UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
CAMPUS CORNÉLIO PROCÓPIO
CURSO DE ENGENHARIA ELÉTRICA

BRUNO SAMUEL LUIZ DE OLIVEIRA

ANÁLISE DE PATOLOGIAS CARDIACAS

TRABALHO DE CONCLUSÃO DE CURSO

CORNÉLIO PROCÓPIO
2018

BRUNO SAMUEL LUIZ DE OLIVEIRA

ANÁLISE DE PATOLOGIAS CARDIACAS

Trabalho de Conclusão de Curso de graduação, apresentado à disciplina Metodologia aplicada ao TCC, do curso de Engenharia Elétrica da Universidade Tecnológica Federal do Paraná – UTFPR, como requisito parcial para a obtenção do título de Bacharel.

Orientador: Prof. Dr. André Sanches Fonseca Sobrinho

CORNÉLIO PROCÓPIO
2018

Universidade Tecnológica Federal do Paraná
Campus Cornélio Procópio
Departamento Acadêmico de Elétrica
Curso de Engenharia Elétrica

FOLHA DE APROVAÇÃO

Bruno Samuel Luiz de Oliveira

Análise de patologias cardíacas

Trabalho de conclusão de curso apresentado às 13:10hs do dia 29/11/2018 como requisito parcial para a obtenção do título de Engenheiro Eletricista no programa de Graduação em Engenharia Elétrica da Universidade Tecnológica Federal do Paraná. O candidato foi arguido pela Banca Avaliadora composta pelos professores abaixo assinados. Após deliberação, a Banca Avaliadora considerou o trabalho aprovado.

_____

Prof(a). Dr(a). André Sanches Fonseca Sobrinho - Presidente (Orientador)

_____

Prof(a). Dr(a). María Eugenia Dajer - (Membro)

_____

Prof(a). Dr(a). Luis Fernando Caparroz Duarte - (Membro)

A folha de aprovação assinada encontra-se na coordenação do curso.

AGRADECIMENTOS

Agradeço primeiramente a Deus, que sempre esteve ao meu lado mesmo nos momentos em que eu me esquecia Dele, e principalmente por me conceder tal visão de projeto.

A todos os professores que nesse tempo me ajudaram muito, no meu desenvolvimento, pois sem eles não chegaria até aqui.

Agradeço ao meu orientador Prof. Dr. André Sanches Fonseca Sobrinho, pela sabedoria com que me guiou nesta trajetória.

Gostaria de deixar registrado também, o meu reconhecimento à minha família, pois acredito que sem o apoio deles seria muito difícil vencer esse desafio.

Enfim, a todos os que por algum motivo contribuíram para a realização desta pesquisa.

Dedico este trabalho totalmente ao meu querido e amado avô Samuel Nunes Pinheiro.

RESUMO

LUIZ DE OLIVEIRA, Bruno Samuel. **ANÁLISE DE PATOLOGIAS CARDÍACAS** 2018. 102 f. Trabalho de Conclusão de Curso (Graduação) – Engenharia Elétrica. Universidade Tecnológica Federal do Paraná. Cornélio Procópio, 2018.

O trabalho apresenta o desenvolvimento de um f*irmware* embarcado em um sistema eletrônico, sendo este capaz de realizar a análise de sinais cardíacos e enviar um alerta de patologias detectadas nos sinais analisados via Servidor Web. Como diferencial, usa o sistema embarcado *Raspberry Pi,* versão 3, para a realização dos testes e desenvolvimento do f*irmware* proposto. Descrevendo o processo necessário para analisar e detectar patologias cardíacas para que se possa obter a filtragem do sinal cardíaco com posterior detecção dos picos R para o cálculo da frequência cardíaca e classificação de patologias cardíacas. Com intuito de se averiguar a capacidade de operação do *firmware*, arquivos com amostras de sinais contendo alguma patologia são aplicados no firmware proposto. Logo, é possível a verificação da implementação dos tratamentos matemáticos em sinais cardíacos, a partir da linguagem Python de programação, viabilizando assim o desenvolvimento de um f*irmware* embarcado em um sistema eletrônico, apresentando uma nova metodologia de aquisição e processamento de sinais eletrocardiográficos, visando detectar patologias cardíacas e o envio de um alerta de patologias detectadas nos sinais analisados via Servidor Web.

.

Palavras-chave: Monitor cardíaco. Servidor Web. Sistema Embarcado.

ABSTRACT

LUIZ DE OLIVEIRA, Bruno Samuel. **ANALYSIS OF CARDIAC PATHOLOGIES** 2018. 102 f. Trabalho de Conclusão de Curso (Graduação) – Engenharia Elétrica, Universidade Tecnológica Federal do Paraná, Cornélio Procópio, 2018.

This work shows the development of a firmware in an embedded system capable of the analysis of cardiac electrical signals and shows via web server, an alert of cardiac pathologists diagnosed in the analyzed signals. As a differential, embedded system the Rasspberry Pi version 3 will be used for the tests and development of the proposed firmware. Describing the process necessary for the analysis and diagnosis of cardiac pathologies in order to obtain the filtering of the cardiac signal with subsequent detection of the R peaks, calculation of the heart rate and classification of cardiac pathologies. In order to investigate the processing capacity of the firmware, files with samples of signals containing some pathology were applied in the proposed firmware. Therefore, it is possible to verify the implementation of the mathematical treatments in cardiac signals, from the Python programming language, thus enabling the development of a firmware embedded in an electronic system, presenting a new methodology for the acquisition and processing of electrocardiographic signals, aiming at the diagnosis of cardiac pathologies and the sending of an alert of pathologies diagnosed in the analyzed signals through Web Server.

Keywords: Cardiac monitor. Web server. Embedded System.

SUMÁRIO

1 INTRODUÇÃO

O trabalho apresenta o desenvolvimento de um f*irmware* embarcado em um sistema eletrônico, sendo este capaz de realizar a análise de sinais cardíacos e enviar um alerta de patologias detectadas nos sinais analisados via Servidor Web. Com isto, tem-se a visão de um *firmware* embarcado em um sistema eletrônico que possa ser utilizado como um acessório, onde o cardiologista pode ser alertado remotamente na ocorrência de mudanças drásticas no quadro do paciente.

Este trabalho estabelece um estudo sobre análise de sinais elétricos cardíacos, feito com base no banco de dados disponibilizados nas teses e dissertações de mestrado e doutorado do professor e orientador André Sanches Fonseca Sobrinho, de titulo Projeto de um sistema para aquisição e analíse de eletrocardiogramas e tendo como principal objetivo a produção de um firmware voltado a sistemas embarcados para estudo de sinais elétricos cardíacos cujo objeto de estudo ou tema central são os sinais elétricos cardíacos. A contribuição deste trabalho consiste em estabelecer uma base informacional, apresentar uma análise quantitativa dos dados oferecidos e no desenvolvimento do firmware embarcado em um sistema eletrônico capaz de realizar a análise de sinais elétricos cardíacos e disponibilizar, via servidor *Web,* um alerta sobre possíveis patologias encontradas nos sinais analisados.

O desenvolvimento do projeto se faz necessário, pois indicará patologias presentes em sinais de eletrocardiograma, prevendo benefícios principalmente a idosos, pessoas com dificuldades de locomoção, pacientes em tratamento e outros casos. O trabalho está organizado da seguinte maneira:

- O segundo capítulo descreve a origem dos sinais bioelétricos, a atividade cardíaca, o eletrocardiograma e as patologias cardíacas;
- O terceiro capítulo apresenta o sistema embarcado, suas características, como também suas linguagens e bibliotecas de programação;
- O quarto capítulo descreve detalhadamente os métodos e matérias utilizados para desenvolvimento deste projeto;
- O quinto capítulo descreve os testes realizados na biblioteca de filtragem e no *firmware*;

- No sexto capítulo são apresentadas conclusões, problemas encontrados no desenvolvimento e possibilidades de expansão do projeto em trabalhos futuros.

1.1 Delimitação do tema

Este trabalho delimitou-se em colher informações sobre os sinais elétricos cardíacos, como também apresentar o processo necessário para a análise de patologias cardíacas, visando o desenvolvimento de Firmware embarcado, capaz de realizar a análise de sinais elétricos cardíacos, como também a filtragem do sinal para eliminação de impurezas não desejadas, tratamento matemáticos no sinal, visando o cálculo da frequência cardíaca, classificação da patologia e disponibilizar, via servidor Web, um alerta sobre possíveis patologias encontradas nos sinais analisados, para isto são utilizados matérias e métodos apresentados no decorrer do trabalho.

1.2 Problemas e premissas

A falta de monitoramento constante em pacientes que apresentem evidencia de problemas cardíacos e a necessidade de detectar de forma rápida em casos que necessitem de intervenção imediata, como também a não existem de sistemas que alertem os familiares e médicos de forma remota, sobre a condição do paciente e qual patologia esta sendo detectada, então inicia-se a busca por alternativas que possa realizar tal monitoramento e alerta.

