos possuem a função para exportação dos dados, estes estão codificados o que faz o seu acesso geralmente ser restrito a softwares específicos os quais possuem custo de licença também elevados (JONES, 2017).

Como o *software* e os equipamentos são normalmente financiados pelo hospital onde está sendo realizado o exame, os alunos do curso de Medicina só têm contato com os aparelhos eletrocardiógrafos quando estão frequentando as clínicas e os hospitais, ou seja, a experiência laboratorial é adquirida com uma frequência baixa.

Além disso, o tratamento dos sinais de eletrocardiograma é realizado de maneiras diferentes dependendo do diagnóstico que deva ser dado, isso devido ao fato do sinal elétrico do coração estar sujeito a ações externas ao coração que, mesmo realizando o exame conforme as recomendações, exercem influência sobre o coração como a respiração, o movimento muscular e a frequência da rede elétrica; também é necessário limitar a faixa de frequências para dar maior destaque a porção do sinal

que é avaliado para determinado exame, como por exemplo em um exame convencional é recomendado pela norma IEC 60601-2-27 que se dê destaque a porção do sinal que está na faixa de frequências de 0,67 a 40 Hz.

A finalidade deste trabalho é apresentar o sistema que foi desenvolvido, o qual é um sistema de código aberto desenvolvido para concentrar a realização de exames relacionados aos sinais de eletrocardiograma, dando suporte a diferentes aparelhos eletrocardiógrafos e oferecendo as funções necessárias para processar os sinais conforme a aplicação médica escolhida, esse sistema também armazena os dados em arquivos binários sem codificação (mais conhecidos como *raw data*) para facilitar a exportação de seus dados e possui as funções de medição e manipulação visando facilitar a análise de resultados do cardiologista responsável por realizar o exame.

Para a exposição dos assuntos abordados, de forma clara e objetiva, este trabalho está divido em 4 capítulos, além das referências.

Capítulo I - Referencial Teórico: apresenta conceitos fundamentais de processamento digital de sinais, linguagens de programação, comportamento dos sinais elétricos do coração, protocolo SPI e apresenta alguns sistemas que estão disponíveis no mercado.

Capítulo II - Método proposto: neste capítulo são descritas as etapas que são necessárias para o desenvolvimento do sistema proposto.

Capítulo III - Implementação do Projeto: descreve detalhadamente os procedimentos realizados durante a implementação do projeto.

Capítulo IV – Resultados Obtidos: apresenta os resultados obtidos ao decorrer do desenvolvimento do projeto.

1 REFERENCIAL TEÓRICO

1.1 FUNDAMENTOS DE PROCESSAMENTO DIGITAL DE SINAIS

Neste tópico serão apresentados alguns dos fundamentos importantes referentes ao processamento digital de sinais e à aquisição de sinais.

1.1.1 Sistemas Lineares e Invariantes no Tempo

1.1.1.1 Invariância no Tempo

Haykin (1999, p. 50) afirma que um sistema é tido como invariante no tempo se um atraso de tempo ou avanço de tempo do sinal de entrada implica na mesma variação de tempo no sinal de saída. Isso implica que um sistema invariante no tempo responde de forma idêntica independente de qual sinal de entrada é aplicado.

A invariância no tempo de um sinal pode ser representada pela equação abaixo.

$$y[n - n_0] = H\{x[n - n_0]\} \quad (1)$$

Onde x é o sinal de entrada, y é o sinal de saída, H é o operador do sistema e n_0 é o deslocamento no tempo.

1.1.1.2 Linearidade

Haykin (1999, p. 51) afirma que um sistema é tido como linear se ele satisfaz o princípio da superposição. Isso é, a resposta de um sistema linear a uma soma ponderada de sinais de entrada é igual a soma ponderada dos sinais de saída, cada sinal de saída sendo associado a um sinal de entrada atuando no sistema independente de todos os outros sinais de entrada.

A linearidade no tempo de um sinal pode ser representada pela equação abaixo.

$$y[n] = H\left\{\sum_{i=1}^{N} a_i x_i[n]\right\} \quad (2)$$

Onde x é o sinal de entrada resultante, y é o sinal de saída, H é o operador do sistema, xi é o sinal de entrada parcial e ai seu respectivo peso.

1.1.1.3 Soma de Convolução

A soma de convolução é uma operação utilizada para que, dado um sinal de entrada e um sistema linear e invariante no tempo o qual se conhece a sua resposta ao sinal impulso, se obtenha o respectivo sinal de saída. A operação da soma de convolução e a sua representação podem ser vistas na equação abaixo. (HAYKIN, 1999)

$$y[n] = x[n] * h[n] = \sum_{k=-\infty}^{\infty} x[k]h[n-k] \tag{3}$$

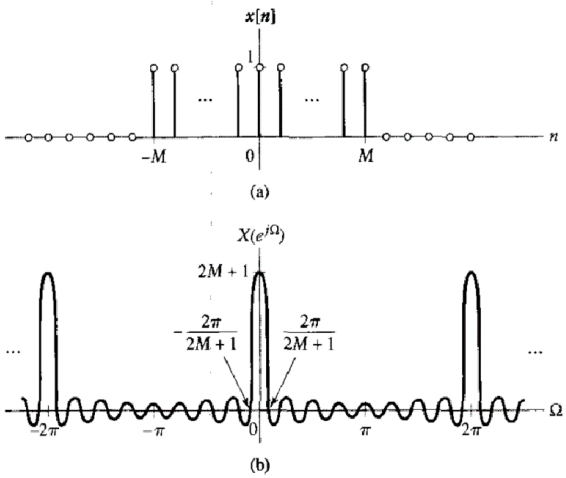
Onde x é o sinal de entrada, h é a resposta do sistema ao impulso, y é o sinal de saída e o asterisco representa o operador para soma de convolução.

1.1.2 Transformada Discreta de Tempo Discreto

Haykin (1999, p. 184) afirma que a transformada de Fourier descreve um sinal como função da frequência senoidal Ω e é tida como a representação no domínio da frequência do sinal. A transformada de Fourier pode ser representada pela equação abaixo e a sua aplicação no sinal de entrada retangular pode ser vista na Figura 1.

$$X\left(e^{j\Omega}\right) = \sum_{n=-\infty}^{\infty} x[n]e^{-j\Omega n} \tag{4}$$

Figura 1 – Domínios dos sinais. (a) Sinal de entrada (b) Sua respectiva DTFT



Fonte: (HAYKIN, 1999)

Uma propriedade bastante importante desta transformada é que a transformada da convolução de dois sinais no domínio do tempo é igual a multiplicação dos dois sinais no domínio da frequência, como a multiplicação é feita ponto a ponto, ela requer menos custo computacional que a convolução que envolve um somatório para cada instante do sinal, por isso muitas das vezes as operações são realizadas no domínio da frequência. A propriedade mencionada pode ser representada pela equação abaixo. (HAYKIN, 1999)

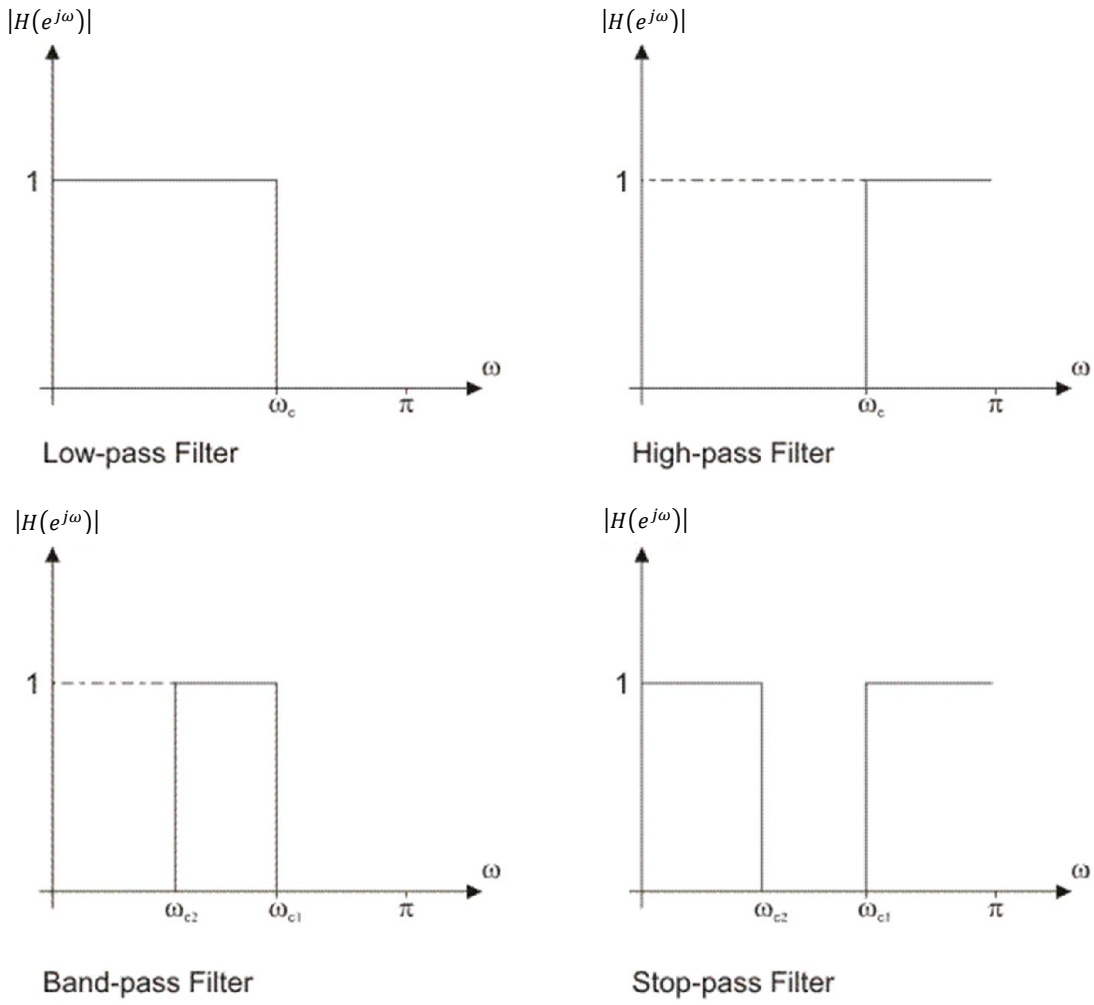
$$\mathcal{F}(x_1[n] * x_2[n]) = X_1(e^{j\Omega})X_2(e^{j\Omega}) \quad (5)$$

1.1.3 Filtros Digitais

Haykin (1994, p. 508) afirma que um filtro digital utiliza computação para implementar a filtragem que era para ser implementada no sinal de tempo contínuo [...]. O filtro digital processa a sequência de números x[n] de amostra a amostra para produzir uma nova sequência de números y[n].

Visto que o objetivo do filtro é atenuar uma faixa de frequências do sinal para dar destaque ao resto do sinal, é possível classificar as formas que os filtros podem assumir em quatro categorias: passa-baixa, passa-alta, passa-faixa e rejeita-faixa, todas as suas representações ideais podem ser vistas na Figura 2. (HAYKIN, 1999)

Figura 2 - Funções de transferência dos quatro filtros padrões



Fonte: (MIKROELETRONIKA, 2013)

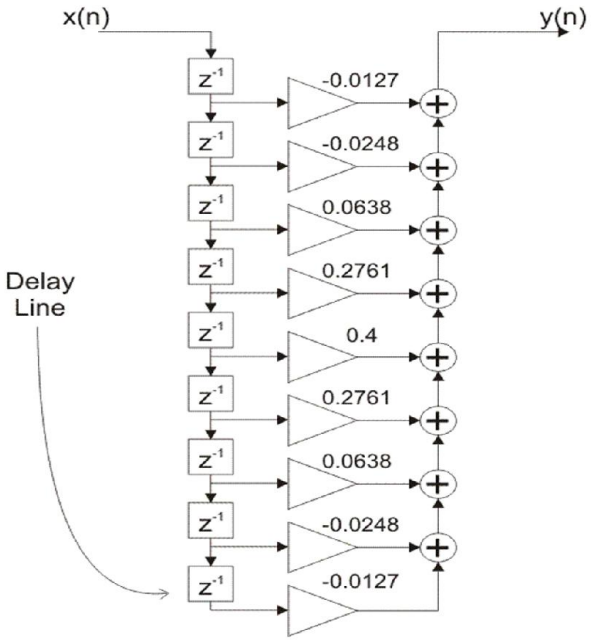
1.1.3.1 Filtros FIR

De acordo com Haykin(1999, p. 509)

> Filtros digitais com resposta ao impulso de duração finita são as operações que são governadas por equações de diferença lineares com coeficientes constantes de natureza não-recursiva. A função de transferência de um filtro digital FIR é um polinômio em z^{-1}. Consequentemente, filtros digitais FIR apresentam três propriedades importantes:
> * Eles possuem memória finita, e portanto, qualquer regime transitório é de duração limitada.
> * Eles são sempre BIBO estáveis.
> * Eles podem apresentar uma resposta de magnitude com uma resposta de fase exatamente linear (sem distorção de fase).

Na Figura 3, pode ser visto um diagrama de blocos que representa um filtro FIR, observe que há um conjunto de blocos multiplicadores com coeficientes constantes e que existe um sistema de memória bastante extenso, pois os filtros FIR precisam de vários coeficientes para se aproximar de seu respectivo filtro ideal; estes sistemas de memória também justificam o fato do regime ser limitado, pois o mesmo deixa de depender do estado inicial do filtro após estar ativo e ser aplicado em N amostras, onde N é o número de coeficientes utilizados.

Figura 3 - Diagrama de blocos de um filtro FIR



Fonte: (MIKROELETRONIKA, 2013)

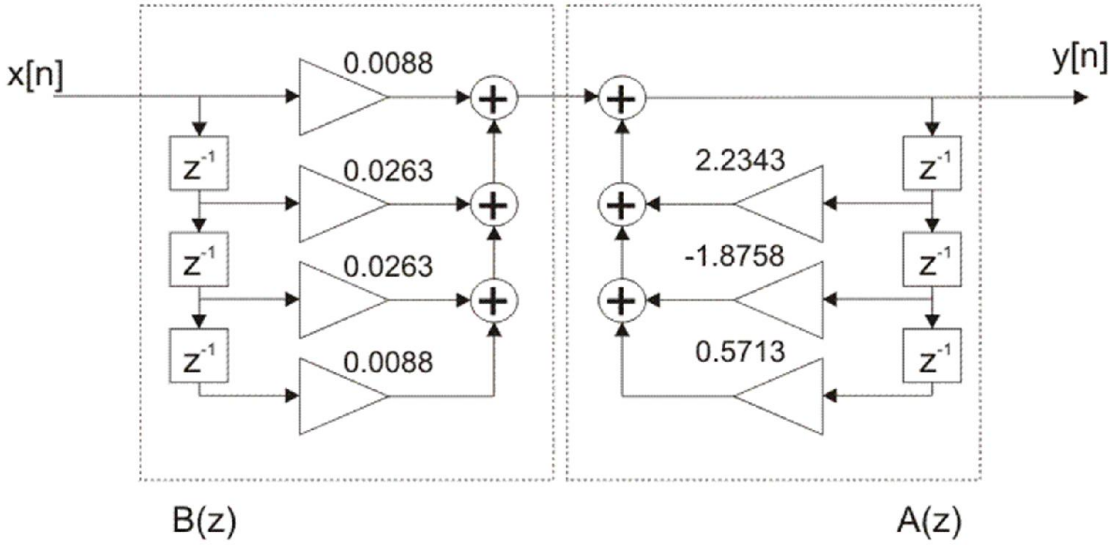
1.1.3.2 Filtros IIR

De acordo com Haykin(1999, p. 509)

> Filtros digitais com resposta ao impulso de duração infinita são as operações que são governadas por equações de diferença lineares com coeficientes constantes de natureza recursiva. A função de transferência de um filtro digital IIR é uma função racional em z^{-1}. Consequentemente, para uma resposta em frequência específica, o uso de um filtro digital IIR geralmente resulta em um filtro de menor comprimento do que o uso do filtro digital FIR correspondente. Entretanto, este melhoramento é conquistado com o custo da distorção de fase e um regime transitório que não é limitado a um intervalo de tempo finito.

Na Figura 4, pode ser visto um diagrama de blocos que representa um filtro IIR, observe que em relação ao filtro FIR, além dos blocos de memória na entrada, existem blocos de memória com retroalimentação na saída. A adição destes blocos de retroalimentação implica na influência do estado inicial do filtro sobre todas as futuras saídas e consequentemente nos problemas que o filtro apresenta. Fazendo-se a expansão de Euler da função de transferência, percebe-se que ela se assemelha a de um filtro FIR, porém com coeficientes infinitos.

Figura 4 - Diagrama de blocos de um filtro IIR



Fonte: (MIKROELETRONIKA, 2013)

1.1.4 Aquisição de Dados

Serrano (2004, p. 2) afirma que aquisição de dados é o processo através do qual os fenômenos físicos do mundo real são transformados em sinais elétricos que, por sua vez, são medidos e convertidos no formato digital para processamento, análise e armazenamento por um computador.