2 Objetivos

2.1 Objetivo geral

Modelar o *firmware* capaz de realizar a análise de um eletrocardiograma e a identificar as seguintes patologias cardíacas, Parada cardíaca, Bradicardia Sinusal, Taquicardia Sinusal, Flütter Atrial e Fibrilação Atrial.

2.2 Objetivos específicos

- Utilizar arquivos com amostras de sinais cardíacos no *firmware,* para testes de funcionalidade, tratamentos matemáticos e cálculo da frequência cardíaca;

- Identificar patologias cardíacas utilizando o cálculo da frequência cardíaca e as identificações morfológicas;

- Disponibilizar alertas de possíveis patologias presentes nos sinais analisados através de um servidor *Web*.

3 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo são abordados os fundamentos teóricos dos sinais elétricos biológicos, métodos para o tratamento destes sinais e formas para transmitir os mesmos para plataformas computacionais que possam fazer análise das patologias médicas presentes nos sinais cardíacos.

3.1 Sinais biológicos

O registro dos sinais bioelétricos é de extrema importância na medicina moderna, pois todos os seres vivos multicelulares ou unicelulares geram sinais de origem biológica. Os sinais elétricos gerados pelos seres vivos são chamados sinais bioelétricos, sendo que as unidades geradoras dos sinais são as células nervosas e musculares. Os efeitos acumulados de todas estas células ativadas produzem um campo elétrico que se propaga pelo corpo. A atividade de alguma rede neural ou de um músculo pode ser medida através de eletrodos posicionados sobre a superfície da pele. [1]

Sinais elétricos biológicos são provenientes das atividades celulares dos músculos. O sinal de eletrocardiograma revela o estado das unidades motoras durante a contração muscular cardíaca. A atividade dos músculos gera sinais que apresentam amplitudes na escala de milivolts, os quais se propagam por meio da pele e podem ser medidos e registrados utilizando eletrodos. [2]

3.2 Anatomia e função do coração

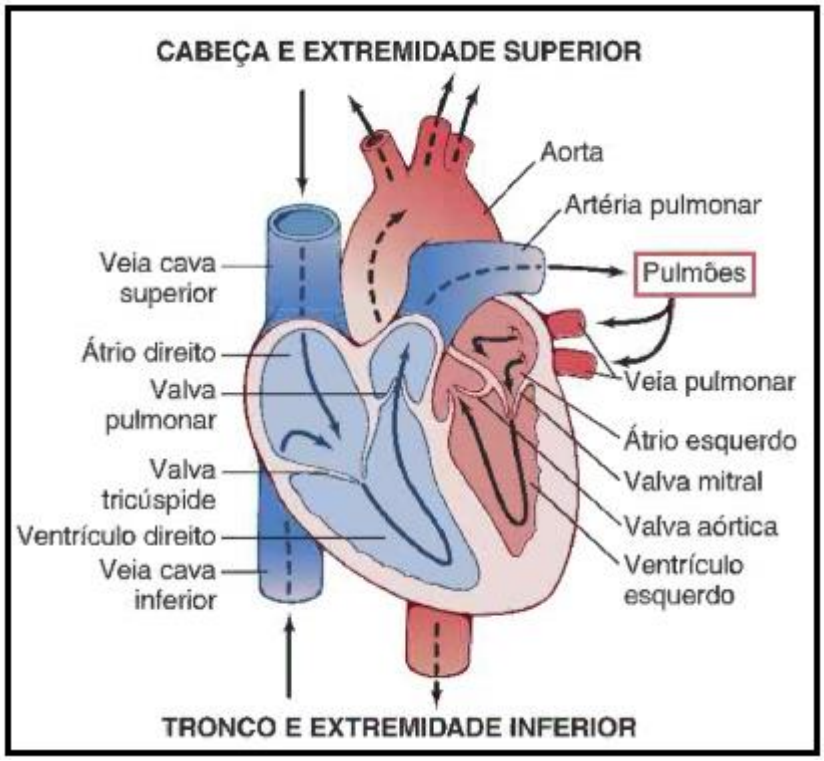
O coração é um órgão muscular oco que se localiza no meio do peito, um pouco deslocado para a esquerda. A parte musculosa do coração é denominada de músculo cardíaco ou miocárdio. O coração possui cavidades, na parte superior estão os átrios, divididos entre o direito e o esquerdo e na parte inferior estão os ventrículos, também direito e esquerdo. O coração está constantemente a relaxar e contrair. [3]

Na metade esquerda circula somente sangue arterial (rico em oxigênio) e na metade direita do coração, só circula sangue venoso (pobre em oxigênio). Seu

trabalho equivale a duas bombas trabalhando juntas. Uma das bombas engloba a aurícula e o ventrículo direitos e a outra a aurícula e o ventrículo esquerdos. [3]

A função do átrio e do ventrículo direitos é mover o sangue para os pulmões, onde se libera o dióxido de carbono e se fornece de oxigênio. O átrio e o ventrículo esquerdos têm o trabalho de mover o sangue enriquecido de oxigênio para todas as partes do corpo. A Figura 1 retrata a imagem da anatomia do coração e tem o intuito de demonstrar as fibras musculares cardíacas. [3]

Figura 1 – Anatomia do coração



Fonte: Tratado de fisiologia médica. (cap.9, p. 107)

3.3 Eletrocardiograma

Os registros dos sinais cardíacos, ou seja, o eletrocardiograma nada mais é do que uma gravação da atividade elétrica do coração e é através de observações de perturbações nos padrões elétricos normais que pode-se detectar diferentes distúrbios cardíacos. [6]

O holandês Willem Einthoven foi um médico e fisiologista conhecido por projetar o galvanômetro de cordas, o qual detectava e registrava traços visuais da

atividade elétrica do coração. Durante o século 20, esta técnica, diagnosticou condições cardíacas. [7]

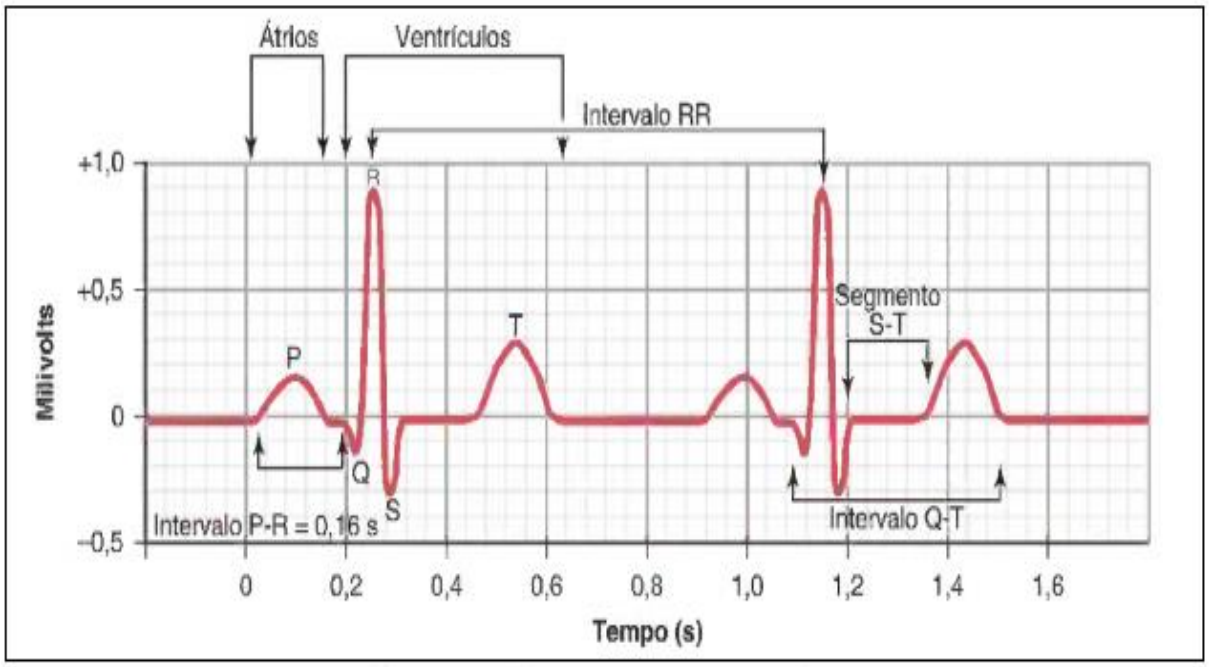
O primeiro galvanômetro de cordas era um dispositivo complexo e grande que necessitava de cinco pessoas para operar. A "corda" era um cristal de quartzo extremamente fino, capaz de conduzir eletricidade. Os impulsos elétricos das contrações cardíacas eram transmitidos à corda por meio de eletrodos. [7]

3.3.1 Características do eletrocardiograma

Um eletrocardiograma normal é composto pela onda P, pelo complexo QRS e pela onda T. Algumas características comuns a eles são: Frequência normal de 60 a 100 batimentos por minuto, espectro de frequência de 0,01 a 200hz e faixa de amplitude do sinal de 0,2mV a 4mV (a amplitude pode ser negativa de acordo com a derivação escolhida para a aquisição do eletrocardiograma). [8]

A Figura 2 retrata a imagem de um eletrocardiograma normal, com seus respectivos intervalos.

Figura 2 – Eletrocardiograma normal



Fonte: Gyton e Hall (Cap.11, p. 129).

A frequência cardíaca é inversamente proporcional ao ciclo cardíaco, ou seja, quando a frequência cardíaca aumenta, a duração de cada ciclo diminui, incluindo as fases de dilatação e relaxamento. A frequência normal é de setenta e dois batimentos por minuto. [3] Normalmente as ondas R são usadas para se calcular à frequência cardíaca, medindo a distancia entre os picos se tem o periodo de cada ciclo.

3.2.1.1 Onda P

A produção da onda P ocorre devido aos potencias elétricos gerados quando os átrios se despolarizam, antes da ocorrência da contração atrial. [9]

Características da onda P:

- Amplitude da tensão máxima está entre 0,25mV a 0,3mV;
- A duração do intervalo varia conforme a idade, em casos normais o tempo é de 0,09s em crianças e 0,10s em adultos;
- Tem morfologia arredondada, podendo apresentar pequenos entalhes desde que não ultrapasse 0,03s. [10]

3.2.1.2 Intervalo P-Q

O intervalo entre o começo do estímulo elétrico dos átrios e o começo dos estímulos dos ventrículos corresponde ao tempo entre início da onda P e o início do complexo. Este intervalo também pode ser chamado de P-R, porque é comum a onda Q não estar presente. [9].

Características da onda P-Q:

- Amplitude da tensão variável conforme a derivação utilizada;
- A duração do intervalo vária conforme a idade, em casos normais o tempo é 0,12s a 0,20s;
- A morfologia do sinal é variável conforme a derivação utilizada. [10].

3.2.1.3 Complexo QRS

O complexo é produzido pelos potências gerados quando os ventrículos se despolarizam antes de uma contração. Tanto a onda P como as componentes do complexo QRS são ondas de despolarização. [9].

Características da onda QRS:

- Amplitude da tensão variável conforme a derivação utilizada;
- A duração do intervalo varia entre os valores de 0,05s a 0,10s em casos normais;
- A morfologia do sinal vária conforme a derivação utilizada. [10].

3.2.1.4 Intervalo ST

Sua duração normalmente não é determinada, pois é avaliado quando englobado ao intervalo QT [10].

3.2.1.5 Onda T

A onda T é conhecida como onda de repolarização, quando os ventrículos se restabelecem do estado de despolarização. Tal processo no músculo ventricular, geralmente ocorre entre 0,25s a 0,35s após a sua despolarização [9].

Características da onda T:

- A morfologia da onda T normal é assimétrica;
- Amplitude geralmente é menor que o QRS, porém não há critérios [10].

3.2.1.6 Onda U

A onda U não é constante no sinal do eletrocardiograma, mas é possível aparecer. Quando aparece possui valores positivos que são considerados normais e quando é negativo indica sinal de patologia. A onda U segue após a onda T [10].

3.2.1.7 Intervalo QT

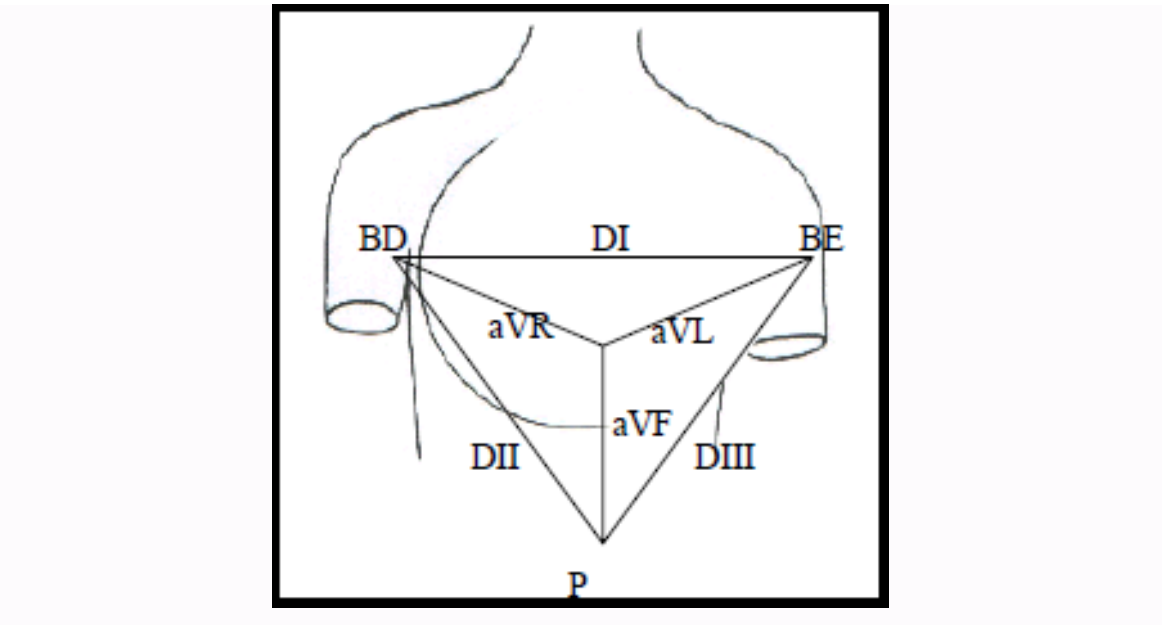
Este intervalo possui em média duração de cerca de 0,35s. A contração do ventrículo dura aproximadamente do inicio da onda Q, ou onda R quando a onda Q é inexistente, até o final da onda T. [10]

3.2.2 Derivações eletrocardiográficas

De acordo com a posição dos eletrodos na superfície do corpo é possível aferir diferenças de potencial consequente dos fenômenos elétricos gerados durante a excitação muscular. Tais valores podem ser medidos e registrados. Desta forma, pontos específicos da superfície da pele são ligados ao eletrocardiograma por meio de eletrodos, onde tais pontos são chamados de derivações. O eletrodo atua como um transdutor, convertendo uma corrente iônica em uma corrente elétrica. [11]

Assim, o conceito de Triângulo de Einthoven diz que a união das extremidades (braços e perna) forma um triângulo equilátero, e que o coração ocupa o centro do triângulo. Sendo assim, AVR, AVL e AVF são potenciais absolutos do braço direito (BD), braço esquerdo (BE) e pernas (P) em relação ao coração, que está no centro do triângulo, desta forma é possível obter as derivações do plano frontal, conforme a Figura 3 demostra [10]

Figura 3 – Derivações do plano frontal



Fonte: CARNEIRO, E. F.

Onde:

- DI é a diferença de potencial entre BE e BD;
- DII é a diferença de potencial entre P e BD;
- DIII é a diferença de potencial entre P e BD;

- AVR é a diferença de potencial entre BD e o coração (potencial de referência);
- AVL é a diferença de potencial entre BE e o coração (potencial de referência);
- AVF é a diferença de potencial entre P e o coração (potencial de referência). [10]

Existem diferentes tipos de eletrodos, como os de placa metálica, os eletrodos descartáveis e os de sucção. Os cabos dos eletrodos do eletrocardiograma devem ser blindados e aterrados juntos. Entre o contato da pele e o eletrodo utiliza-se um gel eletrolítico transparente para melhorar o contato [11]

3.2.3 Métodos para registros de sinais elétricos cardíacos

As correntes geradas pelo músculo cardíaco podem alterar suas polaridades em menos de 0,01 segundos, durante um batimento do coração. Sendo assim, é necessário que se leve em consideração essas mudanças na projeção de equipamentos para registro dos sinais cardíacos [9]

3.3 Ciclo cardíaco e frequência cardíaca

O conjunto de eventos cardíacos é demoninado de ciclo cardíaco, onde este ocorre entre o início de um batimento e o início do próximo. Cada ciclo é iniciado pela geração espontânea de potencial de ação no nodo sinusal. [3]

A frequência cardíaca pode ser encontrada a partir da detecção dos picos R, logo um sistema de detecção pode ser aplicado direto ao sinal de cardíaco, não necessitando da utilização de filtros, já que o complexo QRS tem maior amplitude que as demais ondas. Porém, nem sempre isto ocorre já que em algumas situações a onda R aparece invertida, ou com menor intensidade que a onda T, sendo necessário eliminar essa componente de baixa frequência. Assim, a aplicação do filtro passa-faixa é de extrema importância, fazendo com que essas características sejam eliminadas do sinal e garantindo a leitura exata entre os picos R. [4]

Para análise do sinal cardíaco e extração da frequência cardíaca é necessário que o sinal esteja em sua forma mais pura, ou seja, sem a interferência de outros sinais que distorcem o sinal cardíaco original. Para isso é essencial utilização de filtros para que apenas as frequências desejadas estejam no sinal a ser analisado. A maior parte da energia do complexo QRS está concentrada na faixa de 5 a 15hz, e considerando apenas o complexo QRS como desejado, tem-se que a máxima relação sinal ruído é obtida em 17Hz. [5]

3.4 Patologias cardíacas

Por meio da análise do eletrocardiograma, diversas anormalidades podem ser detectadas, após cálculo da frequência cardíaca. Algumas destas patologias são citadas a seguir:

- Bradicardia Sinusal: frequência cardíaca abaixo de 60 RPM;

- Taquicardia Sinusal: frequência cardíaca acima de 100 BPM;
- Fibrilação Atrial: frequência cardíaca variando entre 360 e 600 BPM;
- Plotter Atrial: frequência cardíaca variando entre 240 e 350 BPM;
- Arritmia Sinusal ou Taquicardia Atrial Multifocal: os intervalos entre os picos R são variáveis.
- Parada cardíaca: uma anormalidade grave do sistema cardíaco, resultante do cancelamento de todos os sinais cardíacos [8].