Um sistema de aquisição de dados básico precisa conter no mínimo os seguintes elementos:

- Transdutores
- Condicionamento de sinal
- Equipamento de medição

- Computador

Transdutores são os equipamentos que realizam a conversão da grandeza física a qual o sistema de aquisição deseja adquirir em sinais elétricos, este sinal geralmente é de baixa amplitude e muitas das vezes não-linear. (SERRANO, 2004)

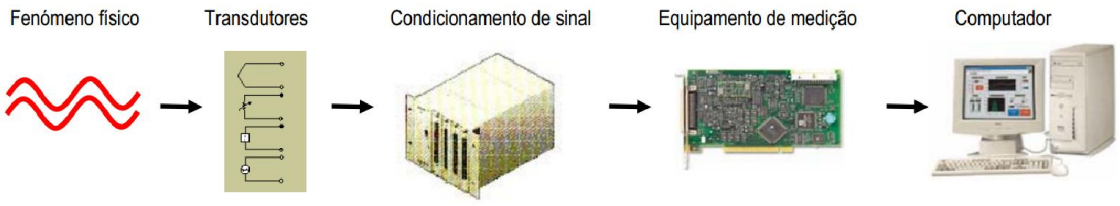
Condicionamento de sinal é a etapa que prepara o sinal para ser propriamente adquirível pelo sistema. Nesta etapa utilizam-se os circuitos apropriados para realizar as tarefas de filtragem, amplificação, linearização, isolamento e alimentação. (SERRANO, 2004)

O equipamento de medição é responsável por capturar sinais digitais e analógicos, para os sinais analógicos, ele também deve realizar o processamento e a conversão para um sinal digital utilizando conversores analógico-digital. (SERRANO, 2004)

Computador é o hardware utilizado para visualização, armazenamento e análise dos sinais de entrada e, com base nestes dados, uma aplicação executada no computador pode apresentar resultados para o operador do sistema. (Serrano, on-line, 2004)

Na Figura 5 é ilustrado um sistema básico de aquisição de dados.

Figura 5 - Diagrama funcional de um sistema de aquisição de dados



Fonte: (SERRANO, 2014)

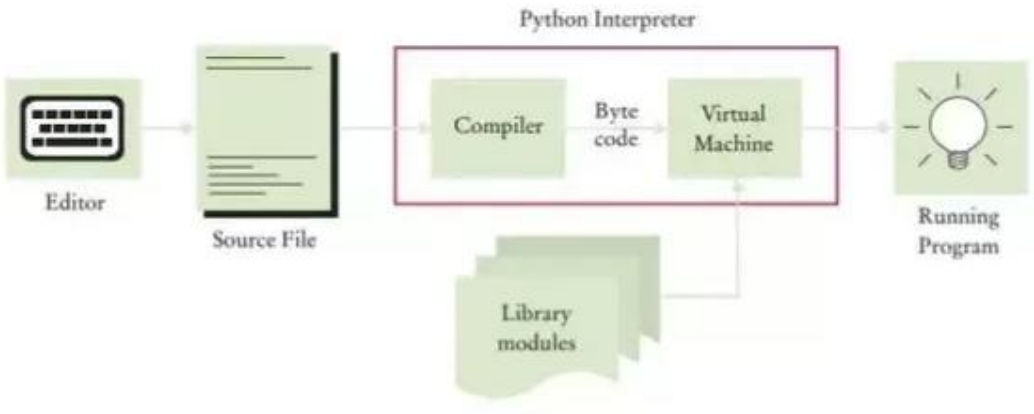
1.2 LINGUAGENS DE PROGRAMAÇÃO

1.2.1 Python

Christensson (2010) afirma que a linguagem de programação Python é uma linguagem de alto nível feita para ser fácil de ser lida e simples de implementar. Ela é de código aberto, o que significa que o seu uso é livre, inclusive para uso comercial. [...] Python é considerada uma linguagem de *script*, assim como Ruby e Perl e é frequentemente utilizada para criar aplicações *Web* e conteúdo *Web* dinâmico.

O seu funcionamento, que está ilustrado na Figura 6, é parecido com o do Java no qual o programa é executado pela Máquina Virtual Python, porém o que pode ser considerada uma grande vantagem em relação ao Java é que no processo de depuração, o programa nunca cessa por entrada de dados indevida ou falha, ele retorna uma exceção que informa o usuário qual linha de código gera o erro. Em compensação, aplicações Python costumam ser ligeiramente mais lentas que aplicações Java. (PYTHON SOFTWARE FOUNDATION, 2017)

Figura 6 - Diagrama de funcionamento do Interpretador Python



Fonte: (PYTHON SOFTWARE FOUNDATION, 2017)

1.2.2 LabVIEW

National Instruments (2014), afirma que o LabVIEW é um ambiente de desenvolvimento de software que contêm numerosos componentes, dos quais muitos são necessários para qualquer tipo de aplicação de teste, medição ou controle.

O LabVIEW utiliza a linguagem de programação G e é baseada em fluxo de dados de tal forma que o mesmo fica mais eficiente quando o fluxo de dados se aproxima mais da programação sequencial que é normalmente utilizada em outras linguagens. Os programas em LabVIEW são chamados de Instrumentos Virtuais (VI) e eles são divididos em *Front Panel* e *Block Diagram*, no *Front Panel* é possível editar os controles e indicadores da interface gráfica que será apresentada ao usuário e o *Block Diagram* é onde deve ser feito o código. (*NATIONAL INSTRUMENTS*, 2014)

As grandes vantagens do LabVIEW são a fácil integração com o hardware, a grande variedade de funções que normalmente requerem um certo tempo de implementação e a interface gráfica já estar diretamente ligada ao código, diferente de outras linguagens onde o acesso a propriedades e eventos da interface gráfica são mais restritos.

Na Figura 7 é possível ver a IDE do LabVIEW com destaque no *Front Panel*.

Figura 7 – IDE do LabVIEW

Fonte: (TEXAS INSTRUMENTS, 2013)

1.3 SINAIS ELÉTRICOS DO CORAÇÃO

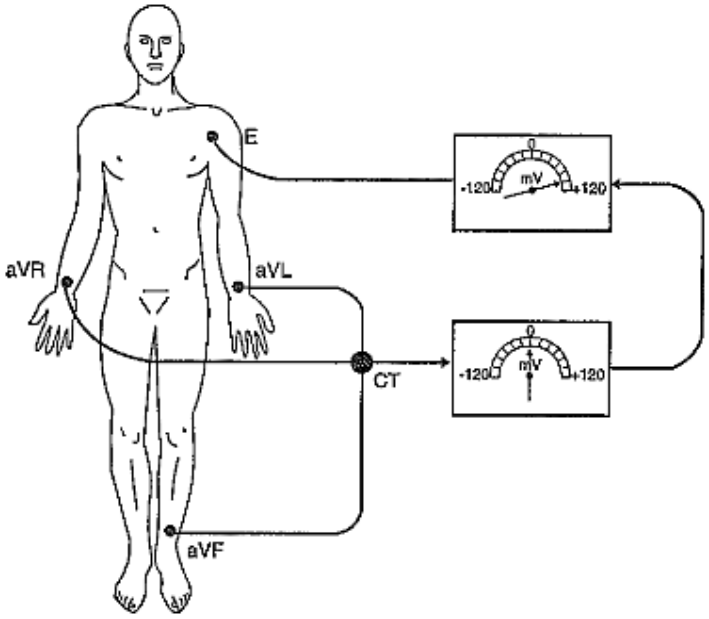
Na superfície do corpo existem diferenças de potencial consequentes aos fenômenos elétricos gerados durante a excitação cardíaca. Estas diferenças podem ser medidas e registradas. Os pontos do corpo a serem explorados são ligados ao aparelho de registro por meio de fios condutores (eletrodos). Dessa forma, obtém-se as chamadas derivações que podem ser definidas de acordo com a posição dos eletrodos. (AZEVEDO, 1999)

A ideia básica é observar o coração em diferentes ângulos, ou seja, cada derivação, representada por um par de eletrodos (um positivo e um negativo), registra uma vista diferente da mesma atividade cardíaca. As derivações podem ser definidas de acordo com a posição dos eletrodos (chamados eletrodos exploradores) no plano frontal (formando as derivações periféricas – bipolares ou unipolares) e no plano horizontal (formando as derivações precordiais, unipolares). (AZEVEDO, 1999)

As derivações que são os sinais capturados para se realizar a análise do coração podem ser divididas em:

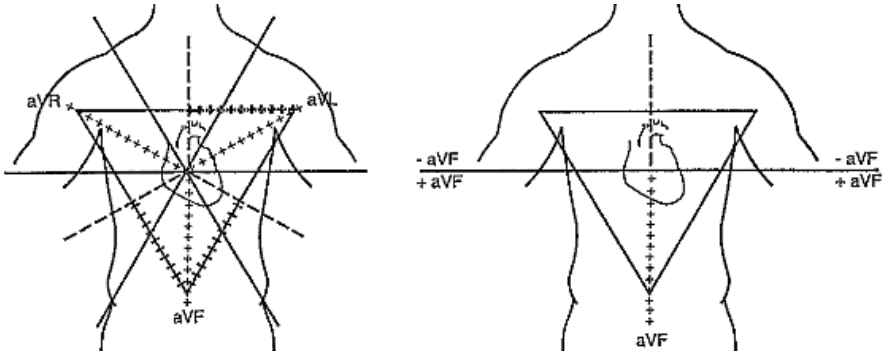
a) **Derivações unipolares das extremidades (de Wilson/Goldberger).** Para se conhecer a magnitude do potencial em cada um dos pontos de captação das derivações bipolares, isoladamente, necessita-se de um potencial elétrico de referência de valor zero. Wilson, que foi o responsável pela criação do método de obtenção dessas derivações, definiu por meio de seus estudos o posicionamento dos eletrodos cujos sinais capturados conseguem representar o hemicampo das derivações unipolares. As derivações obtidas por ele são formadas por três letras, sendo as duas primeiras aV que indicam que o potencial obtido é muito pequeno e a terceira letra que indica o posicionamento do eletrodo podendo ela ser R, L ou F; onde R significa *right* que representa o lado direito, L significa *left* que representa o lado esquerdo e F significa *foot* que representa o pé na língua inglesa. Na Figura 8 e Figura 9 é possível ver o posicionamento destes eletrodos assim como o plano que é representado pelas derivações unipolares. (AZEVEDO, 1999)

Figura 8 – Posicionamento dos eletrodos das derivações unipolares de Wilson



Fonte: (AZEVEDO, 1999)

Figura 9 – Delimitação dos hemicampos das derivações unipolares de Wilson

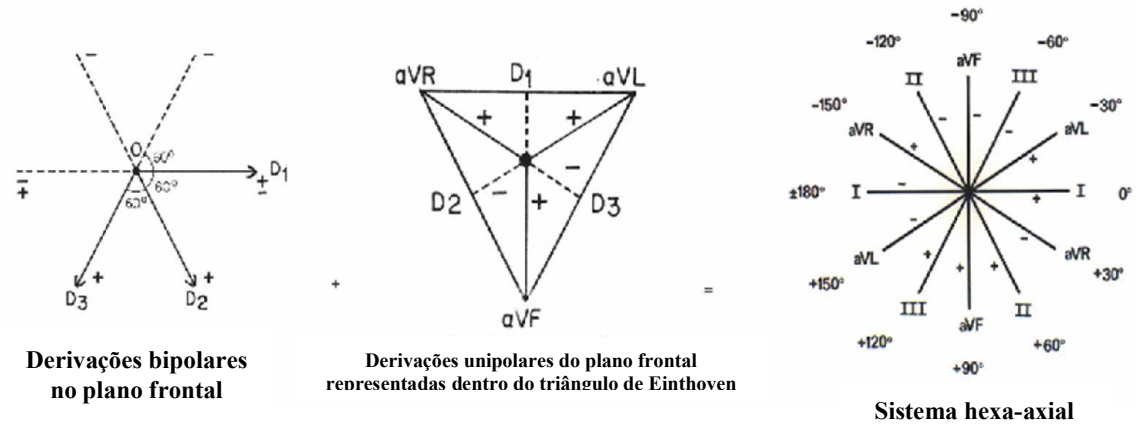


Fonte: (AZEVEDO, 1999)

b) **Derivações no Plano Frontal (Derivações de Membros ou Periféricas).** Medem a diferença de potencial entre os membros (bipolares) ou entre certas partes do corpo e o coração (unipolares). Coloca-se um eletrodo em cada braço (direito/esquerdo) e um na perna esquerda, formando um triângulo (conhecido como **triângulo de Einthoven**). Na perda direita, coloca-se o fio terra, para estabilizar o traçado. Deslocam-se as três linhas de referência, cruzando com precisão o tórax (coração) e obtém-se uma intersecção, formando as derivações bipolares DI, DII e DIII. Em seguida, acrescentam-se outras três linhas de referência nesta intersecção, com

ângulos de 30º entre si e obtêm-se as derivações unipolares dos membros: aVR (direita), aVL (esquerda) e aVF (pé), como pode ser visto na Figura 10. (AZEVEDO, 1999)

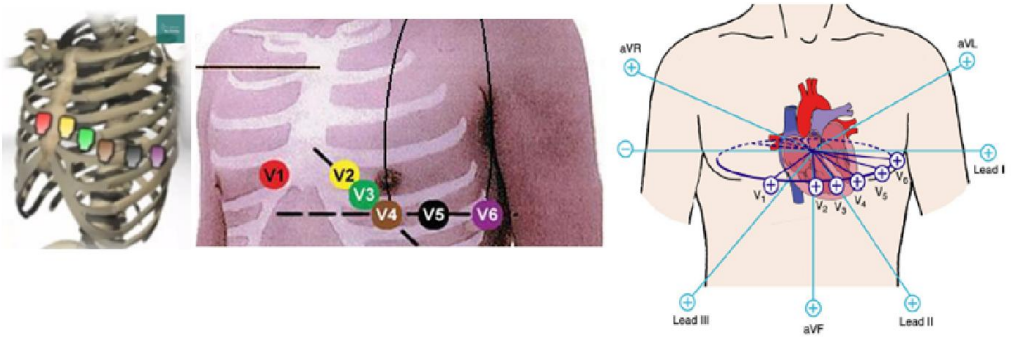
Figura 10 – Derivações no plano frontal – Triângulo de Einthoven



Fonte: (AZEVEDO, 1999)

c) **Derivações no plano horizontal (Derivações precordiais).** Têm-se, com elas, uma visão como em um corte transversal do coração. São as derivações V1, V2, V3, V4, V5 e V6. Medem a diferença de potencial entre o tórax e o centro elétrico do coração (nódulo AV), e vão desde V1 (4º espaço intercostal, na linha paraesternal direita) a V6 (5º espaço intercostal, na linha axilar média esquerda). Em todas essas derivações, considera-se positivo o eletrodo explorador colocado nas seis posições diferentes sobre o tórax, sendo o polo negativo situado no dorso do indivíduo, por meio da projeção das derivações a partir do nódulo AV, como pode ser visto na Figura 11. (AZEVEDO, 1999)

Figura 11 – Derivações no plano horizontal



Fonte: (AZEVEDO, 1999)

1.3.1 Eletrocardiograma de Alta Resolução

O ECGAR é um método não invasivo para análise de sinais eletrocardiográficos que se diferencia do ECG convencional em virtude de incrementos nas escalas de tempo e amplitude. Com isto, permite a detecção dos potenciais tardios da ativação ventricular (PTAVs), identificando indivíduos em risco de desenvolver taquicardia ventricular potencialmente fatal (GOMIS, 1997). Os PTAVs são sinais elétricos de baixa amplitude e alta frequência originados em regiões lesadas do miocárdio ventricular, nas quais a condução dos estímulos elétricos se processa de forma lenta e fragmentada (BERBARI, 2000), razão pela qual não são capturados pelo ECG.

1.3.2 Transformada de Dower

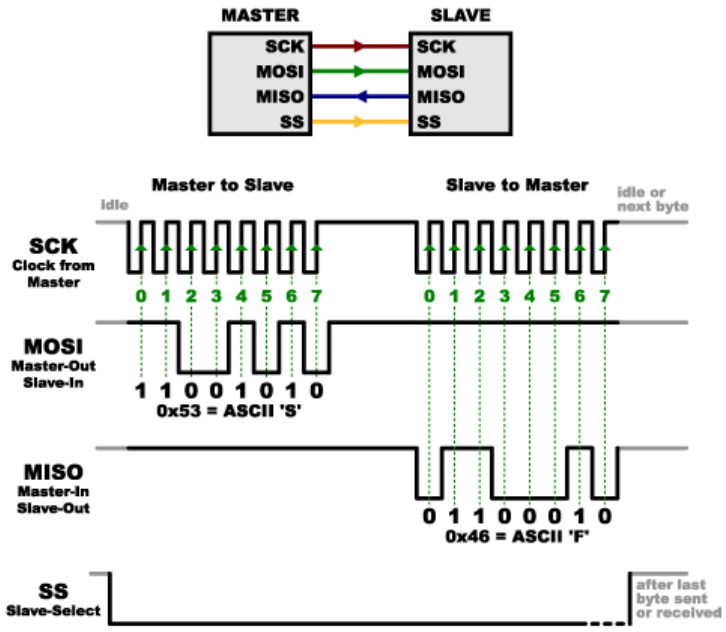
Na década de 60 (DOWER, 1968) propôs um método de simular as doze derivações do eletrocardiograma convencional a partir dos três sinais de eletrocardiograma ortogonais XYZ cuja acurácia foi provada alta o suficiente como um método de transformação do complexo QRS, este método é chamado de Transformada de Dower. Com base na transformada proposta, também foi possível derivar a Transformada Inversa de Dower que é utilizada para obter os três sinais ortogonais XYZ a partir de oito sinais independentes dentre as doze derivações, estes sendo DI, DII, V1, V2, V3, V4, V5 e V6 (Os outros sinais são combinações lineares desses). A equação matricial que caracteriza a Transformada Inversa de Dower pode ser vista na equação 6.

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} 0.156 & 0.009 & -0.172 & -0.073 & 0.122 & 0.231 & 0.239 & 0.194 \\ -0.227 & 0.886 & 0.057 & -0.019 & -0.106 & 0.022 & 0.040 & 0.048 \\ 0.021 & 0.102 & -0.228 & -0.310 & -0.245 & -0.063 & 0.054 & 0.108 \end{bmatrix} \begin{bmatrix} DI \\ DII \\ V1 \\ V2 \\ V3 \\ V4 \\ V5 \\ V6 \end{bmatrix} \quad (6)$$

1.4 PROTOCOLO SPI

O protocolo SPI é um protocolo de transmissão de dados serial e síncrono (SPARKFUN, 2017), o que significa que os dados são enviados todos por um canal a uma taxa de transmissão e essa taxa de transmissão é controlada por meio de um pino de *clock* (SCK ou SCLK) compartilhado entre os dois sistemas da comunicação, quando há um pulso negativo ou positivo no *clock*, o sistema que está recebendo os dados irá verificar o dado que está passando neste instante no pino MISO e atribuirá este dado a um registrador, este mesmo sistema pode enviar dados a partir do pino MOSI. A Figura 12 exemplifica um processo de comunicação onde o primeiro sistema (mestre) envia um byte de valor 0x53 e recebe como resposta do segundo sistema (escravo) um byte de valor 0x46.