3.5 Anomalias do eletrocardiograma

Existem padrões anormais encontrados em pacientes, padrões considerados diferentes do normal, onde suas interpretações acabam por intervir na avaliação do ritmo cardíaco. Padrões como, por exemplo: o prolongamento do complexo QRS, onde com isto se resulta na hipertrofia ou na dilatação cardíaca, e o prolongamento do complexo QRS decorrente do bloqueio do sistema de Purkinje [12].

O complexo QRS é considerado anormalmente longo quando sua duração é maior do que 0,09 segundos e quando dura mais que 0,12 são causados quase que certamente por bloqueio patológico em algum ponto do sistema de condução ventricular [12].

Patrões anormais, na maioria das vezes, são causados por duas condições, sendo a primeira destruição da musculatura cardíaca do sistema ventricular e a segunda causa por pequenos múltiplos bloqueios e locais da condução do impulso em vários pontos do sistema de Purkinge. Esses padrões como resultado passam a conduzir o impulso cardíaco de forma irregular, causando rápidas inversões das tensões e desvios de eixo. Geralmente causa picos duplos ou triplos em algumas das derivações eletrocardiográficas. [12]

3.6 Sistemas embarcados

O intuito desse trabalho é implementar o firmware proposto em um sistema embarcado que possa ser facilmente programado para que realize a interface de um servidor *Web* de forma rápida e simplificada, como também um hardware capaz de processar grandes quantidades de dados de entrada, visando o baixo orçamento.

Assim escolheu-se com auxílio da tabela contida no anexo A disponibilizada no portal Love PI, onde tal contém as principais características dos sistemas embarcados mais difundindo no mercado, o terceiro modelo de microcomputador desenvolvido pela Fundação *Raspberry Pi,* denominado *Raspberry Pi 3*, onde tal modelo é de fácil aquisição no mercado nacional e também apresenta um bom desempenho no desenvolvimento e nos testes do projeto. [13]

Raspberry Pi, diferente de outros sistemas embarcados menos difundidos no mercado, possui um o blog oficial para notícias e atualizações da Fundação *Raspberry Pi*, como também desenvolve iniciativas educacionais, projetos comunitários, com vasto material disponível e vários exemplos de aplicações, em portais na *Web*. [14]

No *Raspberry Pi,* é possível realizar a instalação de sistemas operacionais de distribuições *Linux* ou o sistema operacional *Windows 10 IoT*. O sistema *Raspbian* é o sistema operacional suportado oficialmente pela Fundação Raspbian [16].

O Raspbian disponibiliza pacotes de softwares para educação, programação e uso geral, como por exemplo: Python, Scratch, Sonic Pi, Java, Matemática e outros. O Raspbian é um sistema operacional gratuito baseado no Debian, otimizado para o hardware Raspberry Pi. O Raspbian possui mais de 35.000 pacotes, software pré-compilado, é um projeto comunitário em desenvolvimento ativo [16].

.

4 DESENVOLVIMENTO

O foco do trabalho é modelar o *firmware* capaz de realizar a análise de um eletrocardiograma e disponibilizar via servidor Web um alerta de patologias detectadas nos sinais analisados. Desta forma, não será desenvolvida a solução de hardware responsável pela aquisição de sinais cardíacos e comunicação com o sistema embarcado.

Logo, este capítulo tem por objetivo apresentar o processo necessário para a análise e detecção de patologias cardíacas, onde serão mostradas os matérias e métodos para que se possa obter a filtragem do sinal cardíaco, com posterior detecção dos picos R, cálculo da frequência cardíaca e classificação da patologia.

4.1 Materiais

Com intuito de realizar o projeto foram adquiridos os seguintes equipamentos e componentes:

- Sistema embarcado *Raspberry Pi 3*, utilizado como servidor e processador de sinais;
- Fonte de alimentação CC 5 V (ou *power bank USB*), utilizado para suprir a alimentação do sistema embarcado;
- Mouse e teclado *USB*, onde tais são utilizados para controle e configurações do sistema embarcado;
- Monitor com entrada HDMI, onde tal tem a função de mostrar à interface gráfica do sistema embarcado;
- Roteador WiFi, para a comunicação do *Raspberry PI* com a rede local;
- Computador pessoal, para comunicação com a rede local e para visualização do servidor de alertas em funcionamento;
- Banco de dados contendo amostras de sinais cardíacos, com presença de patologias cardíacas, disponibilizado pelo professor e orientador André Sanches Fonseca Sobrinho, para serem analisados e auxiliar no desenvolvimento do Firmware.

4.1.1 Sistema embarcado *Raspberry Pi 3*

O *Raspberry Pi* é um sistema embarcado de baixo custo, que tem o tamanho de um cartão de crédito, desenvolvido no Reino Unido inicialmente pela Fundação *Raspberry Pi* e lançado em 2012. Tal projeto visa colocar o poder da tecnologia de sistemas embarcados nas mãos de pessoas de todo o mundo, para que elas sejam capazes de entender e dar forma ao mundo digital. [14]

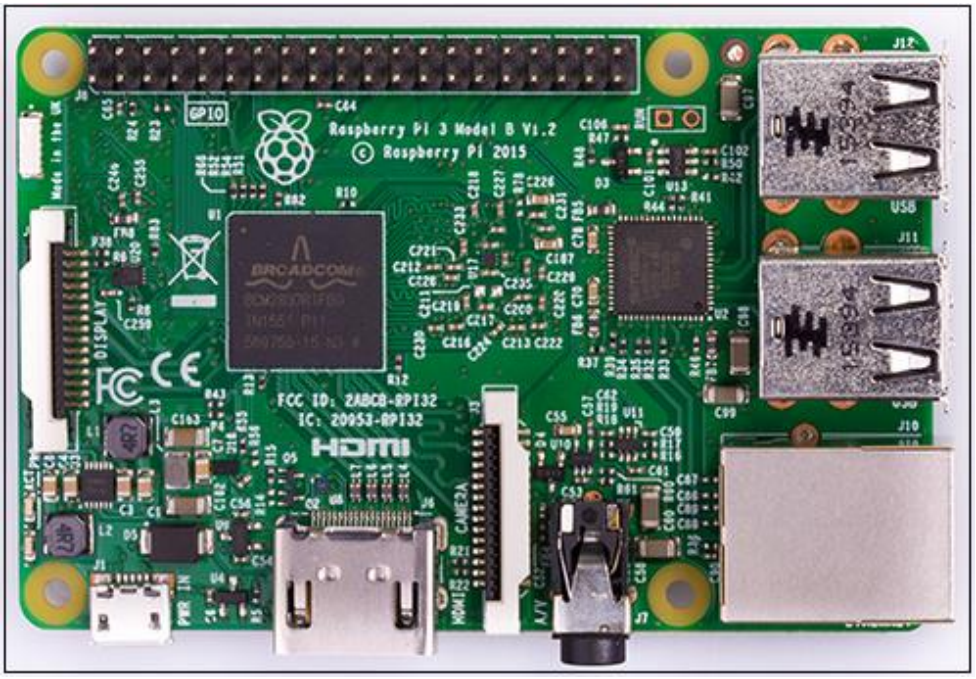
Ele também pode agir como um servidor internet. Na maioria das vezes, servidores como esses estão enviando arquivos *HTML* e imagens para fazer páginas da web.

Características do sistema embarcado *Raspberry Pi* 3:

- Uma *CPU ARMv8 quad-core de 64 bits de 1.2 GHz;*
- *Lan* sem fio 802.11n;
- *Bluetooth;*
- *1GB* de *RAM*;
- 4 Portas *USB*;
- 40 Pinos *GPIO*;
- Saida de video HDMI;
- Conexão de rede;
- Conector de áudio combinada de 3,5 mm e vídeo composto;
- Interface da câmera (CSI);
- Interface de exibição (DSI);
- Ranhura do cartão Micro SD (agora empurrar-puxar em vez de push-push);
- Núcleo de gráficos 3D VideoCore IV. [15].