Figura 12 - Exemplo de Comunicação SPI



Fonte: (SPARKFUN, 2017)

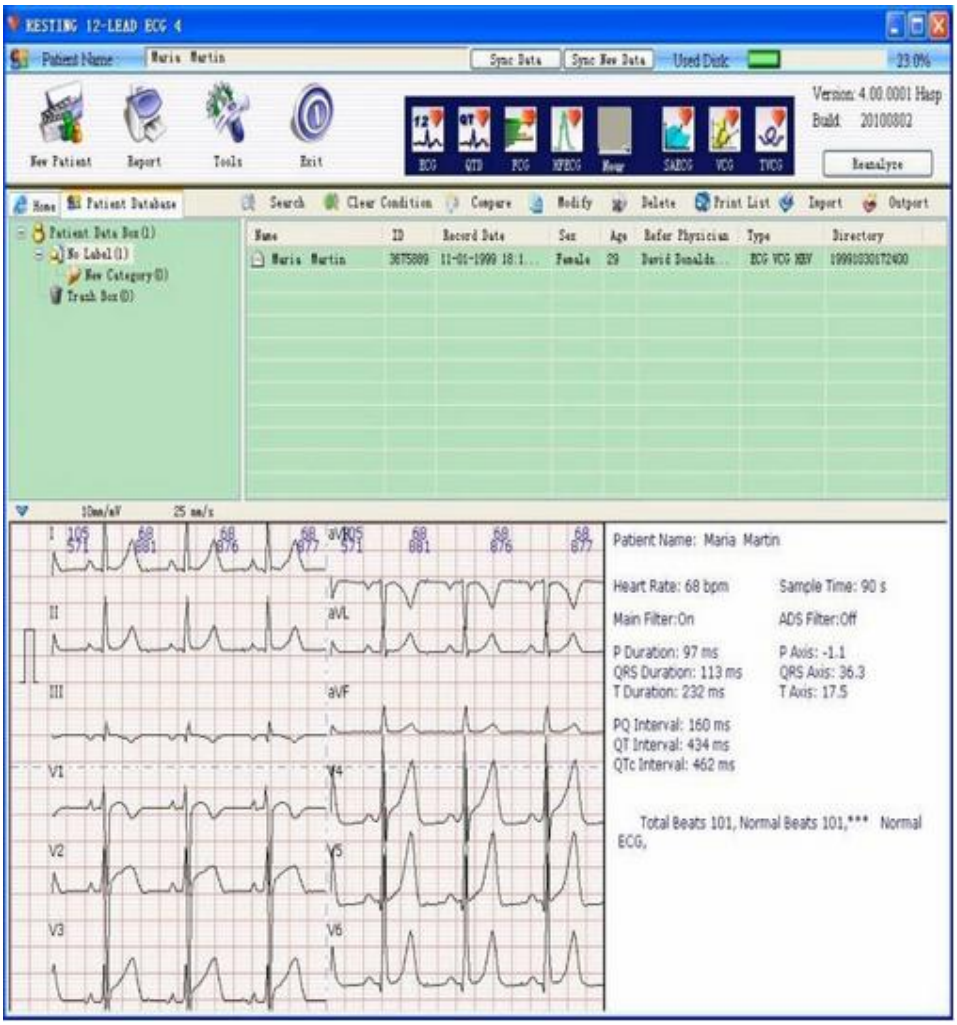
1.5 SISTEMAS EXISTENTES NO MERCADO

Neste tópico serão apresentados alguns dos sistemas disponíveis no mercado, apresentando algumas de suas vantagens e desvantagens.

1.5.1 CardioScan

O software da empresa DM Software faz o gerenciamento de *holters* e ECG de repouso e pode ser instalado em qualquer computador que cumpra os requisitos mínimos, na Figura 13 é ilustrado um exame de eletrocardiograma apresentado por este sistema (DM SOFTWARE, 2017).

Figura 13 - DM Software Cardioscan



Fonte: (DM SOFTWARE, 2017)

A Quadro 1 apresenta algumas das vantagens e desvantagens apresentadas pelo software.

Quadro 1 - Vantagens e Desvantages do CardioScan

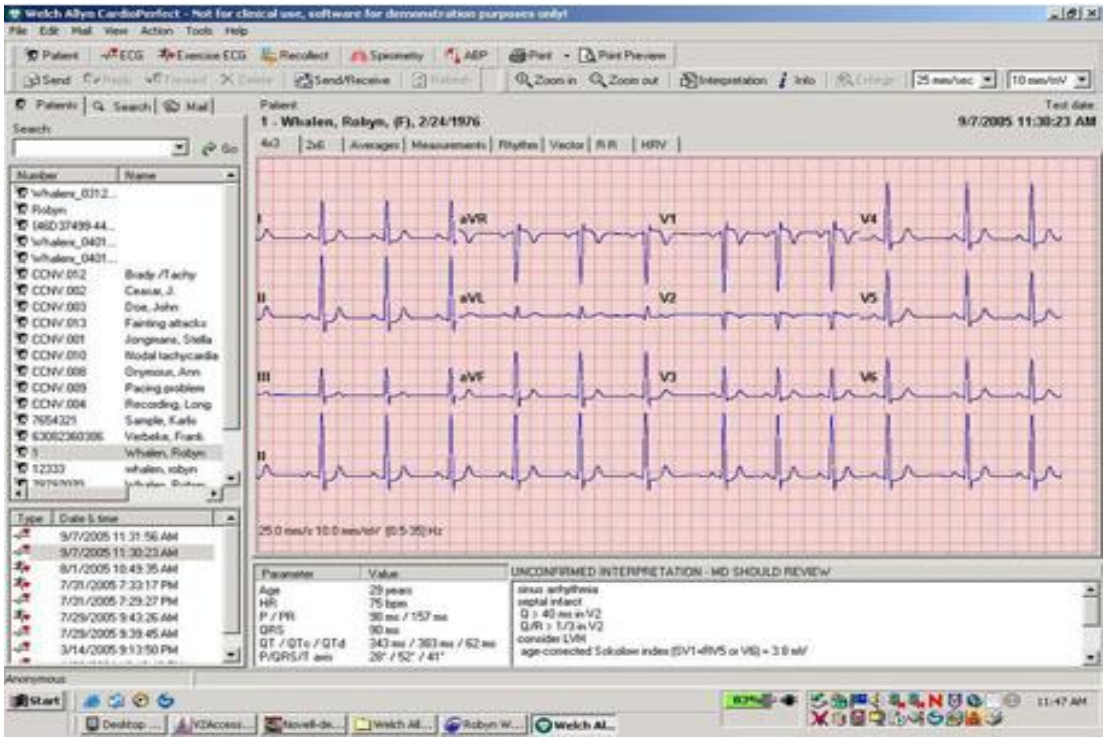
Vantagens	Desvantagens
Possui ferramentas de detecção de vários tipos de arritmia	Funciona apenas com Holters e Eletrocardiógrafos de Repouso
Possui histórico do paciente	Possui suporte apenas aos aparelhos da DM Software
Possui análise de potenciais tardios	O banco de dados só é acessível via software
Preço em torno de US$1000,00	

Fonte: (DM SOFTWARE, 2017)

1.5.2 CardioPerfect WorkStation

O *CardioPerfect Workstation* é o software desenvolvido pela empresa Welch Allyn que realiza o gerenciamento dos dados cardiopulmonares e pode ser instalado em qualquer computador que cumpra os requisitos mínimos. Na Figura 14, é ilustrado um exame de eletrocardiograma apresentado por este sistema (WELCH ALLYN, 2017).

Figura 14 - Welch Allyn CardioPerfect Workstation



Fonte: (WELCH ALLYN, 2017)

O Quadro 2 apresenta algumas das vantagens e desvantagens apresentadas pelo software.

Quadro 2 - Vantagens e Desvantagens do CardioPerfect Workstation

Vantagens	**Desvantagens**
Possui suporte a diferentes aparelhos eletrocardiógrafos da Welch Allyn	A licença do software não é parte integrante dos aparelhos eletrocardiógrafos
Possui suporte a banco de dados SQL Server	Preço em torno de US$5000,00
Possui suporte a outros aparelhos como medidor de pressão sanguínea e espirômetro	Não possui análises de potenciais tardios
Possui integração com o CardioPerfect Webstation que pode visualizar os dados que já foram gravados	
Possui histórico do paciente	

Fonte: (WELCH ALLYN, 2017)

2 MÉTODO PROPOSTO

O presente projeto foi desenvolvido para dar continuidade ao trabalho iniciado no Projeto de Iniciação Científica realizado no ano de 2016, nele verificou-se a necessidade dos cardiologistas em possuir uma interface de baixo custo que pudesse realizar os exames de eletrocardiograma de alta resolução.

Após uma pesquisa detalhada de como realizar o processo e quais equipamentos seriam necessários, foi decidido que o sistema utilizaria a placa de aquisição de sinais ADS1298ECGFE-PDF por se tratar de uma placa destinada à aquisição de biossinais e ser possuir as certificações internacionais IEC-60601-2-27 e IEC-60601-2-51, o sistema inicialmente foi projetado para ser utilizado em um computador pessoal utilizando a linguagem LabVIEW.

Porém, no decorrer do projeto, verificou-se que apesar de o código implementado em LabVIEW poder ser um código aberto, a acessibilidade a ele seria reduzida visto que ele é um *software* pago, então decidiu-se refatorar o código em Python e utilizar a placa Raspberry Pi 3 para executar o sistema, com o propósito de reduzir o seu custo.

Durante o desenvolvimento do sistema, foram realizadas quatro etapas que juntas compõem este trabalho.

A primeira etapa consiste em desenvolver scripts em linguagem Python responsáveis por fazer o processamento do sinal de eletrocardiograma aplicando filtros IIR.

A segunda etapa consiste em desenvolver um sistema em linguagem LabVIEW que seja capaz de apresentar e modificar os dados resultantes do processamento dos dados capturados pelo software de aquisição desenvolvido pela Texas Instruments assim como realizar o armazenamento e resultado final do exame.

A terceira etapa consiste em desenvolver um sistema de aquisição de dados em linguagem Python utilizando a placa ADS1X98ECG FE que é responsável pela aquisição de dados em conjunto a placa Raspberry Pi 3 utilizando o protocolo de comunicação SPI.

A quarta etapa consiste no desenvolvimento de uma interface totalmente escrita em Python que utilize as funções desenvolvidas na primeira e na terceira etapa para unificar todo o software em apenas uma linguagem open-source e que ofereça suporte facilitado a introdução de novos equipamentos e exames utilizando o

paradigma de programação orientada a fluxo.

2.1 ESPECIFICAÇÃO DOS DISPOSITIVOS E SOFTWARES UTILIZADOS NO PROJETO

Este tópico detalha os dispositivos utilizados para realizar o processamento dos sinais de eletrocardiograma. Também estão citados os softwares utilizados para a processamento dos sinais e criação das interfaces.

2.1.1 Raspberry Pi 3

Esta plataforma (pode ser vista na Figura 15) é responsável pelo processamento dos sinais e apresentação da interface gráfica tendo o funcionamento semelhante ao de um computador rodando o sistema operacional Linux Ubuntu Mate, porém com as vantagens de portabilidade, baixo custo, possuir nativamente as interfaces utilizadas por microcontroladores (GPIO, SPI, I2C, entre outros) e por poder rodar aplicações desenvolvidas em um computador visto que esta placa está rodando o sistema operacional Ubuntu Mate.

Figura 15 - Raspberry Pi 3 Model B

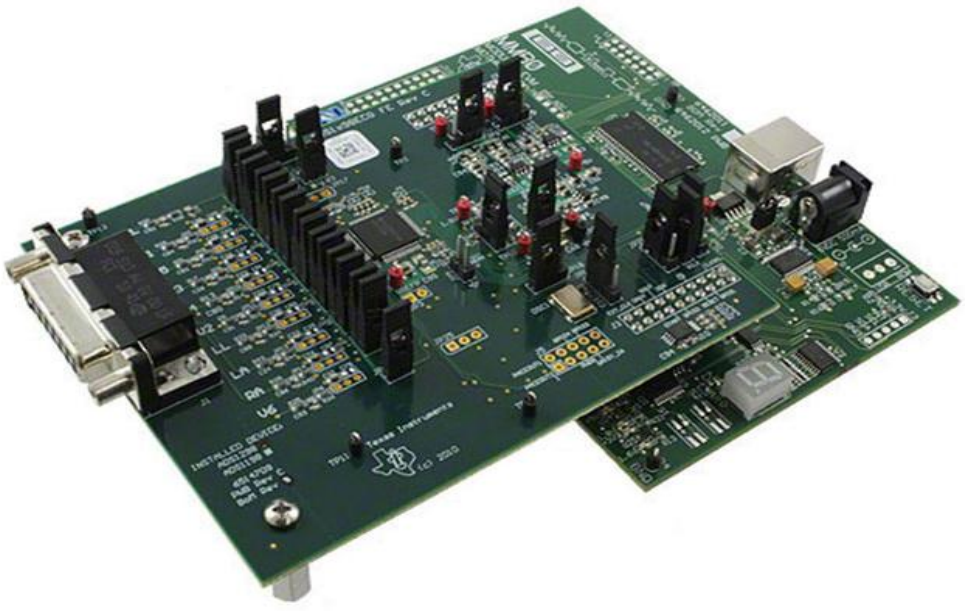
Fonte: (RASPBERRY PI FOUNDATION, 2017)

2.1.2 Placa de Aquisição de Biossinais ADS1298

A placa de aquisição que foi escolhida para se utilizar no sistema de aquisição foi a ADS1298ECGFE-PDK da *Texas Instruments*, que pode ser vista na Figura 16. Esta placa foi escolhida visto que a mesma já realiza o condicionamento e a medição, assim como não possui a necessidade de utilizar pré-amplificadores pois possui uma resolução muito alta, também o conector do tipo DB15 para os cabos de eletrocardiógrafo. As características principais para escolha da placa foram:

- Resolução de 24-bits
- Taxa de transferência de dados de 250 SPS a 32kSPS
- Ganho programável
- Possui filtros analógicos para rede elétrica (60 Hz) ativáveis
- Programável via comunicação SPI
- Possui um software responsável pela aquisição de dados desenvolvido em LabVIEW
- Composta por duas placas, uma que realiza a captura dos sinais chamada de ADS1X98ECG FE e a outra que realiza o condicionamento do sinal chamada de MMB0

Figura 16 - Placa de Aquisição de Biossinais ADS1298



Fonte: (TEXAS INSTRUMENTS, 2017)

2.1.3 Cabos e eletrodos do paciente

Os cabos escolhidos para a realização do exame foram os cabos da Dixtal de 10 vias com conectores jacaré e conector DB15 pois estes já conectariam ao ADS1298 sem nenhuma intervenção. Estes cabos possuem uma blindagem eletrostática e seguem as normas da ANVISA. Os eletrodos utilizados foram eletrodos descartáveis do tipo ventosa da 3M. A Figura 17 apresenta o cabo do paciente e o eletrodo utilizado para a aquisição dos sinais.

Figura 17 - Cabo e eletrodos do paciente

Fonte: (MEDICALMED, 2017)

2.1.4 Simulador de sinais cardíacos HS-15

Para a simulação dos sinais cardíacos utilizados na aquisição desse sistema, foi utilizado o simulador de sinais cardíacos HS-15, desenvolvido pela empresa Handy Sim, que permite gerar pulsos de eletrocardiograma nas frequências de 30, 60, 80, 120, 180 ou 240 bpm, este simulador pode ser visto na Figura 18.

Figura 18 - Simulador de Sinais Cardíacos HS-15

Fonte: (R&D MEDIQ, 2017)

2.1.5 Software de Aquisição da Placa ADS1298ECGFE-PDK

O *software* da placa ADS1298ECGFE-PDK foi desenvolvido pela *Texas Instruments* em linguagem LabVIEW e ele possui as funções necessárias para aquisição de dados de ECG. Este *software* é de código aberto, logo foi possível aplicar as modificações necessárias para integrá-lo ao sistema desenvolvido em LabVIEW. A tela principal deste *software* pode ser vista na Figura 19.

Figura 19 - Tela de aquisição - Software da placa ADS1298

Fonte: (TEXAS INSTRUMENTS, 2017)

2.1.6 Python

Para o desenvolvimento das funções de processamento de sinais foi utilizada a linguagem Python 2.7 em conjunto as bibliotecas NumPy, SciPy e Matplotlib que são bibliotecas destinadas a aplicação de técnicas matemáticas como transformada de Fourier, filtros e plotagem de gráficos. Já para o desenvolvimento das funções para a aquisição de dados foi utilizada a biblioteca periphery que é utilizada para habilitar o uso das interfaces de comunicação do Raspberry Pi 3, neste caso, a função de SPI.