A Figura 4, foi retirada do portal *Raspberry Pi*, onde tal figura retrata o sistema embarcado para a simulação do *firmware* a ser implementado [15].

Figura 4 – Sistema embarcado Raspberry Pi 3



Fonte: Retirada do portal Raspberry Pi. (2017)

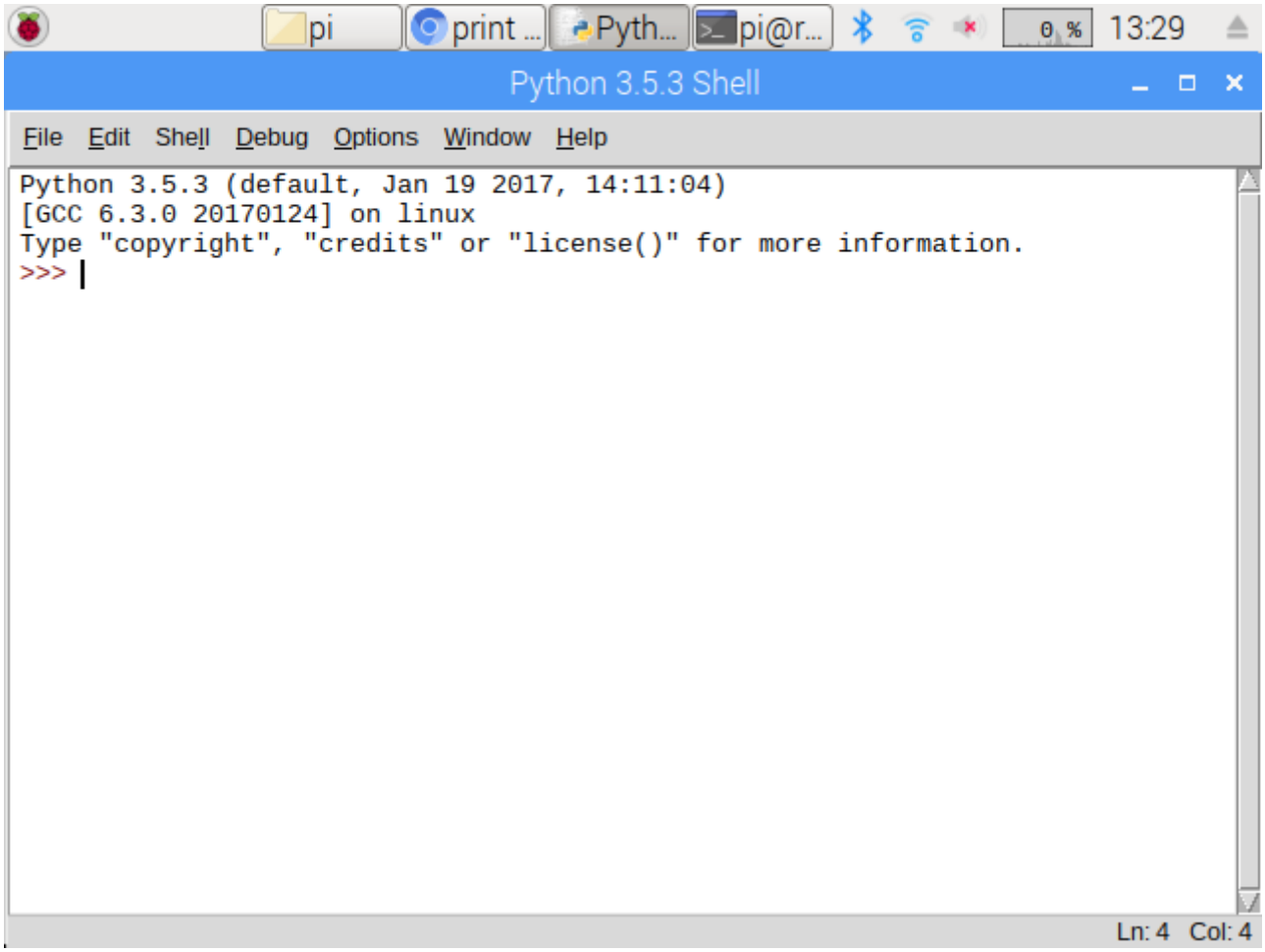
Partindo das linguagens suportadas oficialmente, foi escolhida para este projeto a linguagem *Python*, sendo esta muito rica em bibliotecas que facilitam o processamento de grandes quantidades de dados, implementações de filtros digitais para cálculo da frequência cardíaca e cria de forma fácil um servidor Web. Vale lembrar que outra linguagem pode ser utilizada na plataforma.

Python é um produto *da Python Software Foundation*, uma organização sem fins lucrativos que detém os direitos autorais do Python, que é uma linguagem de programação de alto nível. Sua sintaxe elegante de natureza interpretada, torna *Python* ideal para *scripting* e para o desenvolvimento rápido de aplicações em diversas áreas e na maioria dos sistemas e plataformas. [17].

O Python e sua extensa biblioteca padrão estão disponíveis na forma de código fonte ou binário para a maioria dos sistemas operacionais e deve ser distribuído livremente. [17].

Similar ao Matlab, o Python é dividido em duas janelas, sendo a janela Shell, Figura 5, responsável por mostrar o estado da execução do módulo (Código fonte), tanto de um script completo quanto linha por linha.

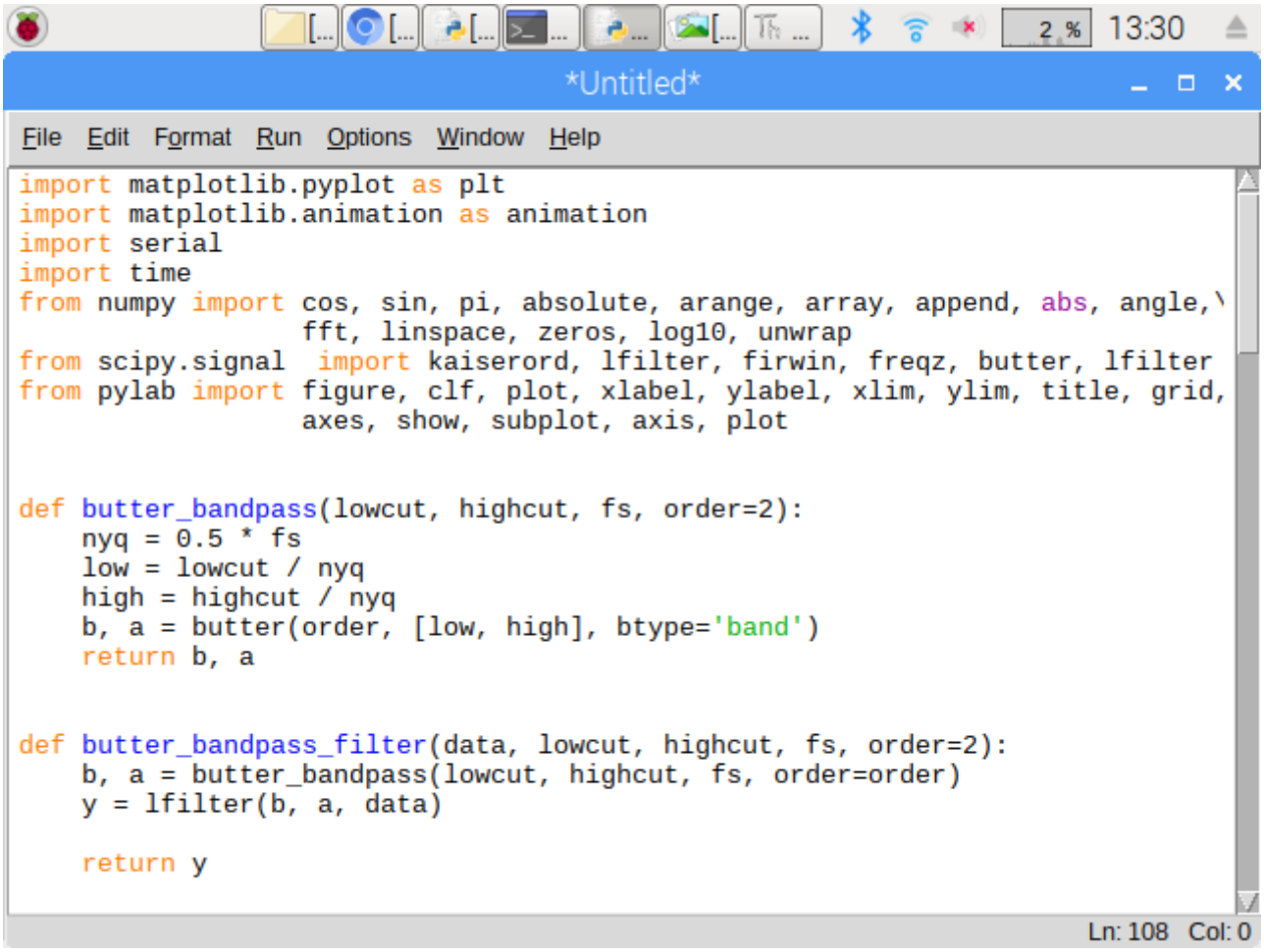
Figura 5 – *Shell Python*



Fonte: Autoria Própria.