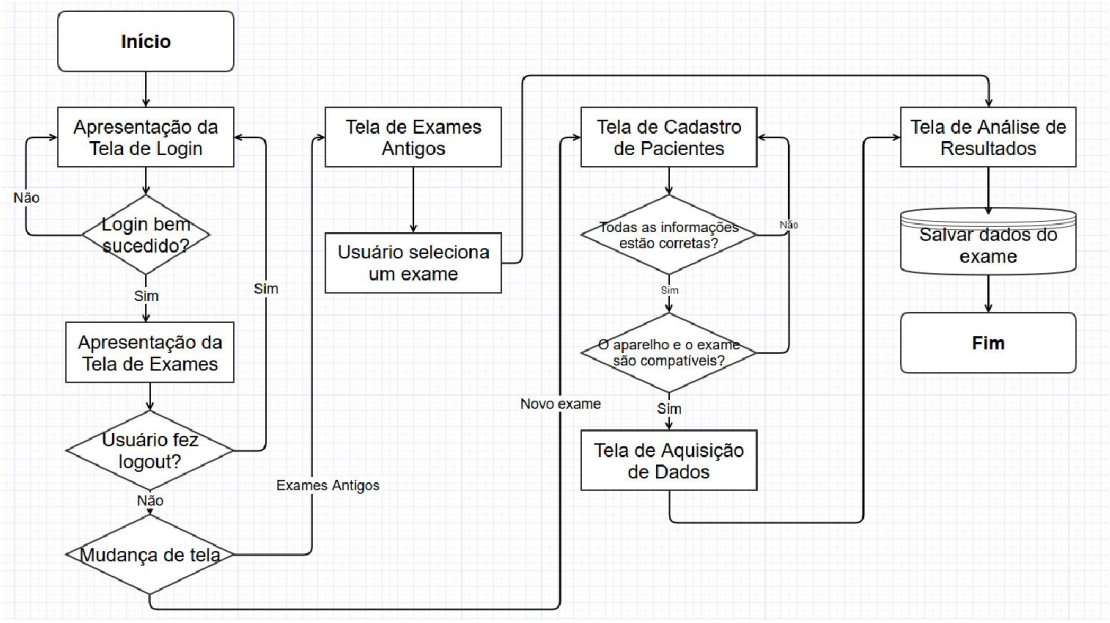
Para a criação da interface gráfica, foi utilizada a biblioteca PyQt4 que é uma biblioteca que integra as funcionalidades do Qt a linguagem de programação Python.

2.2 DIAGRAMA EM BLOCOS DO PROJETO

O diagrama em blocos da Figura 20 demonstra o funcionamento do sistema final como um todo. O sistema apresenta inicialmente a tela de *login* na qual o usuário cadastrado deve informar suas credenciais para prosseguir, após isso, o sistema abrirá a Tela de Exames na qual o usuário deve optar entre realizar um novo exame, verificar um antigo ou realizar o *logout* do sistema, retornando a tela de *login*, caso ele opte por escolher um exame antigo, ele irá ser redirecionado a Tela de Exames Antigos onde terá acesso a todos os exames antigos; caso ele opte por realizar um novo exame, ele será redirecionado a tela de cadastro de pacientes. Na tela de cadastro de pacientes, ele deve colocar todas as informações pertinentes ao paciente necessárias para realizar o registro do exame no sistema incluindo o exame que deseja realizar e qual aparelho será utilizado para a aquisição de dados, preenchidas todas as informações o usuário prossegue para uma tela de aquisição de dados na qual a aquisição de dados será realizada apresentando os canais utilizados pela placa de aquisição.

Após adquirir os dados necessários para o exame, é aberta a tela de análise de resultados na qual é possível realizar as medições necessárias por meio de cursores em gráficos que apresentam os sinais já processados.

Figura 20 - Diagrama em blocos do projeto



Fonte: (Elaborado pelo autor, 2017)

3 IMPLEMENTAÇÃO DO PROJETO

O capítulo de implementação do projeto está dividido em quatro etapas:

- Implementação dos *scripts* de processamento de sinais
- Implementação da interface gráfica em LabVIEW
- Desenvolvimento do sistema de aquisição de dados da placa ADS1X98ECG FE
- Desenvolvimento do software final e integração das etapas
- Validação dos resultados

3.1 IMPLEMENTAÇÃO DOS SCRIPTS DE PROCESSAMENTO DE SINAIS

Para a construção do vetor magnitude que é necessário para o desenvolvimento do exame de Eletrocardiograma de Alta Resolução, são realizadas seis etapas de processamento do sinal capturado pela placa ADS1298.

A primeira etapa trata da conversão dos oito sinais capturados em três sinais ortogonais, aplicando a Transformada Inversa de Dower, após essa transformada é possível que os batimentos dos três sinais estejam desalinhados.

A segunda etapa consiste no alinhamento dos batimentos de cada um dos três sinais, esse processo é chamado de Remoção do *Baseline Drift*. O *baseline drift* é um fenômeno que ocorre por existência de componentes de frequência baixa que apresentam esse comportamento de desalinhamento. Verificou-se que o filtro mais adequado para a remoção do *baseline drift* foi o filtro digital IIR passa-alta do tipo Chebyshev II com banda de parada de 0,17Hz e banda passante de 1,17Hz com a atenuação de -40dB na banda de parada.

A terceira etapa corresponde a criação de um sinal de *template* que é o sinal cuja forma é tomada como base para a detecção dos outros batimentos, nessa etapa é feita a detecção de todos os batimentos que serão utilizados na próxima etapa por meio da aplicação da técnica de Correlação Cruzada.

A quarta etapa consiste na aplicação da Média Coerente de todos os batimentos encontrados na etapa anterior com o objetivo de obter um sinal de *template* com o efeito dos ruídos amenizado visto que os ruídos são sinais de comportamento

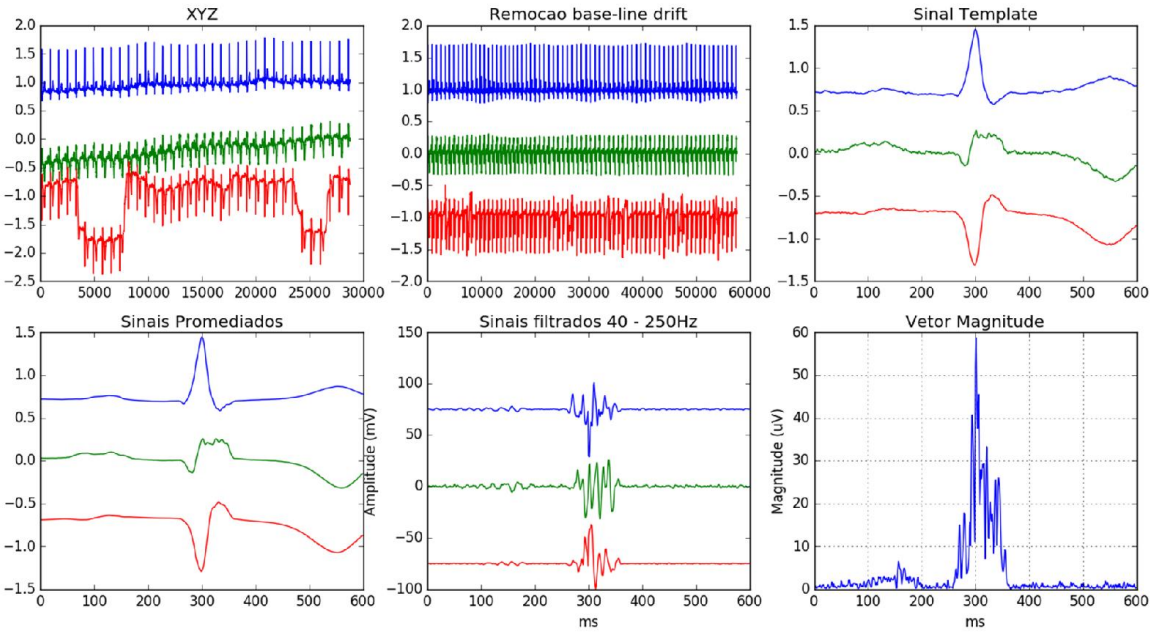
aleatório e de alta frequência.

A quinta etapa consiste na aplicação do filtro IIR do tipo Butterworth de 4ª ordem passa-faixa com frequências de corte de 40Hz e 250Hz, esta faixa de frequência é a faixa na qual é feita a análise dos potenciais tardios.

Na sexta etapa, é realizada a média quadrada dos três sinais obtidos na quinta etapa, obtendo um sinal resultante chamado de vetor Magnitude no qual são extraídos os 3 parâmetros necessários para o diagnóstico.

A Figura 21 apresenta os sinais obtidos em cada uma das etapas e a biblioteca que contêm todas as funções necessárias para realizar essas etapas (ecgartools.py) pode ser vista no apêndice A.

Figura 21 - Etapas dos scripts de Processamento de Sinais



Fonte: (Elaborado pelo autor, 2017)

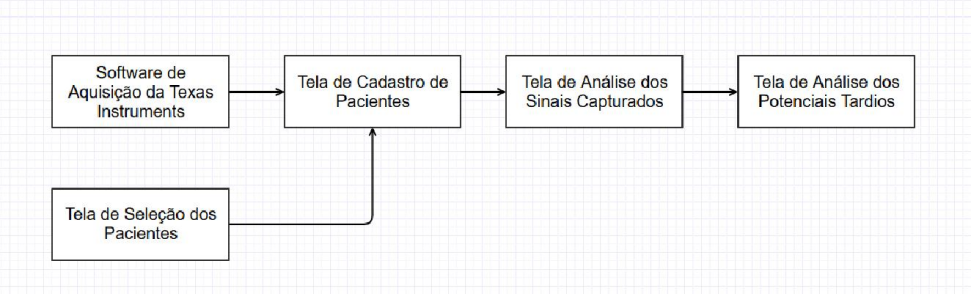
3.2 IMPLEMENTAÇÃO DA INTERFACE GRÁFICA EM LABVIEW

Para a interface gráfica em LabVIEW foram criadas quatro telas que foram integradas ao software desenvolvido pela Texas Instruments, estas telas são:

- Tela de Cadastro de Pacientes
- Tela de Análise dos Sinais Capturados
- Tela de Análise dos Potenciais Tardios
- Tela de Seleção dos Pacientes

A interação entre essas telas pode ser representada no fluxograma da Figura 22.

Figura 22 - Fluxograma das Telas do Software em LabVIEW



Fonte: (Elaborado pelo autor, 2017)

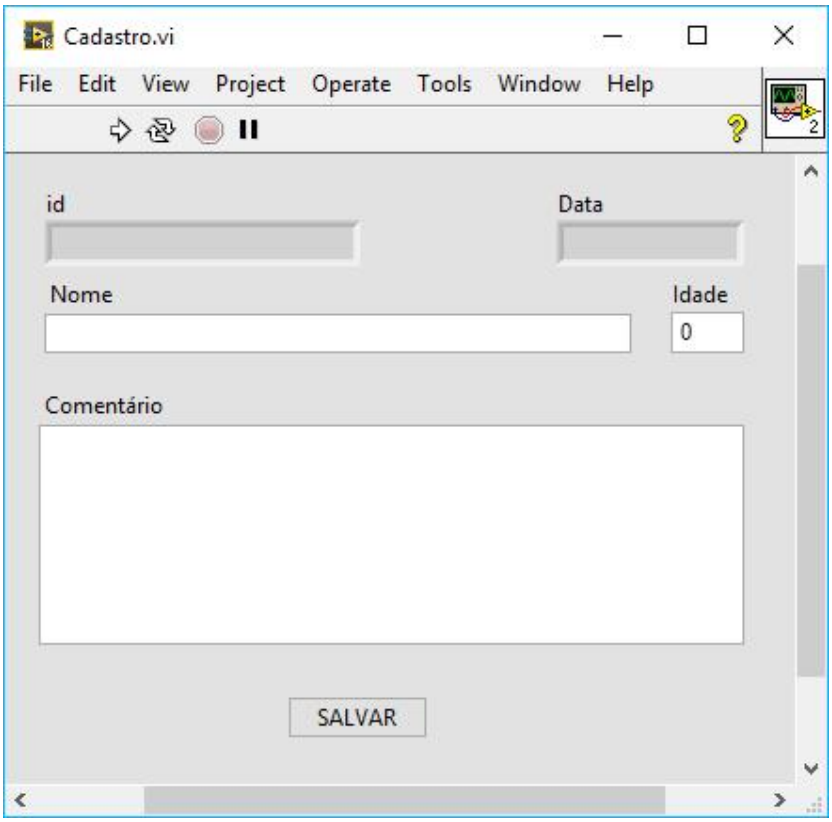
Para o propósito do trabalho, o sistema em LabVIEW não é adequado visto que o LabVIEW é um software de licença paga, mas foi necessário desenvolver essa etapa, pois paralelamente ao Trabalho de Conclusão estava sendo desenvolvido um artigo para o Congresso Brasileiro de Eletromiografia e Cinesiologia e seria necessário o desenvolvimento desse sistema.

3.2.1 Tela de Cadastro de Pacientes

Após adquirir os dados necessários no Software de Aquisição da Placa ADS1298ECGFE-PDK, o usuário é redirecionado a esta tela onde um número de identificação de 9 dígitos é gerado automaticamente de forma aleatória (caso o número gerado já esteja cadastrado, é gerado outro número até que se encontre um número de 9 dígitos não cadastrado). Nesta tela (pode ser vista na Figura 23), o

usuário pode cadastrar o Nome, Idade e um Comentário caso necessário, estes dados serão cadastrados em um banco de dados SQLite3, após finalizado o preenchimento, deve-se clicar no botão SALVAR para prosseguir para a Tela de Análise dos Sinais Capturados.

Figura 23 - Tela de Cadastro de Pacientes



Fonte: (Elaborado pelo autor, 2017)

3.2.2 Tela de Análise dos Sinais Capturados

Nesta tela (pode ser vista na Figura 24) são apresentados os dados capturados pelo software da Texas Instruments, caso essa tela seja chamada pela tela de Cadastro de Pacientes, o sistema irá chamar o script de Remoção de *Baseline Drift* e de Remoção de Altas Frequências para poder obter um sinal alinhado e com menos ruídos.

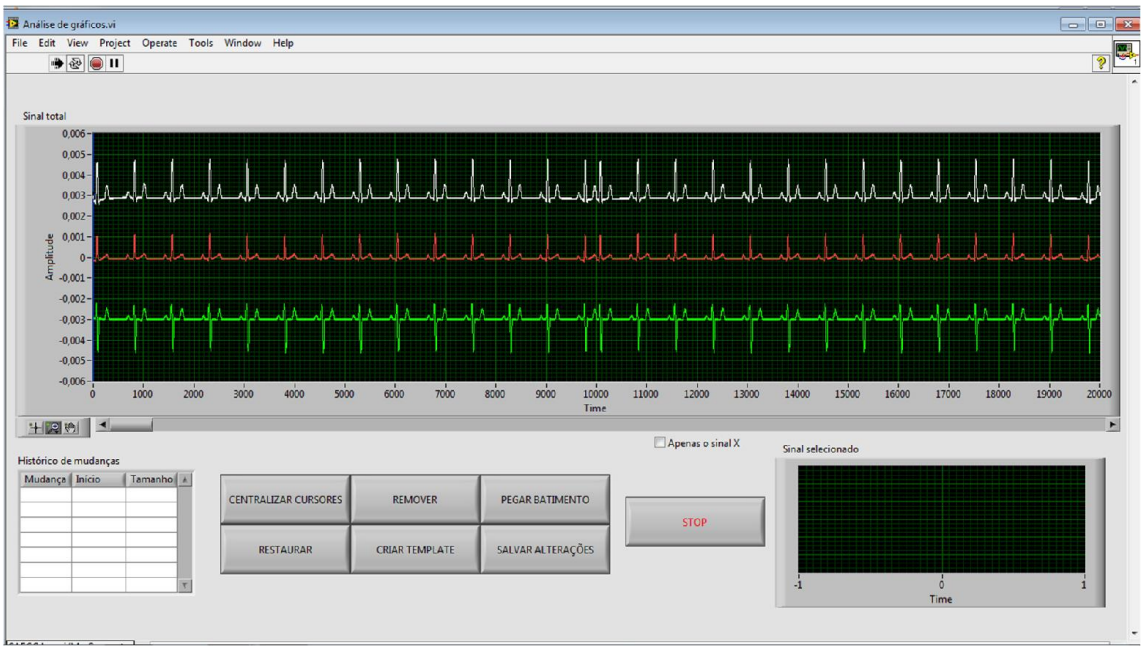
Nesta tela é possível realizar a remoção de batimentos indesejados os quais os filtros não puderam eliminar, pois alguns desses batimentos afetam diretamente componentes de frequência que estão contidas dentro da faixa de frequências em

análise, logo sua atenuação pode afetar o resultado final. Para poder fazer a seleção dos dados a serem removidos, o sistema dispõe de dois cursores os quais são utilizados para delimitar a faixa de dados que devem ser removidas. Para facilitar a visualização, ao posicionar os dois cursores é possível observar a faixa de dados selecionada no gráfico no canto inferior direito da tela.

Cada um dos botões contidos na tela possui uma ação específica, essas ações são:

- Centralizar Cursores: Os cursores são movidos para o centro do gráfico, o primeiro cursor fica na posição equivalente a dois quintos da largura do gráfico e o segundo cursor fica na posição equivalente a três quintos da largura do gráfico.
- Remover: Esta ação serve para remover a faixa de dados que está contida entre os dois cursores, essa remoção é registrada no Histórico de Mudanças que pode ser localizado no canto inferior esquerdo da tela.
- Pegar Batimento: O sistema posiciona automaticamente os cursores no batimento mais próximo detectado, o primeiro estando no início do batimento e o segundo no final.
- Restaurar: Ao selecionar um elemento do Histórico de Mudanças, é possível desfazer todas as ações posteriores a essa mudança.
- Criar Template: O sinal selecionado entre os cursores será utilizado como *template* para os scripts de Correlação, Média dos Sinais e Aplicação do Filtro Butterworth de forma a obter o sinal de ECGAR.
- Salvar Alterações: Todas as alterações feitas no sinal e o histórico de mudança são salvos em arquivos.