A Figura 6 demonstra a janela *IDLE* do *Python,* onde nesta é possível à escrita por completo do código fonte.

Figura 6 – *Idle python 3.5.3*



Fonte: Autoria Própria.

4.1.2 Bibliotecas Python

Na Web existe uma ampla gama de aplicações e extensões *Python*, muitas dessas são supridas por portais de comunidades de desenvolvimento brasileiras, uma das maiores vantagens da linguagem são as suas bibliotecas embutidas, que oferecem diversas facilidades ao resolver problemas diários enquanto desenvolvemos uma aplicação. [18].

4.1.2.1 Biblioteca web

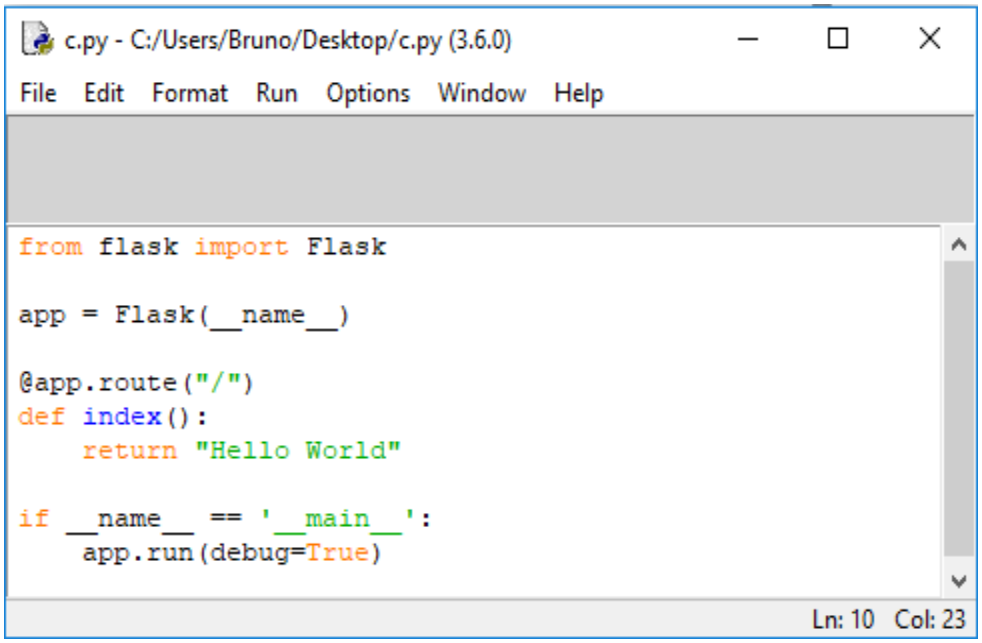
A linguagem *Python* possui uma biblioteca *Web*, chamada *Flask*, que transforma o *Raspberry Pi* em um servidor web dinâmico, onde tal servidor será

responsável por dar a respostas de patologias presentes no sinal cardíaco analisado.

Além da fácil instalação e uso, há também uma grande quantidade de tutoriais detalhados, que mostram como criar uma aplicação *Web Flask* completa. Tal extensão depende de duas bibliotecas externas: o mecanismo de modelo *Jinja2* e o conjunto de ferramentas *Werkzeug WSGI.* [19]

Flask pode ser usado para criação de sites e projetos utilizando *Python*, podendo ser implementado de várias formas, porém a estrutura mínima para uma aplicação em Flask, pode ser montada conforme o código a presentado na Figura 7.

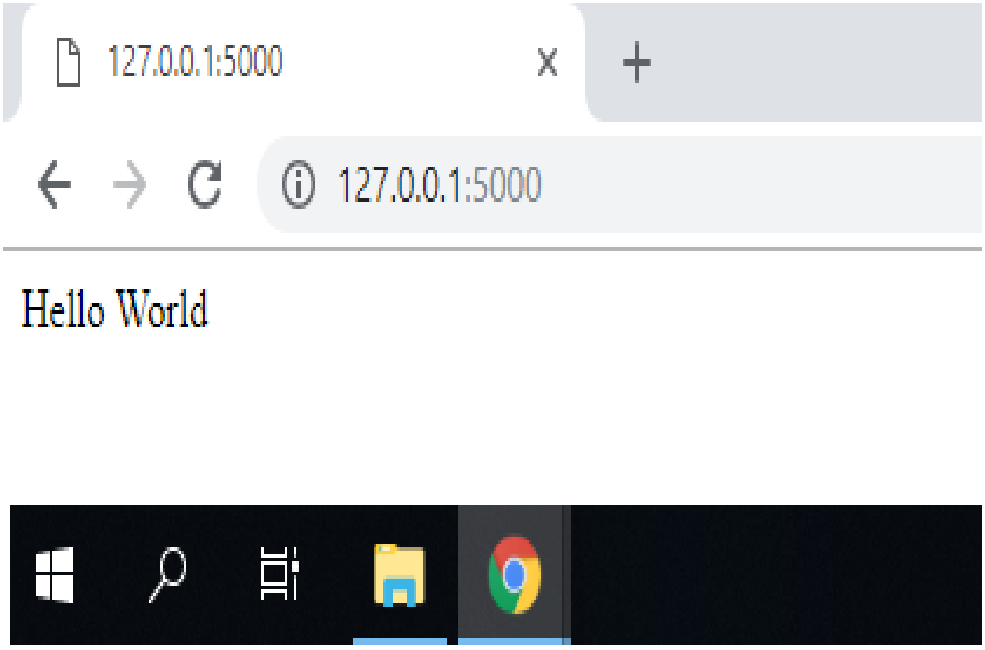
Figura 7 - Estrutura mínima para uma aplicação em Flask



Fonte: Autoria própria.

Para acessar a aplicação criada deve-se acessar o endereço: < http://localhost:5000> de um navegador Web, para visualizar o site em operação como qualquer outro, conforme a Figura 8 demostra. A página também pode ser aprimorada, mas para efeito didático se optou por utiliza-lo da forma mais simplificada.

Figura 8 - Site de exemplo gerado pela aplicação *Flask*



Fonte: Autoria própria.

4.1.2.2 Biblioteca matemática SciPy

SciPy é um ecossistema baseado em Python de software de código aberto para matemática, ciência e engenharia, baseado na Numpy, Matplotlib, Pandas e SciPy library.

Entretanto a biblioteca SciPy library é o principal pacote que compõem a pilha SciPy. Ela fornece muitas rotinas numéricas amigáveis e eficientes, como rotinas para integração numérica e otimização. [20]

4.1.2.3 Biblioteca Numpy

NumPy é o pacote fundamental para a computação científica com o Python. Contém um poderoso objeto de matriz N-dimensional, funções sofisticadas (transmissão), ferramentas para integrar C / C ++, código Fortran, álgebra linear, transformada de Fourier e capacidades de gerar números aleatórios. [21]

Além de seus usos científicos, o NumPy também pode ser usado como um eficiente recipiente multidimensional de dados genéricos. Tipos de dados arbitrários podem ser definidos. Isso permite que o NumPy se integre de forma simples e rápida com uma grande variedade de bancos de dado entre outras coisas.

NumPy é licenciado sob a licença BSD, permitindo a reutilização com poucas restrições. [21]

4.1.2.4 Biblioteca Matplotlib

Matplotlib é uma biblioteca de traçado gráfico em Python de duas dimensões, que produz números de qualidade de publicação em uma variedade de formatos impressos e ambientes interativos em plataformas. O Matplotlib pode ser usado em scripts Python, o shell Python e Ipython. [22]

A licença Matplotlib é baseada na licença Python Software Foundation (PSF). Onde, com apenas algumas linhas de código, pode-se gerar gráficos, histogramas, gráficos de barras, gráficos de erros, diagramas de dispersão e outros. Através de uma interface orientada a objetos ou através de um conjunto de funções familiares aos usuários do MATLAB. [22]

O Matplotlib foi desenvolvido por John Hunter, que, juntamente com seus colaboradores, colocou uma quantidade de tempo e esforço para desenvolvimento deste. Como tal biblioteca contribui para o desenvolvimento deste projeto, pois sem ela não seriam possíveis as analises gráficas apresentadas, o desenvolvedor do matplotlib pede a gentileza de reconhecer esse fato citando-o, usando uma entrada em BibTeX pré-fabricada, conforme apresentada pela Figura 9.