Figura 24 - Tela de Análise dos Sinais Capturados

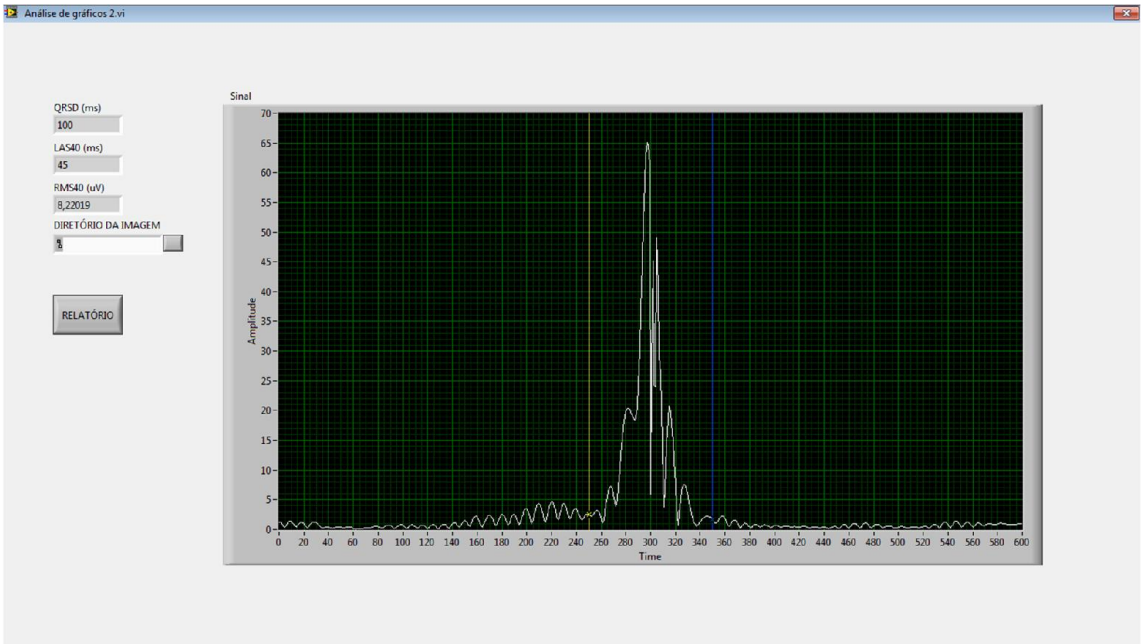


Fonte: (Elaborado pelo autor, 2017)

3.2.3 Tela de Análise dos Potenciais Tardios

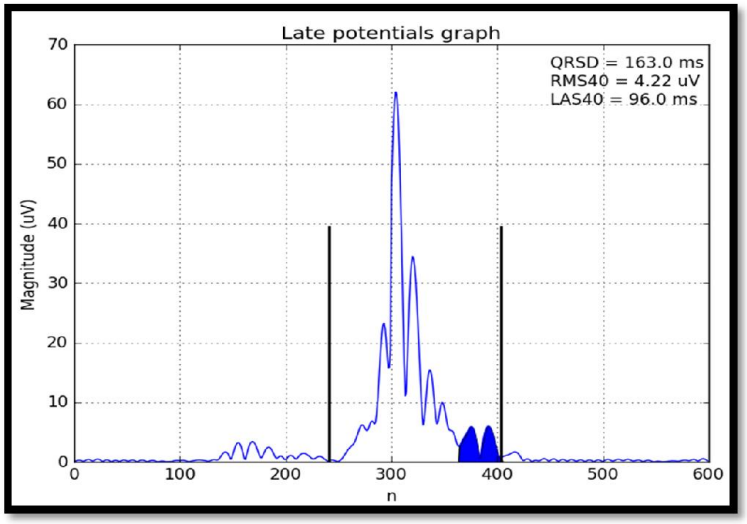
Nessa tela (pode ser vista na Figura 25), é possível ver o vetor magnitude resultante da média quadrática dos *templates* dos sinais X, Y e Z e a partir dele é possível obter os três parâmetros necessários para o exame do ECGAR, o QRSd, o LAS40 e o RMS40. Essa tela dispõe de dois cursores e à medida que o usuário os move no gráfico, o sistema calcula os três parâmetros com base nos dados que estão contidos nessa seleção, com estes cursores, o usuário deve delimitar o segmento QRS do sinal. Quando o usuário define completamente o segmento desejado, deve clicar no botão Relatório, isso gerará uma imagem que conterá um gráfico apresentando o vetor magnitude, a faixa selecionada, os três parâmetros do ECGAR em suas respectivas unidades e a porção do sinal que foi utilizada para definir o parâmetro LAS40, esse resultado pode ser visto na Figura 26.

Figura 25 - Tela de Análise dos Potenciais Tardios



Fonte: (Elaborado pelo autor, 2017)

Figura 26 - Resultado da Análise dos Potenciais Tardios

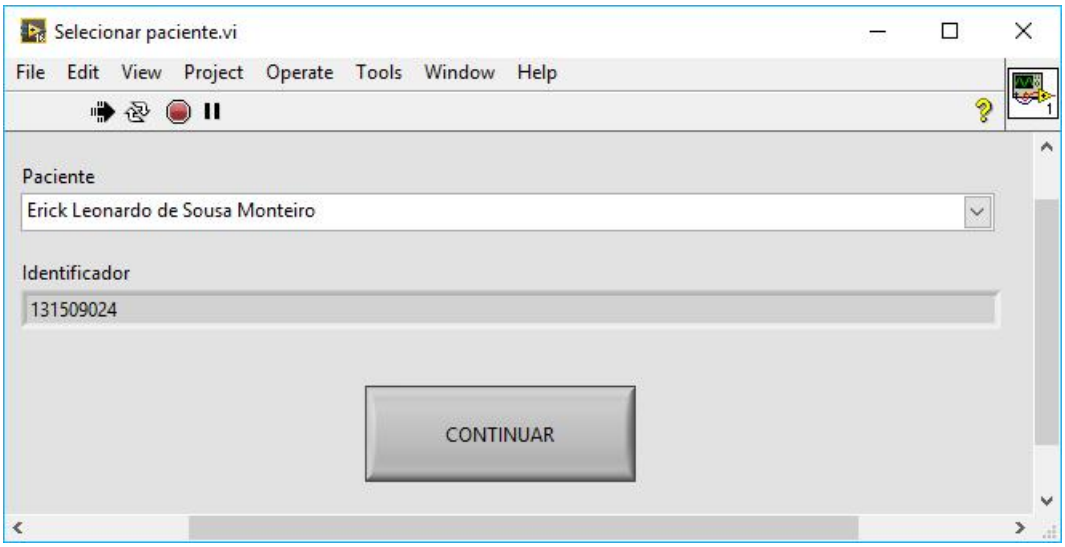


Fonte: (Elaborado pelo autor, 2017)

3.2.4 Tela de Seleção dos Pacientes

Nessa tela (pode ser vista na Figura 27), é possível escolher algum paciente para acessar os dados de seu exame e modificá-los. Para escolher algum paciente, deve-se clicar no *Combo Box* Paciente e selecionar o paciente desejado, ao selecionar o paciente, o campo Identificador é preenchido com o identificador gerado no cadastro.

Figura 27 - Tela de Seleção de Pacientes



Fonte: (Elaborado pelo autor, 2017)

3.3 DESENVOLVIMENTO DO SISTEMA DE AQUISIÇÃO DE DADOS DA PLACA ADS1X98ECG FE

Para desenvolver o software totalmente em linguagem Python, foi necessário entender o funcionamento do sistema de aquisição da placa ADS1X98ECG FE. Esta parte do código desenvolvido pela Texas Instruments utiliza uma biblioteca já compilada, portanto não seria possível analisar o código desenvolvido e traduzí-lo para a linguagem Python, logo foi necessário estudar o *datasheet* do dispositivo.

Após estudar o *datasheet* do ADS1298, percebeu-se que seria necessário utilizar o protocolo SPI para realizar a comunicação entre o Raspberry Pi e a placa ADS1X98ECG FE. Com o uso de *jumpers*, foram feitas as ligações entre os pinos do Raspberry Pi e a placa ADS1X98ECG FE, as relações entre os pinos podem ser vistas no Quadro 3.

Quadro 3 - Relação entre os pinos do Raspberry Pi e da placa ADS1X98ECG FE

Raspberry Pi	ADS1X98ECG FE
Pino 1 (3.3V)	*Header* J4 – Pino 9 (DVDD)
Pino 2 (5V)	*Header* J4 – Pino 10 (AVDD)
Pino 6 (GND)	*Header* J3 – Pino 18 (AVSS)
Pino 19 (MOSI)	*Header* J3 – Pino 11 (SPI_IN)
Pino 20 (GND)	*Header* J4 – Pino 5 (DGND)
Pino 21 (MISO)	*Header* J3 – Pino 13 (SPI_OUT)
Pino 23 (SCLK)	*Header* J3 – Pino 3 (SPI_CLK)
Pino 36 (GPIO16)	*Header* J3 – Pino 1 (SPI_CS)
Pino 37 (GPIO26)	*Header* J3 – Pino 14 (SPI_START)
Pino 38 (GPIO20)	*Header* J3 – Pino 15 (SPI_DRDY)
Pino 40 (GPIO21)	*Header* J3 – Pino 8 (RESET)

Fonte: (Elaborado pelo autor, 2017)

Os registros do circuito integrado ADS1298 são voláteis, então sempre que o circuito integrado for energizado, ele estará em sua configuração padrão que não é a adequada para o sistema. A configuração utilizada para a aquisição foi:

$Registro\ 01h\ CONFIG1\ (85h)$: $HR\ (80h) + DR2\ (04h) + DR0\ (01h)$

O bit HR representa que o sistema está trabalhando em modo de alta resolução, isto é, está utilizando todos os 24-bits para a aquisição. A composição dos bits DR2 e DR0 que a aquisição será realizada na taxa de amostragem de 1000 amostras por segundo.

$Registro\ 02h\ CONFIG2\ (00h)$

Com nenhum bit ativo, esse registro irá adquirir os seus sinais dos eletrodos ligados ao conector DB15.

$Registro\ 03h\ CONFIG3\ (DCh)$: $\overline{PD_{REFBUF}}\ (80h) + BIT6(40h) + RLD_{MEAS}(10h)$
$+ RLDREF_{INT}(08h) + \overline{PD_{RLD}}(04h)$

Com o bit $\overline{PD_{REFBUF}}$ ativo, a placa habilita o *buffer* interno do *power-down*, o bit 6 deve sempre estar ativo, o bit RLD_{MEAS} habilita a medição do sinal RLD, o bit

RLD_{MEAS} determina que a referência analógica da medição do sinal RLD é igual a (AVDD – AVSS) / 2, neste caso, 2.5V; e o bit $\overline{PD_{RLD}}$ determina que o *buffer* RLD está ligado.

$Registros\ 05h\ a\ 0Ch\ CHxSET\ (60h)$: $GAINx1\ (40h) + GAINx0\ (20h)$

Os registradores CH1SET, CH2SET, CH3SET, CH4SET, CH5SET, CH6SET, CH7SET e CH8SET definem as configurações de cada canal, como todos os canais devem ter a mesma configuração, todos eles possuem os mesmos bits ativos, estes bits $GAINx1$ e $GAINx0$ são utilizados para definir que o ganho dos amplificadores PGA é igual a 3.

$Registro\ 0Dh\ RLD_{SENSP}(06h)$: $RLD3P\ (04h) + RLD2P(02h)$

O registrador RLD_{SENSP} define quais os canais de sinal positivo serão utilizados para a derivação do sinal RLD. Com essa configuração do registrador, os canais de sinal positivo utilizados são os canais 2 e 3.

$Registro\ 0Eh\ RLD_{SENSN}\ (02h)$: $RLD2N\ (02h)$

O registrador RLD_{SENSN} define quais os canais de sinal negativo serão utilizados para a derivação do sinal RLD. Com essa configuração do registrador, o canal de sinal negativo utilizado é o canal 2.

$Registro\ 18h\ WCT1\ (0Ah)$: $\overline{PD_{WCTA}}\ (08h) + WCTA1(02h)$

O registrador WCT1 define qual o sinal que será utilizado para o cálculo do sinal do WCT. Com a configuração utilizada, o canal de sinal positivo 2 é conectado ao amplificador WCTA.

$Registro\ 19h\ WCT2\ (E3h)$: $\overline{PD_{WCTC}}\ (80h) + \overline{PD_{WCTB}}\ (40h) + WCTB2\ (20h)$
$+ WCTC1\ (02h) + WCTC0\ (01h)$

O registrador WCT2 define quais sinais serão utilizados para o cálculo do sinal do WCT. Com a configuração utilizada, o canal de sinal positivo 3 é conectado ao amplificador WCTB e o canal de sinal negativo 2 é conectado ao amplificador WCTC.

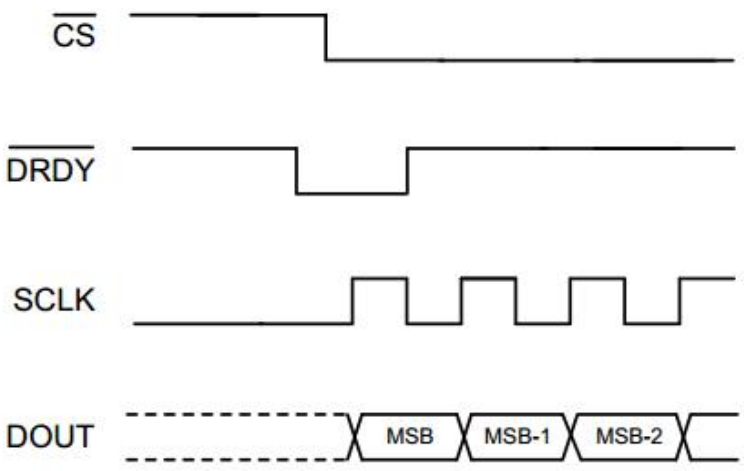
Para realizar a escrita dos valores no registrador, é preciso utilizar o protocolo

SPI, onde devem ser mandados três bytes em sequência, o valor do primeiro byte é $40h$ mais o endereço do registrador desejado, o segundo é $0h$ e o terceiro é o valor desejado.

Para o envio de um comando, é preciso mandar apenas o valor do byte do comando. Os comandos utilizados para a aquisição são o START $(08h)$ que é enviado para que o ADS1298 inicie as conversões, STOP $(0Ah)$ que é enviado para que o ADS1298 pare de realizar as conversões, o comando RDATAC $(10h)$ no qual os comandos de leitura terão como retorno os dados das conversões realizadas (neste modo não é possível realizar a escrita dos registradores) e o comando SDATAC $(11h)$ permite que o sistema leia ou escreva valores nos registradores.

A leitura dos dados das conversões é feita assim que houver um pulso negativo no pino DRDY indicando que foi feita a conversão de uma amostra (a Figura 28 exemplifica esse processo de leitura), a partir desse momento, o Raspberry Pi enviará 27 bytes de valor 0h para ler os 27 bytes de cada amostra. Essa leitura deve ser realizada em menos de 1ms (que é o período para a taxa de amostragem de 1000 amostras por segundo), logo o *clock* do SPI utilizado deve ser capaz de mandar todos os 216 pulsos de *clock* no pino SCLK e os respectivos tempos residuais em menos de 1ms. Como a linguagem utilizada foi a linguagem Python que é uma linguagem de alto nível (e por consequência, suas instruções levam mais tempo que uma de baixo de nível) e o hardware utilizado (Raspberry Pi 3) para o tratamento destes dados não é dedicado apenas a essa tarefa e roda um sistema operacional, o tempo residual é muito grande.

Figura 28 - Processo de leitura de amostras do protocolo SPI

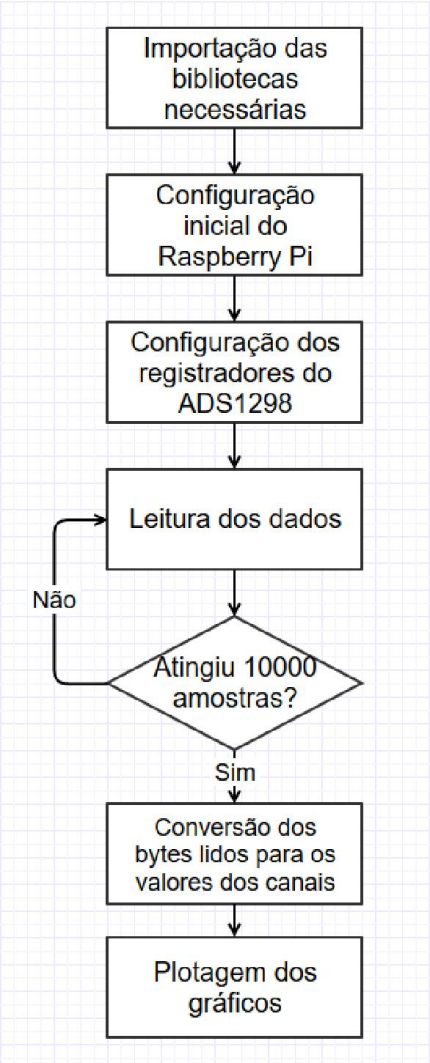


Fonte: (TEXAS INSTRUMENTS, 2015)

Consequentemente, a frequência de *clock* utilizada teve que ser bem mais alta que a frequência necessária para enviar apenas os 216 pulsos de *clock* (300kHz considerando o tempo entre bytes enviados), tendo resultados satisfatórios apenas com frequências superiores a 1MHz.

Para a construção do código de aquisição dos dados do SPI, adotou-se o processo descrito no diagrama em blocos da Figura 29.