Figura 9 – Entrada em BibTeX pré-fabricada

```
@Article{Hunter:2007,
  Author    = {Hunter, J. D.},
  Title     = {Matplotlib: A 2D graphics environment},
  Journal   = {Computing In Science \& Engineering},
  Volume    = {9},
  Number    = {3},
  Pages     = {90--95},
  abstract  = {Matplotlib is a 2D graphics package used for Python
  for application development, interactive scripting, and
  publication-quality image generation across user
  interfaces and operating systems.},
  publisher = {IEEE COMPUTER SOC},
  doi = {10.1109/MCSE.2007.55},
  year      = 2007
}
```

Fonte: PORTAL Matplotlib. Disponível em < https://matplotlib.org/citing.html>. Acesso em: 09 julho 2017.

4.1.2.5 Biblioteca pandas

O *Pandas* é uma biblioteca licenciada com código aberto que oferece estruturas de dados de alto desempenho e ferramentas de análise de dados para a linguagem de programação *Python* [23].

Python com *Pandas* é utilizado em uma ampla variedade de domínios acadêmicos e comerciais, incluindo Finanças, Neurociências, Economia, Estatística, Publicidade, Web Analíticas e *também* ajuda a preencher lacunas de programação, permitindo realizar todo o fluxo de trabalho de análise de dados em *Python* sem ter que mudar para um idioma mais específico do domínio como R [23].

Combinado com o excelente conjunto de ferramentas *IPython* e outras bibliotecas, o ambiente para fazer análise de dados no Python se destaca em desempenho, produtividade e capacidade de colaboração. Algumas características importantes de se destacar desta biblioteca estão apresentadas a seguir:

- Tem um objeto DataFrame rápido e eficiente para manipulação de dados com indexação integrada;
- Ferramentas para leitura e escrita de dados entre estruturas de diferentes formatos: arquivos de texto, CSV, Microsoft Excel, bancos de dados SQL e o formato HDF5 rápido;

- Alinhamento inteligente de dados e manuseio integrado de dados faltantes**,** ganho de alinhamento automático;
- Manipulação fácil de dados desordenados em uma forma ordenada;
- Remodelação flexível e articulação de conjuntos de dados;
- As colunas podem ser inseridas e excluídas das estruturas de dados para a mutabilidade do tamanho;
- Funcionalidade de séries temporais: geração de data e geração de frequência, estatísticas de janela em movimento, regressões lineares de janela em movimento, mudança de data e atraso [23].

4.1.3 Circuito para aquisição do sinal cardíaco

O *Raspberry PI*, não possui um circuito elétrico capaz de realizar aquisição de sinais cardíacos, pois, este não possui entradas analógicas e muito menos sistemas de amplificação de sinais.

Na *Web* e no mercado se encontram inúmeras soluções para tal aplicação, sendo uma destas possíveis soluções a utilização de um circuito integrado que realize tal função. As comunicações entre tais circuitos podem ser feitas via interfaces *input/output*, interface periférica serial (*SPI*), circuito *integrado de comunicação (I ^{2} C)* e *receptor transmissor assíncrono universal (UART)*.

Um componente fácil de ser encontrado no mercado é o circuito integrado da família *ADAS1000*, onde sua baixa potência e tamanho pequeno o tornam adequado para utilização em dispositivos portáteis, que exigem baixo consumo. Tais componentes são controlados através de uma interface serial padrão, permitindo a comunicação com sistemas embarcados e captação de dados do eletrocardiograma. [24]

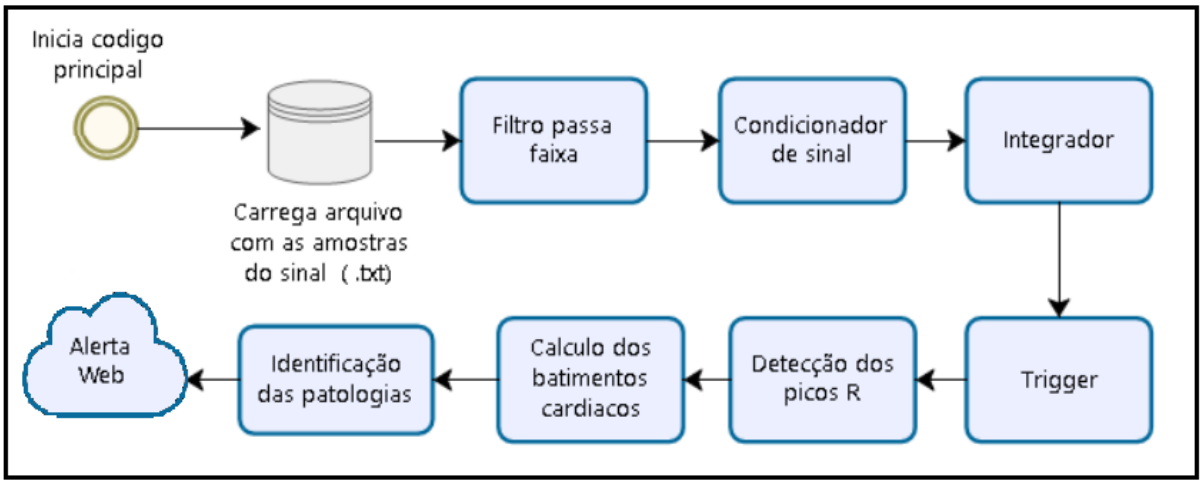
4.2 Firmware proposto

A proposta do *firmware* é fazer o *Raspberry Pi* analisar um banco de arquivos (.txt), disponibilizado pelo professor e orientador André Sanches Fonseca Sobrinho, cujo os arquivos disponibilizam vetores contendo as amplitudes de tensão de sinais cardíacos e caso exista algumas das patologias descritas no capítulo dois

o *firmware* envia um alerta de patologias via Servidor *Web*. Portanto, a partir das amostras de sinais é possível identificar com ajuda do *firmware* proposto as seguintes patologias cardíacas, Parada cardíaca, Bradicardia Sinusal, Taquicardia Sinusal, Flütter Atrial e Fibrilação Atrial.

Para a construção do *firmware* proposto é utilizado o fluxograma demonstrado pela Figura 10, onde este parte do carregamento dos arquivos, com posterior tratamento matemático, cálculo da frequência cardíaca, identificação das possíveis patologias encontradas no sinal analisado e envio do alerta ao servidor.

Figura 10 – Representação do funcionamento da solução proposta



Fonte: Autoria própria.

O fluxograma é composto por nove etapas sendo estas:

- O primeiro bloco tem por objetivo realizar a leitura do banco de dados do sinal cardíaco para posterior analise do *firmware*;
- O segundo bloco tem por objetivo usar um filtro passa-faixa para filtrar os sinais cardíacos apresentados no item anterior, desejado apenas o complexo QRS que tem maior quantidade de energia concentrada na faixa de 5 a 15 Hz, então é aplicado o filtro para que as impurezas e as componentes presentes em outras ondas que compõem o sinal não interfiram no cálculo da frequência cardíaca, pois, algumas vezes estas apresentam amplitudes maiores que os picos R, dificultando o cálculo da frequência cardíaca; [25]
- O terceiro bloco tem por objetivo eliminar os valores negativos da onda com intuito de realçar as componentes do sinal de maior amplitude do

complexo QRS em comparação com os demais componentes do eletrocardiograma; [25]

- O quarto bloco tem por objetivo aplicar um filtro de media móvel sobre o sinal resultante do bloco anterior, com a finalidade de transformar os vários picos encontrados no complexo QRS em pulsos mais largos, facilitando a detecção dos picos R; [25]

- O quinto tem por objetivo aplicar uma função denominada de *trigger,* na geração de um novo sinal contendo amostra de sinais maiores que um dado valor, este tem o objetivo de retirar possíveis sinais ruidosos de pequena amplitude ainda presentes no sinal, que possam interferir no calculo da frequência cardíaca;

- O sexto bloco tem por objetivo a detecção dos picos R no sinal gerado pelo filtro integrador. A detecção é feita por uma função da linguagem *Python* projetada para encontrar picos, onde para cada ponto encontrado se encontra um ponto de máximo do pico do R;

- O sétimo bloco tem por objetivo o cálculo da frequência cardíaca, com base na diferença entre os valores de posição dos picos R respectivos, identificados pela função do item anterior e os multiplicado por uma constante se pode encontrar o valor da frequência cardíaca para o sinal analisado, como o sinal estudado possui mais que dois picos R e plausível se calcular uma média;

- O oitavo bloco tem por objetivo encontrar as patologias levando em conta como característica de análise somente a frequência cardíaca. Desta forma, as patologias que já foram citadas podem ser identificadas nesta parte do Firmware;