Figura 29 - Fluxograma de aquisição de dados

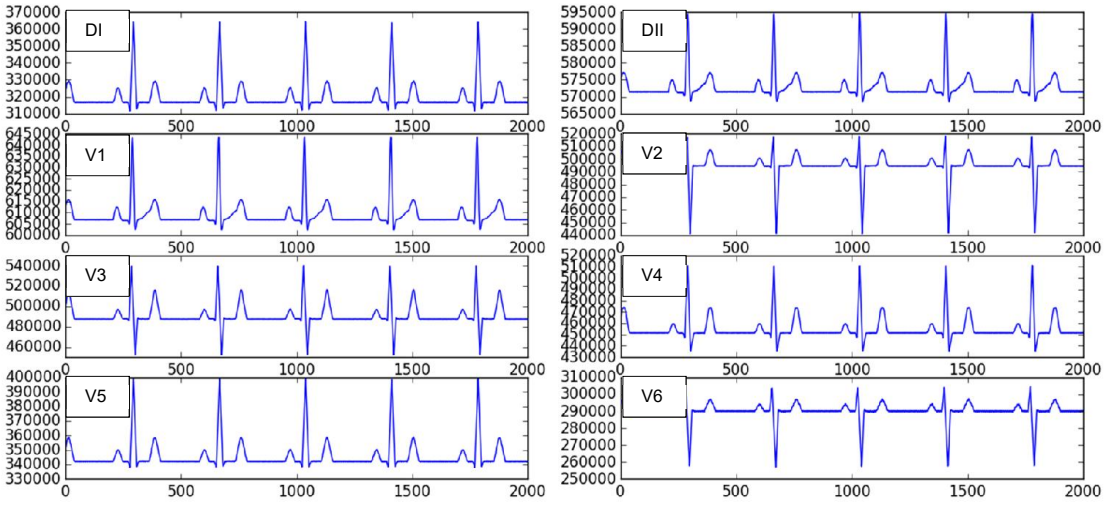


Fonte:(Elaborado pelo autor, 2017)

O código utilizado para obter o gráfico apresentado na Figura 30 pode ser visto no apêndice B. Paralelamente foi analisado o resultado da comunicação em um analisador de sinais digitais como pode ser visto na Figura 31, esta figura apresenta

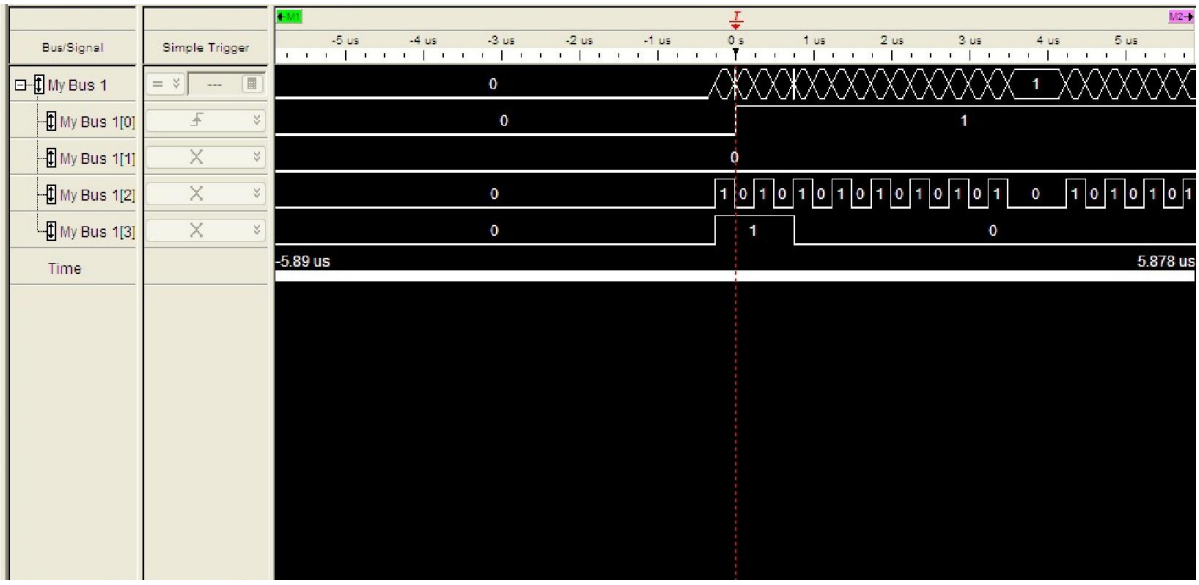
em ordem os sinais dos pinos MISO, GND, CS e DRDY do Raspberry Pi.

Figura 30 - Resultado da Leitura dos oito canais do ADS1298



Fonte: (Elaborado pelo autor, 2017)

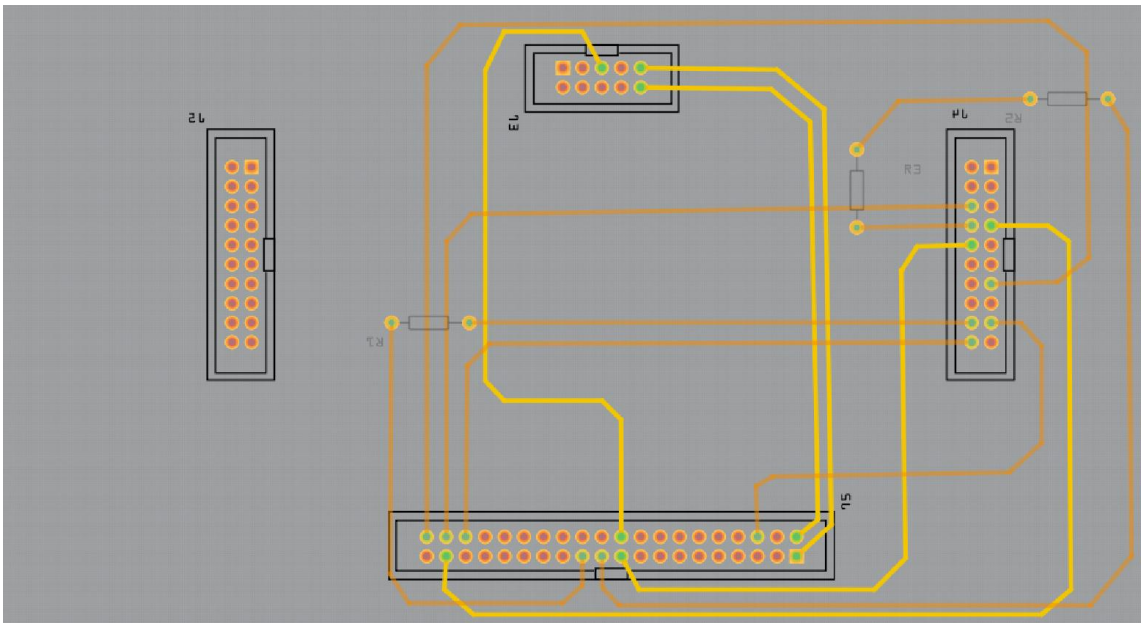
Figura 31 - Comunicação SPI do sistema



Fonte: (Elaborado pelo autor, 2017)

Para a finalização desta etapa, foi utilizada uma placa universal perfurada para desenhar o circuito de ligação dos pinos do Raspberry Pi e da placa ADS1X98ECG FE, o esquemático dessa placa foi desenvolvido no software Fritzing e pode ser visto na Figura 32.

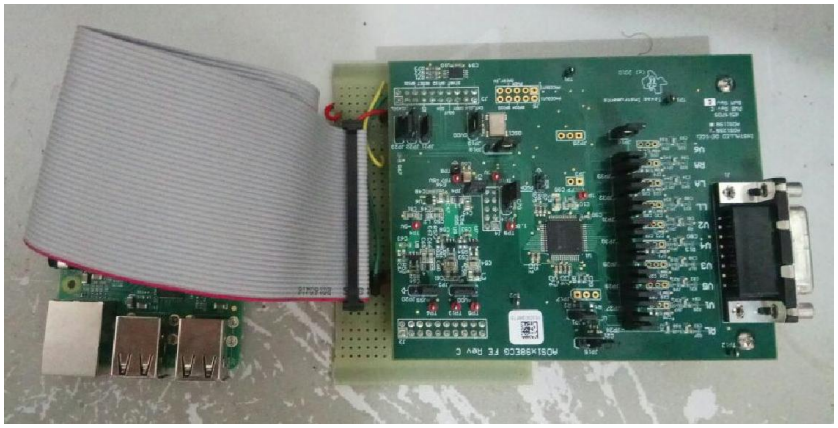
Figura 32 - Layout da Placa Desenvolvida



Fonte: (Desenvolvido pelo autor, 2017)

Esta placa é anexada abaixo da placa ADS1X98ECG FE e conectada ao Raspberry Pi via cabo flat, desse modo funcionando como um sistema *plug and play* (do inglês ligue e use), sendo assim um sistema periférico ao Raspberry Pi, a Figura 33 apresenta o sistema totalmente unido.

Figura 33 - Raspberry Pi e ADS1X98ECG FE conectados pela placa desenvolvida



Fonte: (Desenvolvido pelo autor, 2017)

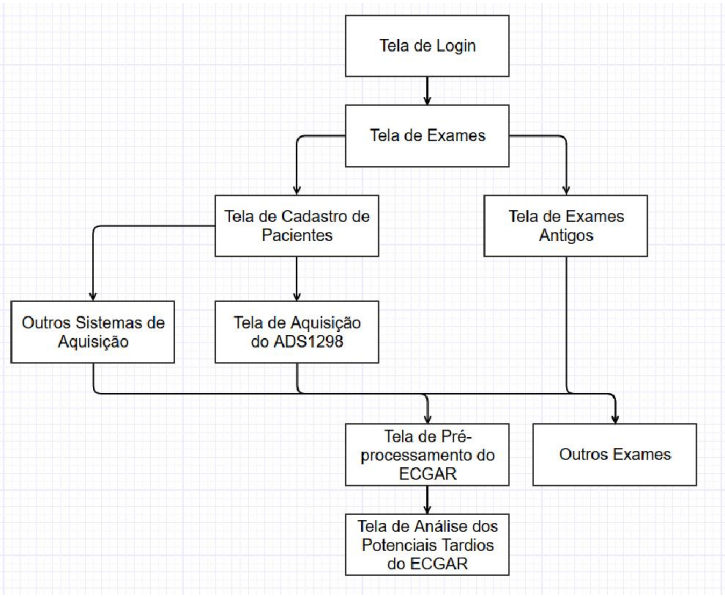
3.4 DESENVOLVIMENTO DO SOFTWARE FINAL E INTEGRAÇÃO DAS ETAPAS

Para o desenvolvimento do *software* em Python foram desenvolvidas sete telas utilizando o *framework* Qt, o *software* em Python foi baseado no paradigma de programação orientada a fluxo, neste caso, todas as telas são programas diferentes (cujos códigos podem ser vistos no repositório do projeto[1]) e a comunicação entre elas é feita por meio de fluxo de dados em arquivos, essa opção pode ser mais lenta, porém possui a vantagem de poder adicionar novos sistemas de aquisição e exames sem a necessidade de refatorar o código já desenvolvido. As telas desenvolvidas são:

- Tela de Login
- Tela de Exames
- Tela de Registro de Pacientes
- Tela de Aquisição do ADS1298
- Tela de Pré-processamento do ECGAR
- Tela de Análise dos Potenciais Tardios do ECGAR
- Tela de Exames Antigos

A interação entre essas telas pode ser vista no diagrama em blocos da Figura 34.

Figura 34 – Fluxograma das Telas do Software em Python



Fonte: (Elaborado pelo autor, 2017)

[1] Disponível em: <https://bitbucket.org/emonteir0/tcc-ecg/src>. Acesso em: 11 de novembro de 2017

Para possibilitar essa adaptabilidade do software, foi adotada uma estrutura de pastas para os dispositivos de aquisição, a pasta principal **acquisition** está localizada no diretório principal do software. Um dispositivo de aquisição do sistema deve possuir uma pasta dentro da pasta *acquisition* que deve conter todos os arquivos necessários para a sua execução incluindo o arquivo **init** que deve ter uma mensagem json que apresente, no mínimo, a seguinte estrutura:

```
{"Name": "#Nome do Dispositivo#", "File": "#Caminho relativo do programa a ser chamado#", "fanout": #Número de sinais que o dispositivo fornece#}
```

Analogamente, os exames possuem uma estrutura de pastas, onde a pasta principal chamada **exams** está localizada no diretório principal do software. Um exame do sistema deve possuir uma pasta dentro da pasta *exams* que deve conter todos os arquivos necessários para a sua execução incluindo o arquivo **init** que deve ter uma mensagem json que apresente, no mínimo, a seguinte estrutura:

```
{"Name": "#Nome do Dispositivo#", "File": "#Caminho relativo do programa a ser chamado#", "fanin": #Número de sinais que o exame precisa#, "fanout": #Número de sinais que o dispositivo fornece#}
```

Dessa forma, cada tela funciona como um programa independente e estas comunicam entre si através de um fluxo de dados por meio de arquivos temporários. Esse modo de programação tem uma desvantagem em relação ao tempo de execução visto que os sinais capturados devem ser recarregados na memória RAM toda vez que uma nova tela é aberta, porém, como o fluxo de dados é baixo (cerca de 4,57MB para a gravação de 10 minutos de um canal com taxa de amostragem de 1000 SPS), o tempo adicional não compromete o uso do sistema. Em contrapartida, a vantagem obtida por utilizar esse método possibilita o desenvolvimento paralelo facilitado de novos programas para realização de aquisição e de exames, visto que as interações com a aplicação principal são minimizadas.

3.4.1 Tela de Login

Nesta tela (pode ser vista na Figura 35), o usuário deve preencher os campos Usuário e Senha com os seus respectivos usuário e senha que devem estar registrados no banco de dados do sistema, após preenchidos o usuário deve clicar no botão Login.

Se o usuário clicar em Login deixando algum dos campos em branco, o sistema

irá mostrar uma janela *pop-up* com a mensagem “Campo usuário ou senha em branco”. Caso o usuário clique em Login informando credenciais não registradas no banco de dados, o sistema mostrará uma janela *pop-up* com a mensagem “Usuário inexistente”. Caso o usuário clique no botão Login informando um usuário existente, porém uma senha errada, o sistema mostrará uma janela *pop-up* com a mensagem “Senha incorreta”. E caso nenhuma dessas situações ocorrerem, as credenciais informadas foram corretas e o usuário pode prosseguir para a próxima tela.

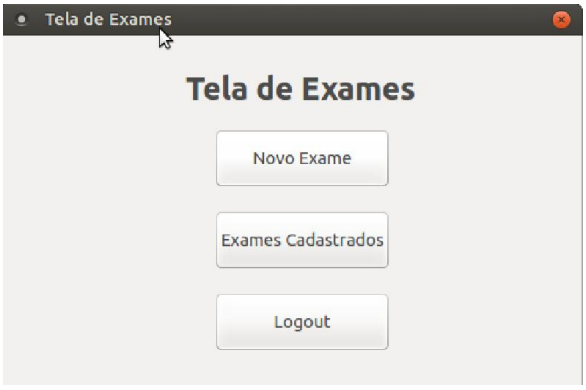
Figura 35 - Tela de Login

Fonte: (Elaborado pelo autor, 2017)

3.4.2 Tela de Exames

Nesta tela (pode ser vista na Figura 36), o usuário pode escolher realizar um novo exame clicando no botão Novo Exame, verificar um exame já cadastrado ao clicar no botão Exames Cadastrados ou realizar o *logout* clicando no botão Logout.

Figura 36 - Tela de Exames



Fonte: (Elaborado pelo autor, 2017)

3.4.3 Tela de Registro de Pacientes

Nesta tela (pode ser vista na Figura 37), o usuário deve preencher as seguintes informações do paciente: Nome, Gênero, Data de Nascimento, RG, CPF e Comentários caso se julgue necessário. Também é necessário informar o exame que se deseja realizar e qual o aparelho deve ser utilizado, para voltar a Tela de Exames, deve-se clicar no botão Voltar e para prosseguir para uma Tela de Aquisição deve-se clicar no botão Avançar.

Caso o usuário deixa alguma das informações em branco (com exceção do Comentário) ou preencha o RG ou CPF com números a menos, o sistema mostrará uma janela *pop-up* com a mensagem “Preencha corretamente todos os campos obrigatórios”. Caso o usuário selecione um aparelho cujo *fanout* de seu arquivo init não seja igual ao *fanin* do arquivo init do exame selecionado, o sistema mostrará uma janela *pop-up* com a mensagem “Exame incompatível com o aparelho selecionado”. Caso nenhuma das situações ocorra, o sistema pode prosseguir para a Tela de Aquisição.

Figura 37 - Tela de Registro de Pacientes

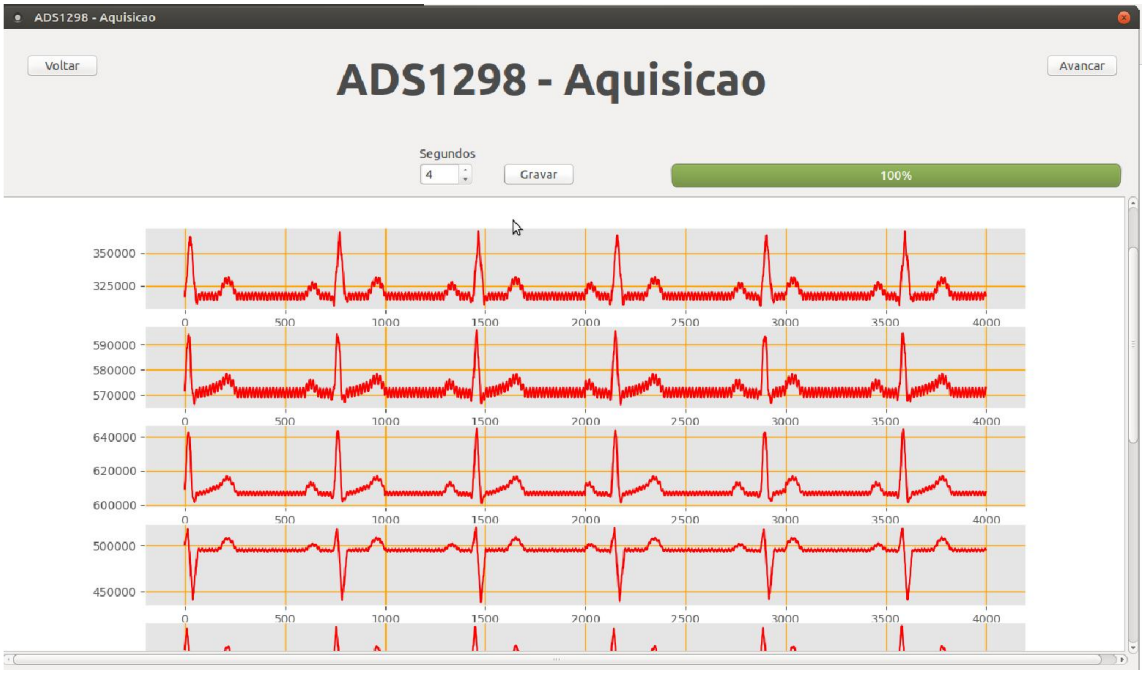
Fonte: (Elaborado pelo autor, 2017)

3.4.4 Tela de Aquisição do ADS1298

Caso o usuário tenha selecionado o ADS1298 como Aparelho na Tela de Registro de Pacientes, a tela que será aberta é a Tela de Aquisição do ADS1298. Essa tela que pode ser vista na Figura 38 apresenta oito gráficos representando os oito canais do ADS1298 que serão inicialmente preenchidos com uma amostra de 4 segundos de teste. O usuário pode definir o número de segundos que deseja gravar (com limite de 3600 segundos) e clicar em Gravar para iniciar a gravação, as funções utilizadas para a gravação são as mesmas que foram desenvolvidas no tópico 3.3.

Caso o usuário deseje voltar a Tela de Registro de Pacientes, deve clicar no botão Voltar e caso deseje prosseguir a Tela de Exame do exame selecionado na Tela de Registro de Pacientes, deve clicar em Avançar.

Figura 38 - Tela de Aquisição do ADS1298



Fonte: (Elaborado pelo ator, 2017)

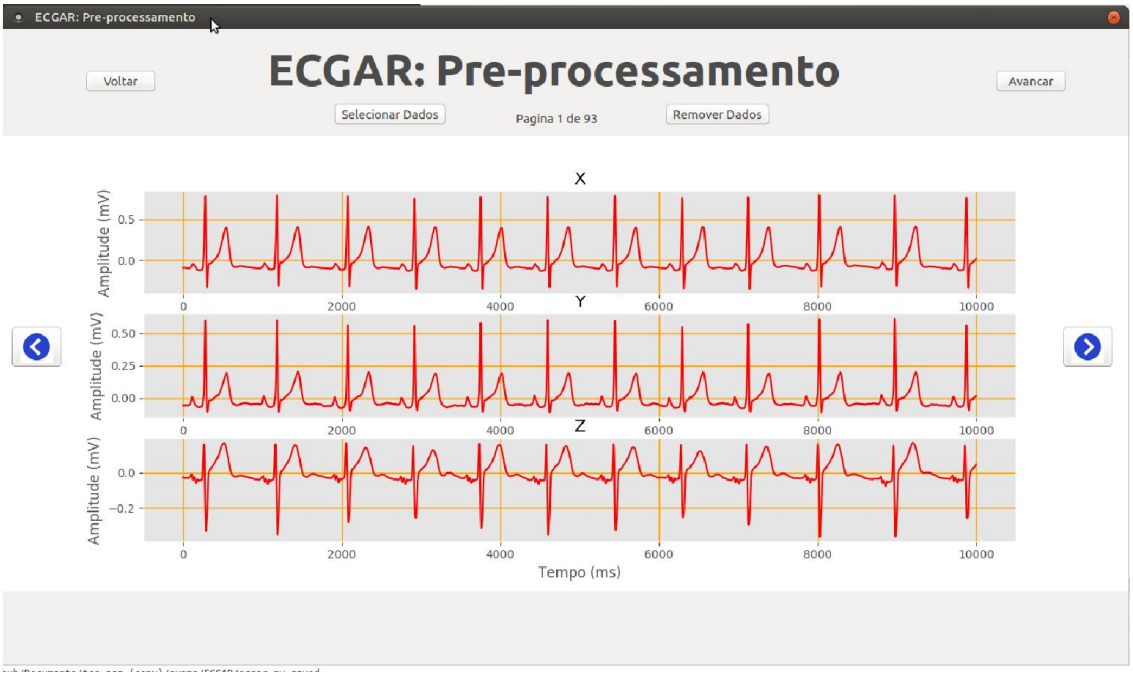
3.4.5 Tela de Pré-Processamento do ECGAR

Nesta tela, que pode ser vista na Figura 39, os oito sinais que foram adquiridos na Tela de Aquisição são convertidos em três sinais ortogonais por meio da Transformada Inversa de Dower e logo em seguida aplica os scripts de Remoção do

Baseline Drift e de Remoção de Altas Frequências, o sistema divide o sistema em páginas, cada página contêm 10000 amostras e a navegação pode ser feita pelos botões laterais. O propósito desta tela é remover os batimentos indesejados os quais os filtros não puderam atenuar, pois alguns desses batimentos afetam diretamente componentes de frequência que estão contidas dentro da faixa de frequências em análise. Para selecionar uma faixa de frequências para a remoção dos dados, deve-se clicar no botão Selecionar Dados, logo após o sistema habilitará a movimentação de um cursor que deve ser posicionado em algum ponto da tela, logo em seguida, é habilitada a movimentação do segundo cursor, o intervalo de amostras. Para remover os dados, deve-se clicar no botão Remover Dados com um intervalo de amostras selecionado.

Caso o usuário deseje voltar à Tela de Aquisição, deve clicar em Voltar e caso deseje prosseguir para a Tela de Análise dos Potenciais Tardios do ECGAR e nesse caso, o sistema irá criar um *template* automaticamente para ser utilizado na próxima tela.

Figura 39 - Tela de Pré-Processamento do ECGAR

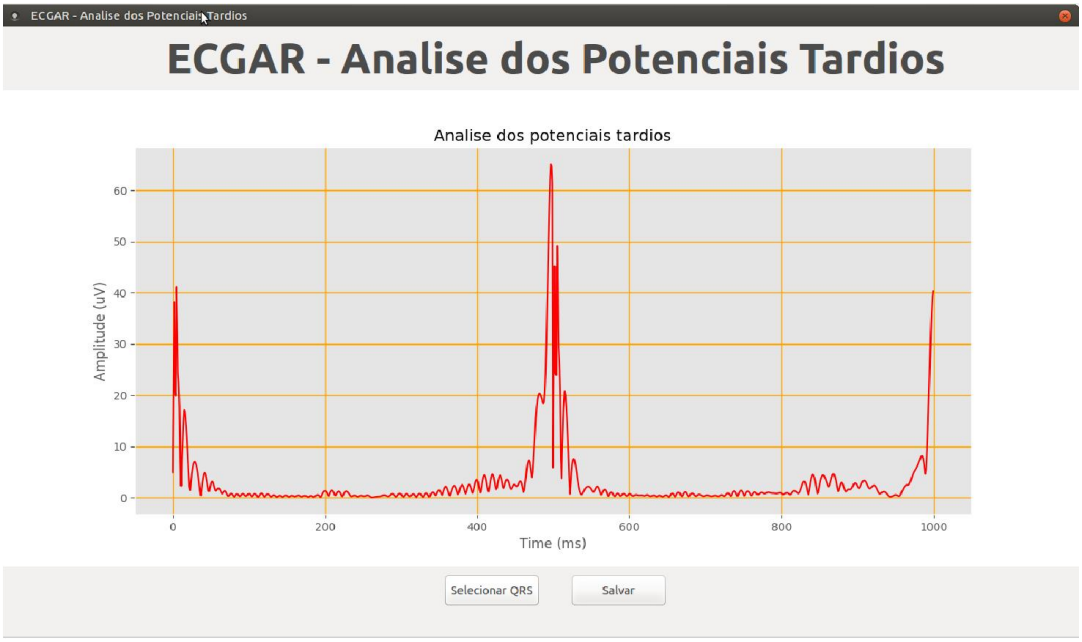


Fonte: (Elaborado pelo autor, 2017)

3.4.6 Tela de Análise dos Potenciais Tardios do ECGAR

Nesta tela, que pode ser vista na Figura 40, é gerado o vetor magnitude após aplicar os scripts Python de Correlação, Média dos Sinais e Aplicação do Filtro Butterworth. O usuário pode obter os três parâmetros do ECGAR ao clicar em Selecionar QRS onde o sistema irá dispor de dois cursores e após o posicionamento do primeiro, é possível ver a mudança dos parâmetros conforme a movimentação do segundo cursor. Para salvar os sinais obtidos, o usuário deve clicar no botão Salvar.

Figura 40 - Tela de Análise dos Potenciais Tardios

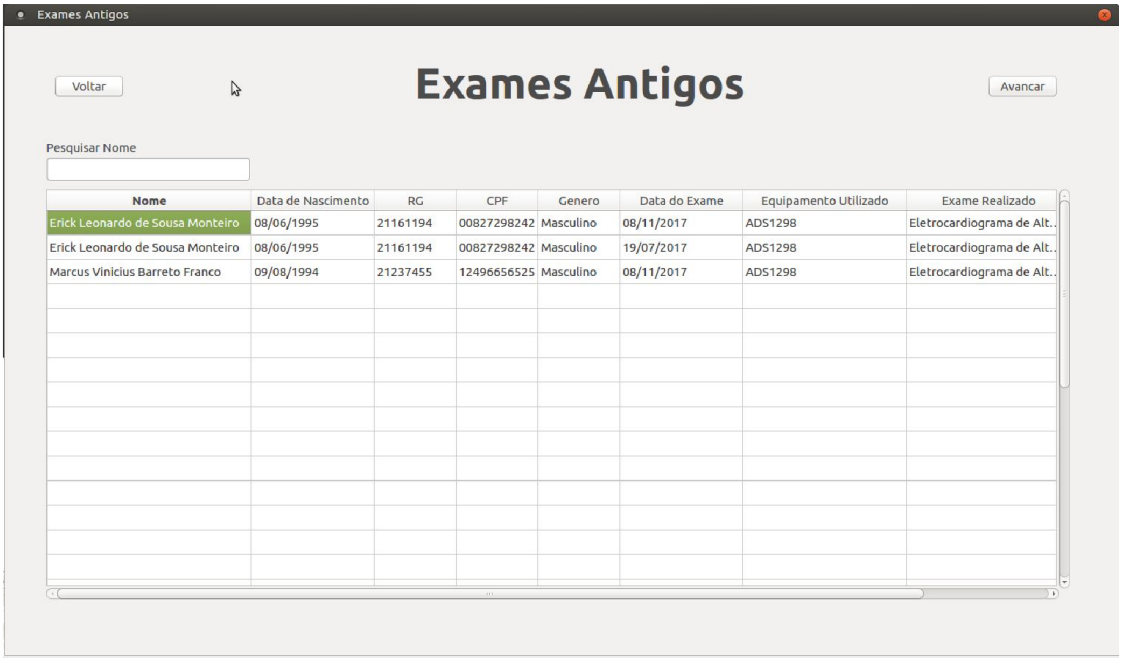


Fonte: (Elaborado pelo autor, 2017)

3.4.7 Tela de Exames Antigos

Nessa tela, que pode ser vista na Figura 41, é possível ver todos os exames que foram feitos pelo usuário que realizou o *login* no sistema e também é possível poder analisá-los novamente ao selecionar o nome de um dos pacientes e clicar no botão Avançar. Essa tela também dispõe de uma barra de pesquisa chamada Pesquisar Nome na qual, conforme for sendo digitada uma palavra nela, o sistema filtra os exames por nome do paciente onde a palavra digitada na barra de pesquisa é um prefixo do nome do paciente. Caso o usuário deseje voltar à Tela de Exames, deve clicar no botão Voltar.

Figura 41 - Tela de Exames Antigos



Fonte: (Elaborado pelo autor, 2017)

4 RESULTADOS OBTIDOS

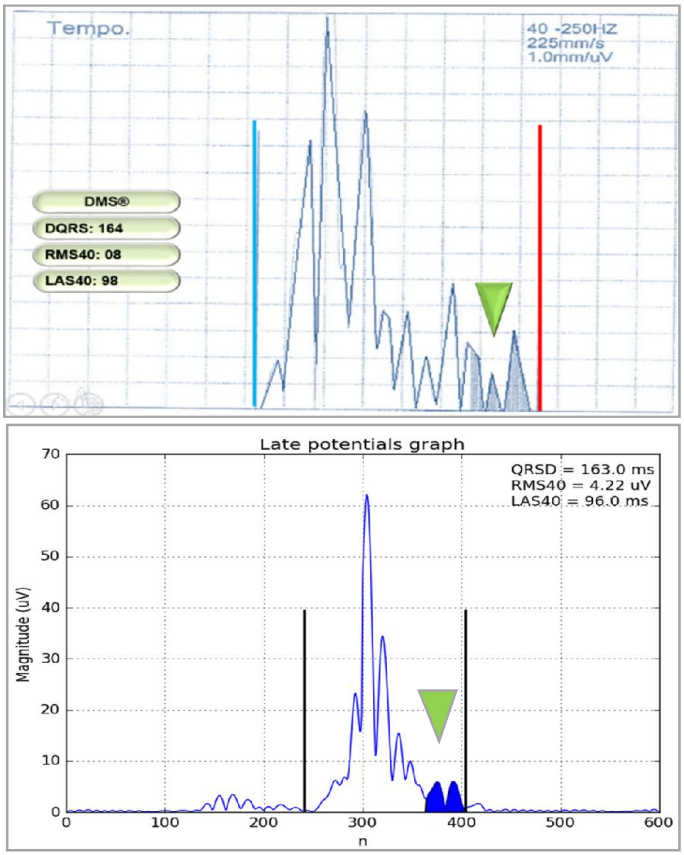
Após a criação de todas as telas, foi possível observar a integração entre todas as etapas do sistema desde a aquisição até o resultado final e consequente armazenamento dos dados, obtendo bons resultados finais.

Verifica-se também que o sistema atende as premissas iniciais de adaptabilidade, pois os programas que realizam a execução e a aquisição de dados são totalmente independentes entre si e do sistema principal, sendo integrados apenas pelo fluxo de dados; e a acessibilidade, pois o sistema desenvolvido possui código aberto desenvolvido em uma linguagem gratuita (Python) publicado em um repositório online e está embarcado em uma placa de baixo custo (Raspberry Pi 3), necessitando apenas dos periféricos básicos (monitor, mouse e teclado) para funcionar.

Os testes de integração do sistema foram realizados utilizando o simulador de sinais de eletrocardiograma (HS-15) e, após testado, o sistema foi utilizado para realizar o teste com pacientes. Foram selecionados 3 dentre os pacientes da clínica para serem testados pelo sistema. Primeiro os pacientes eram examinados utilizando o aparelho da clínica, que utiliza o software CardioScan, e logo após foram examinados pelo software desenvolvido.

A Figura 42 apresenta os resultados de um desses pacientes examinado no software CardioScan e no software desenvolvido neste trabalho.

Figura 42 - Comparação entre os sinais resultantes



Fonte: (Elaborado pelo autor, 2017)

Com base nesse resultado, foi possível verificar, pela proximidade das formas de onda e parâmetros do exame, que o sistema conseguiu realizar o exame de forma similar à forma que é realizado no sistema comercial.

Também como resultado deste trabalho, foi publicado o artigo Detecção de Potenciais Tardios de Ativação Ventricular: Um Protótipo para Eletrocardiograma de Alta Resolução como coautor no Congresso Brasileiro de Eletromiografia e Cinesiologia realizado pela Universidade Federal de Uberlândia.

CONCLUSÃO

Com os resultados obtidos no desenvolvimento desse trabalho conclui-se que foi possível desenvolver um sistema generalizado para processamento e manipulação de sinais de eletrocardiograma, visto que o mesmo conseguiu se integrar com êxito ao sistema de aquisição da placa ADS1298ECG-FE e ao exame de Eletrocardiograma de Alta Resolução.

O sistema apresentou a adaptabilidade desejada visto que o exame e o sistema selecionados não fazem parte do programa principal, mas a comunicação entre os programas é realizada com uma boa eficiência, o que favorece o desenvolvimento paralelo de programas capazes de realizar outros tipos de exame usando outros sistemas de aquisição e capazes de se integrar ao sistema de forma facilitada.

Foram encontradas algumas dificuldades no desenvolvimento do projeto, principalmente relacionadas ao sistema de aquisição utilizado, pois grande parte dos trabalhos que o utilizam recorrem somente ao software fornecido pela Texas Instruments e há pouquíssimos trabalhos integrando-o a outros dispositivos microprocessados. Também foi verificado que como o sistema foi desenvolvido em uma linguagem de alto nível, as instruções da interface gráfica levavam um tempo relativamente grande, na ordem de milissegundos, o que poderia causar atrasos na comunicação SPI resultando na perda de algumas amostras; resultando na indesejada falta de *feedback* para o usuário durante o processo de aquisição.

Este trabalho abre ainda a possibilidade de continuidade, onde o software desenvolvido pode ser refatorado em uma linguagem de baixo nível como C++, que também oferece os recursos do Qt, e o tratamento dos dados não seja feito no sistema principal, mas sim em um hardware dedicado a esta funcionalidade, possibilitando que o sistema possa também ser executado em computadores pessoais sem nenhuma modificação no código.

REFERÊNCIAS

AZEVEDO, Decio F. **Iniciação à Eletrocardiografia**. Porto Alegre: Editora Artes Médicas Sul Ltda, 1999.

BERBARI, E. J.. Principles of Electrocardiography. In: BRONZINO, J. D.. **The Biomedical Engineering Handbook**. 2 ed. Estados Unidos: CRC, 2000. p. 100-130.

CHRISTENSSON, Per. **Python Definition**. Disponível em: <https://techterms.com/definition/python>. Acesso em: 26 abr. 2017.

DM SOFTWARE. **CardioScan**. Disponível em: <http://www.holterdms.com/CardioScan_HolterECG_Systems.php>. Acesso em: 13 nov. 2017.

DOWER, G. E.. A lead synthesizer for the Frank system to simulate the standard 12-lead electrocardiogram. **J Electrocardiology**. Estados Unidos; v. 1, n. 1, p. 101-116, jan. 1968.

GOMIS, P.; JONES, D. L.; CAMINAL, P.. Analysis of abnormal signals within the QRS complex of the high-resolution electrocardiogram. **IEEE Transactions on Biomedical Engineering**. v. 44, n. 8, p. 681-693, mar. 1997.

HAYKIN, Simon. VEEN, Barry Van. **Signals and Systems**. 2 ed. New York: John Wiley & Sons, Inc., 1999.

HEALTH, Center For Devices And Radiological. **Diagnostic ECG Guidance (Including Non-Alarming ST Segment Measurement)**. Disponível em: <https://www.fda.gov/MedicalDevices/ucm073942.htm>. Acesso em: 5 nov. 1998.

IEC. **60601-2-27 Particular requirements for the basic safety and essential performance of electrocardiographic monitoring equipment**. International Standard: IEC, 2011.

JONES, Janice. **How do I obtain raw ECG data**. Disponível em: <https://alivecor.zendesk.com/hc/en-us/articles/231584088-How-do-I-obtain-the-raw-ECG-data->. Acesso em: 23 abr. 2017.

MEDICALMED. **Cabo de ECG 10 vias**. Disponível em: < https://www.wikihow.com/Calculate-Heart-Rate-from-ECG>. Acesso em: 10 jul. 2017.

MELIN, Barbara White; PART-ENANDER, Eva; ISAKSSON, P.; et al. **The MATLAB Handbook**. 1 ed. Boston: Addison-Wesley Longman, 1996.

MIKROELEKTRONIKA. **Digital Filter Design**. Disponível em: <https://learn.mikroe.com/ebooks/digitalfilterdesign/>. Acesso em: 20 abr. 2017.

NATIONAL INSTRUMENTS. **O que é o LabVIEW?**. Disponível em: <http://www.ni.com/newsletter/51141/pt/>. Acesso em: 26 abr. 2017.

ORACLE. **About the Java Technology**. Disponível em: <https://docs.oracle.com/javase/tutorial/getStarted/intro/definition.html>. Acesso em: 23 abr. 2017.

PYTHON SOFTWARE FOUNDATION. **What is Python? Executive Summary**. Disponível em: <https://www.python.org/doc/essays/blurb/>. Acesso em: 26 abr. 2017.

RASPBERRY PI FOUNDATION. **Raspberry Pi 3 Model B**. Disponível em: <https://www.raspberrypi.org/products/raspberry-pi-3-model-b/>. Acesso em:17 out. 2017.

R&D MEDIQ. **Budget Product R&D**. Disponível em: <https://www.rdmediq.com.br/en/budget>. Acesso em: 18 out. 2017.

SERRANO, Luís; ALCOBIA, Carlos; MATEUS, Mário Luis Oliveira de Sousa. Sistemas de aquisição, processamento e armazenamento de dados. In: ENCONTRO NACIONAL - SOCIEDADE PORTUGUESA DE METROLOGIA. 1., 2004, Caparica. **Artigo**... Caparica: SPMet, 2004. p. 1-15.

SPARKFUN. **Serial Peripheral Interface (SPI)**. Disponível em: <https://learn.sparkfun.com/tutorials/serial-peripheral-interface-spi>. Acesso em: 19 abr. 2017.

TEXAS INSTRUMENTS. **Precision Data Converters**. Disponível em: <https://e2e.ti.com/support/data_converters/precision_data_converters/f/73/t/291783>. Acesso em: 15 out. 2017.

WANN, Samuel. Cardiologists make progress but still face problems adopting electronic health records. **Cardiology Today,** Sydney, 1 fev. 2006. Disponível em: <http://www.healio.com/cardiology/practice-management/news/print/cardiology-today/%7B6aa12756-ea53-4092-b999-f8791bde610b%7D/cardiologists-make-progress-but-still-face-problems-adopting-electronic-health-records>. Acesso em: 19 abr. 2017.

WEBMD, LLC. **Electrocardiogram**. Disponível em: <http://www.webmd.com/heart-disease/electrocardiogram#1>. Acesso em: 19 abr. 2017.

WELCH ALLYN. **CardioPerfect Workstation**. Disponível em: <https://learn.sparkfun.com/tutorials/serial-peripheral-interface-spi>. Acesso em: 13 out. 2017.

APÊNDICE A – BIBLIOTECA DOS ALGORITMOS DE PROCESSAMENTO DE SINAIS

ecartools.py

```
import numpy as np
import scipy.signal as scisig
import wfdb


class ECGAR(object):

    def __init__(self):
        pass

    def read_files(self, path, type_file):
        pass

    def convert_to_xyz(self, type_name, sig):
        x = []
        y = []
        z = []
        signals = np.array(sig)
        if type_name == 'Dower':

            x = (-0.172*signals[0] - 0.073*signals[1] + 0.122*signals[2] +
0.231*signals[3] + 0.239*signals[4]
                 + 0.193*signals[5] + 0.156*signals[6] - 0.009*signals[7])

            y = (0.057*signals[0] - 0.019*signals[1] - 0.106*signals[2] -
0.022*signals[3] + 0.040*signals[4]
                 + 0.048*signals[5] - 0.227*signals[6] - 0.886*signals[7])

            z = (-0.228*signals[0] - 0.310*signals[1] - 0.245*signals[2] -
0.063*signals[3] + 0.054*signals[4]
                 + 0.108*signals[5] + 0.021*signals[6] + 0.102*signals[7])

        elif type_name == 'PLSV':

            x = (-0.172*signals[0] - 0.073*signals[1] + 0.122*signals[2] +
0.231*signals[3] + 0.239*signals[4]
                 + 0.193*signals[5] + 0.156*signals[6] - 0.009*signals[7])

            y = (0.057*signals[0] - 0.019*signals[1] - 0.106*signals[2] -
0.022*signals[3] + 0.040*signals[4]
                 + 0.048*signals[5] - 0.227*signals[6] - 0.886*signals[7])

            z = (-0.228*signals[0] - 0.310*signals[1] - 0.245*signals[2] -
0.063*signals[3] + 0.054*signals[4]
                 + 0.108*signals[5] + 0.021*signals[6] + 0.102*signals[7])

        elif type_name == 'QLSV':

            x = (-0.172*signals[0] - 0.073*signals[1] + 0.122*signals[2] +
0.231*signals[3] + 0.239*signals[4]
                 + 0.193*signals[5] + 0.156*signals[6] - 0.009*signals[7])
```

```
            y = (0.057*signals[0] - 0.019*signals[1] - 0.106*signals[2] -
0.022*signals[3] + 0.040*signals[4]
                 + 0.048*signals[5] - 0.227*signals[6] - 0.886*signals[7])

            z = (-0.228*signals[0] - 0.310*signals[1] - 0.245*signals[2] -
0.063*signals[3] + 0.054*signals[4]
                 + 0.108*signals[5] + 0.021*signals[6] + 0.102*signals[7])

        converted_signals = [x, y, z]

        return converted_signals

    """ param = [fsamp, delta, Rp, Rs]  """
    def remove_base_line_drift(self, param, sig):
        print 'base-line'
        fsamp = float(param[0])
        delta = float(param[1])
        Rp = float(param[2])
        Rs = float(param[3])
        fcuts = [.67-delta, .67+delta]
        nhp, Wnhp = scisig.cheb2ord((2.0*fcuts[1])/fsamp,
(2.0*fcuts[0])/fsamp, Rp, Rs)
        b1, a1 = scisig.cheby2(nhp, Rs, Wnhp, 'high')
        x = scisig.lfilter(b1, a1, sig[0][::-1], axis=0)
        y = scisig.lfilter(b1, a1, sig[1][::-1], axis=0)
        z = scisig.lfilter(b1, a1, sig[2][::-1], axis=0)

        x = scisig.lfilter(b1, a1, x[::-1], axis=0)
        y = scisig.lfilter(b1, a1, y[::-1], axis=0)
        z = scisig.lfilter(b1, a1, z[::-1], axis=0)


        base_line_drift_signals = [x, y, z]

        return base_line_drift_signals

    """ param = [fsamp, delta, Rp, Rs]  """
    def remove_high_frequency_noise(self, param, sig):

        fsamp = param[0]
        delta = param[1]
        Rp = param[2]
        Rs = param[3]
        fcuts = [430-delta, 160+delta]
        nlp, Wnlp = scisig.cheb2ord((2.0*fcuts[0])/fsamp,
(2.0*fcuts[1])/fsamp, Rp, Rs)
        b2, a2 = scisig.cheby2(nlp, Rs, Wnlp, 'low')

        x = scisig.lfilter(b2, a2, sig[0][::-1], axis=0)
        y = scisig.lfilter(b2, a2, sig[1][::-1], axis=0)
        z = scisig.lfilter(b2, a2, sig[2][::-1], axis=0)

        x = scisig.lfilter(b2, a2, x[::-1], axis=0)
        y = scisig.lfilter(b2, a2, y[::-1], axis=0)
        z = scisig.lfilter(b2, a2, z[::-1], axis=0)

        remove_high_frequency_noise_signals = [x, y, z]

        return remove_high_frequency_noise_signals

    def cross_correlation(self, sig, fs=1000.0, adcgain=1000.0, adczero=0):
```

```
        peaks_indexes = wfdb.processing.gqrs_detect(x=sig[0], fs=fs,
adcgain=adcgain, adczero=adczero, threshold=1.0)
        print peaks_indexes[0]
        min_bpm = 10
        max_bpm = 350
        min_gap = fs*60/min_bpm
        max_gap = fs*60/max_bpm
        new_indexes = wfdb.processing.correct_peaks(x=sig[0],
peak_indices=peaks_indexes, min_gap=min_gap, max_gap=max_gap,
smooth_window=150)
        tamplate_x = sig[0][new_indexes[0]-500:new_indexes[0]+500]
        tamplate_y = sig[0][new_indexes[0]-500:new_indexes[0]+500]
        tamplate_z = sig[0][new_indexes[0]-500:new_indexes[0]+500]
        t1 = int(new_indexes[0]-500)
        t2 = int(new_indexes[0]+500)
        #t1 = int(200)
        #t2 = int(1199)
        x = sig[0][t1:t2]
        w = np.correlate(sig[0], tamplate_x)
        print len(w), len(sig[0])
        t = np.linspace(0, len(x)-1, len(x))
        print x.shape

        return w

    """ param = [trhesh_const]  """
    def mean_signals(self, threshold, sig, sig_cor):

        w = sig_cor
        thresh_const = float(threshold)
        index = []
        thresh = thresh_const*(w[0:5000].max())

        start = 1

        for i in range(start, len(w)-1):
            if ((w[i] >  thresh) and (w[i] >  w[i-1]) and  (w[i] >
w[i+1])):
                index.append(i)

        lint = 1000
        ave1 = np.zeros(lint)
        ave2 = np.zeros(lint)
        ave3 = np.zeros(lint)
        counter = 0
        print index
        for i in range(1, len(index)):
            k = index[i]
            if k+lint <= 1000000:
                ave1 += sig[0][k:k+lint]
                ave2 += sig[1][k:k+lint]
                ave3 += sig[2][k:k+lint]
                counter += 1

        ave1 = ave1/counter
        ave2 = ave2/counter
        ave3 = ave3/counter

        print counter
```

```
        sig1 = ave1
        sig2 = ave2
        sig3 = ave3

        mean_signals = [sig1, sig2, sig3]
        return mean_signals

    """ param = [fsamp, order, fcut1, fcut2, type_filter, fiducial_point]
"""
    def butterworth(self, param, sig):

        fsamp = float(param[0])
        order = float(param[1])
        fcut1 = float(param[2])
        fcut2 = float(param[3])
        type_filter = str(param[4])
        fiducial_point = int(param[5])

        B, A = scisig.butter(order, [fcut1/(fsamp/2), fcut2/(fsamp/2)],
type_filter)

        b1 = scisig.lfilter(B,A,sig[0][:fiducial_point:-1])
        b2 = scisig.lfilter(B,A,sig[1][:fiducial_point:-1])
        b3 = scisig.lfilter(B,A,sig[2][:fiducial_point:-1])

        b1 = b1[::-1]
        b2 = b2[::-1]
        b3 = b3[::-1]

        b11 =  scisig.lfilter(B,A,sig[0][0:fiducial_point])
        b22 =  scisig.lfilter(B,A,sig[0][0:fiducial_point])
        b33 =  scisig.lfilter(B,A,sig[0][0:fiducial_point])

        c1 = np.hstack((b11,b1))
        c2 = np.hstack((b22,b2))
        c3 = np.hstack((b33,b3))

        butterworth_signals = [c1, c2, c3]

        return butterworth_signals

    def vector_magnitude(self, sig):

        print 'VM'
        VM =
np.sqrt(np.square((sig[0]))+np.square((sig[1]))+np.square((sig[2])))

        return VM
```

APÊNDICE B – CÓDIGO DO SISTEMA DE AQUISIÇÃO

SPI_COM.py

```
####################IMPORTAÇÃO DAS BIBLIOTECAS############################
import RPi.GPIO as GPIO #Biblioteca paea utilizar a interface GPIO do
Raspberry Pi
from periphery import SPI #Biblioteca para utilizar a interface SPI do
Raspberry Pi
import time #Biblioteca para aplicar as pausas em microssegundos
import matplotlib.pyplot as plt #Biblioteca utilizada para plotar os
resultados

PIN_CS = 36 #Pino de SPI CS
PIN_START = 37 #Pino de SPI START
PIN_DRDY = 38 #Pino de SPI DRDY
PIN_RESET = 40 #Pino de RESET

############################FUNÇÕES#####################################
def adc_send_command(spi, cmd): #Função para envio dos comandos
    GPIO.output(PIN_CS, 0)
    spi.transfer([cmd])
    time.sleep(2e-6)
    GPIO.output(PIN_CS, 1)
    time.sleep(10e-6)

def adc_wreg(spi, reg, val): #Função para escrita nos registradores
    GPIO.output(PIN_CS, 0)
    spi.transfer([0x40+reg])
    spi.transfer([0])
    out = spi.transfer([val])
    time.sleep(10e-6)
    GPIO.output(PIN_CS, 1)

####################CONFIGURAÇÕES INICIAIS#############################
GPIO.setmode(GPIO.BOARD) #A referêcia utilizada para a numeração dos pinos
é a mesma que está impressa na placa
GPIO.setup(PIN_START, GPIO.OUT) #Define o pino PIN_START como pino de saída
GPIO.setup(PIN_RESET, GPIO.OUT) #Define o pino PIN_RESET como pino de saída
GPIO.setup(PIN_CS, GPIO.OUT) #Define o pino PIN_CS como pino de saída
GPIO.setup(PIN_DRDY, GPIO.IN, pull_up_down = GPIO.PUD_UP) #Define o pino
PIN DRDY como pino de entrada com um resistor de pull-up
GPIO.output(PIN_START, 0) #Inicia o pino START com o valor 0
GPIO.output(PIN_CS, 1) #Inicia o pino CS com o valor 1
GPIO.output(PIN_RESET, 0) #Reinicia o ADS1298
time.sleep(1)
GPIO.output(PIN_RESET, 1)
spi = SPI("/dev/spidev0.0", 0, 2000000) #Inicia a comunicação SPI com uma
frequência de clock de 2MHz
spi.mode = 1
time.sleep(1)

################CONFIGURAÇÃO DOS REGISTRADORES#########################
adc_send_command(spi, 0x11) #Inicia o modo de escrita
time.sleep(0.1)
# CONFIG1
adc_wreg(spi, 0x01, 0x85)
# CONFIG2
```

```
adc_wreg(spi, 0x02, 0x00)
# CONFIG3
adc_wreg(spi, 0x03, 0xDC)
# CH1SET
adc_wreg(spi, 0x05, 0x00)
# CH2SET
adc_wreg(spi, 0x06, 0x00)
# CH3SET
adc_wreg(spi, 0x07, 0x00)
# CH4SET
adc_wreg(spi, 0x08, 0x00)
# CH5SET
adc_wreg(spi, 0x09, 0x00)
# CH6SET
adc_wreg(spi, 0x0A, 0x00)
# CH7SET
adc_wreg(spi, 0x0B, 0x00)
# CH8SET
adc_wreg(spi, 0x0C, 0x00)
# RLD_SENSP
adc_wreg(spi, 0x0D, 0x06)
# RLD_SENSN
adc_wreg(spi, 0x0E, 0x02)
#WCT1
adc_wreg(spi, 0x18, 0x0A)
#WCT2
adc_wreg(spi, 0x19, 0xE3)


##########################LEITURA DOS DADOS###################################
while True:
      R = []
      m = [[], [], [], [], [], [], [], []]
      pacote = 27 * [0]
      adc_send_command(spi, 0x08) #Envia o comando START
      adc_send_command(spi, 0x10) #Envia o comndo RDATAC
      N = 0
      while N < 10000: #Repete-se o processo até ler 10000 amostras
            if GPIO.input(PIN_DRDY) == 0: #Espera o pino DRDY ir para o
nível baixo
                  #Processo de leitura de uma amostra
                  GPIO.output(PIN_CS, 0)
                  R += spi.transfer(pacote)
                  GPIO.output(PIN_CS, 1)
                  N += 1
      adc_send_command(spi, 0x0A) #Envia o comando STOP
      for i in range(0, N):
            x = 27*i
            for j in range(1, 9):
                  m[j-1].append((R[x+j*3] << 16) + (R[x+j*3+1] << 8) +
R[x+j*3+2])
                  #Conversão dos bytes lidos para os valores de cada canal
      #Plotagem dos gráficos
      f, arr = plt.subplots(4,2)
      arr[0, 0].plot(m[0])
      arr[0, 1].plot(m[1])
      arr[1, 0].plot(m[2])
      arr[1, 1].plot(m[3])
      arr[2, 0].plot(m[4])
      arr[2, 1].plot(m[5])
      arr[3, 0].plot(m[6])
```

```
        arr[3, 1].plot(m[7])
        plt.show()
time.sleep(1)
spi.close()
GPIO.cleanup()
```