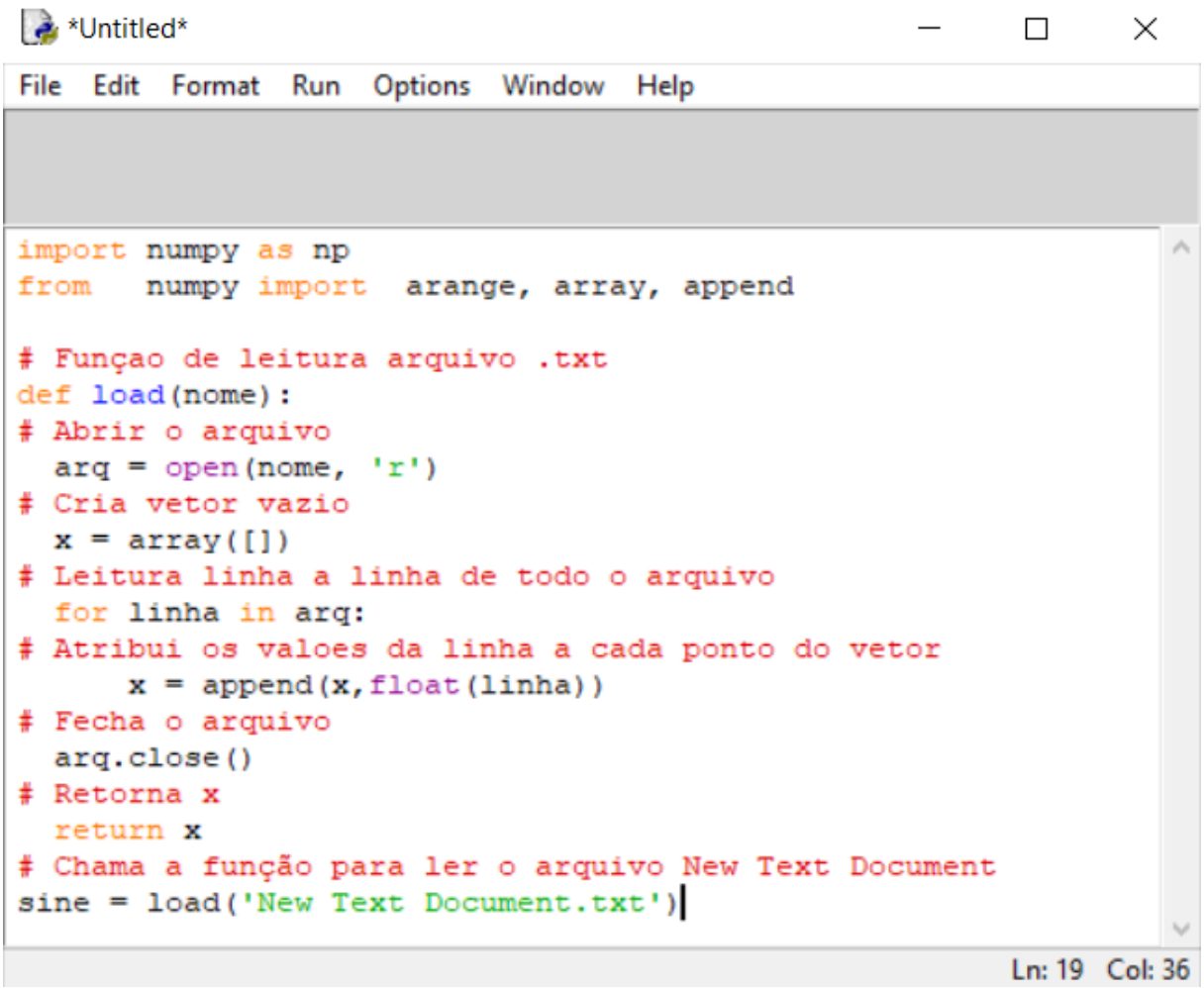
- O nono bloco tem por objetivo iniciar o servidor Web caso exista a presença de alguma patologia e informar qual foi a patologia encontrada no sinal analisado.

Com base no fluxograma proposto e na explicação da finalidade de cada item, é possível o desenvolvimento do Firmware proposto, para que este seja capaz de realizar a análise de um eletrocardiograma e identificar as seguintes patologias cardíacas, Parada cardíaca, Bradicardia Sinusal, Taquicardia Sinusal, Flütter Atrial e Fibrilação Atrial.

4.2.1 Leitura de dados

Dando inicio ao desenvolvimento do firmware é necessário que Python possa ler as amostras do sinal cardíaco e posteriormente fazer as análises foi necessário a programação de um algoritmo, com auxilio da biblioteca *Numpy*, onde tal algoritmo transfere os valores contidos em um arquivo de texto (. txt), para um vetor, onde este possui uma frequência de amostragem igual a 100 Hz, com mil amostras e contendo como patologia cardíaca Bradicardia Sinusal e com frequência de 60bpm, conforme demostra a Figura 11.

Figura 11 – Leitura de dadas

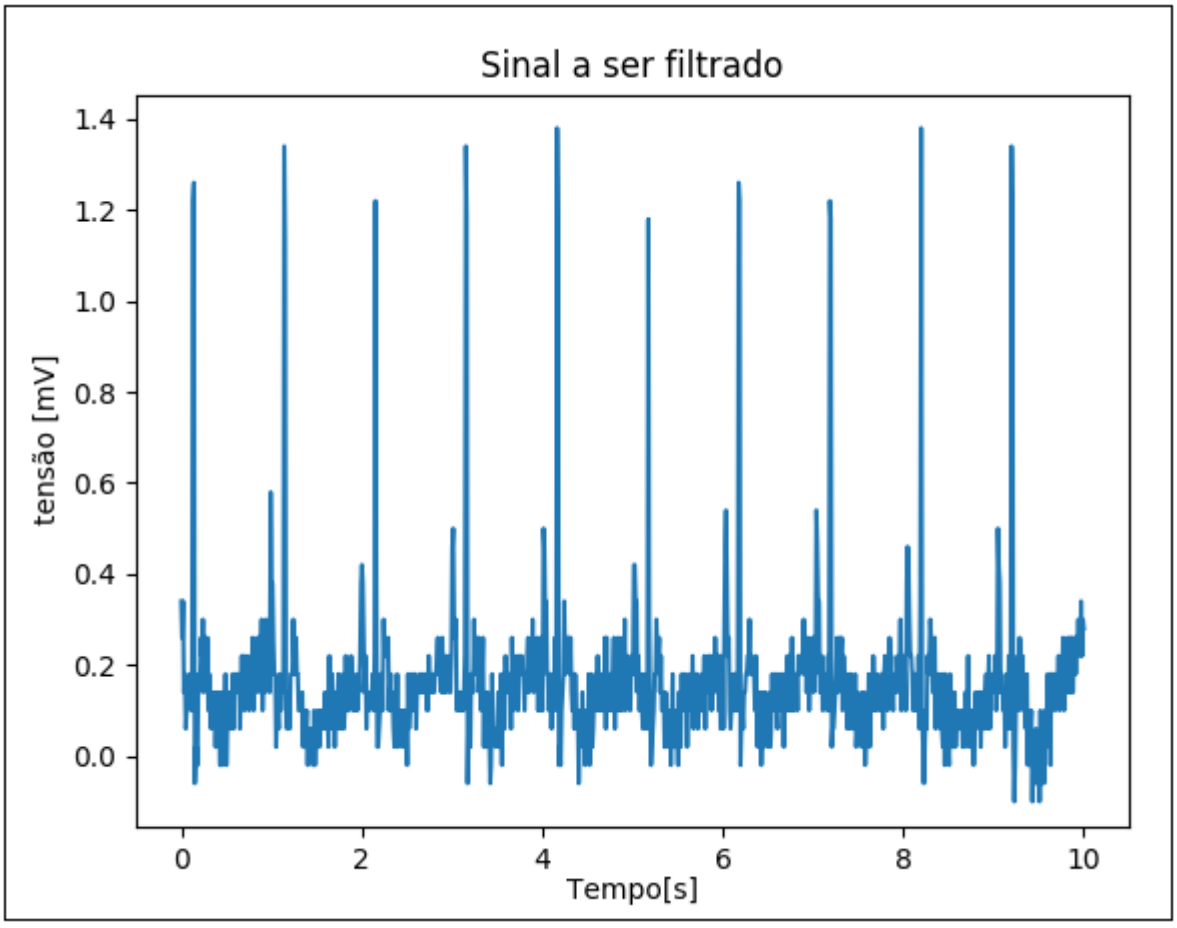


Fonte: Autoria própria.

Após o carregamento dos dados é possível gerar um gráfico contendo o sinal cardíaco a ser analisado com o auxilio da biblioteca *Matplotlib*, utilizando o seguinte comando *''plot(sine)''* gera se um gráfico com a amplitude das amostras de

tensão do sinal cardíaco em milivolts pela quantidade de amostras, no entanto, o interesse é visualizar os sinais em escala de tempo, sendo assim é necessário a realização da conversão de numero de amostras para tempo, de forma manual, ou seja com o inverso do valor da frequência de amostragem multiplicado pela quantidade de amostras, se pode encontra o valor do eixo em função do tempo, sabendo que o sinal utilizado neste exemplo tem frequência de 100Hz, obtém-se a Figura 12, dada em milivolts por segundos.

Figura 12 – Gráfico contendo o sinal cardíaco *plot(sine)*



Fonte: Autoria própria.

4.2.2 Filtragem sinal

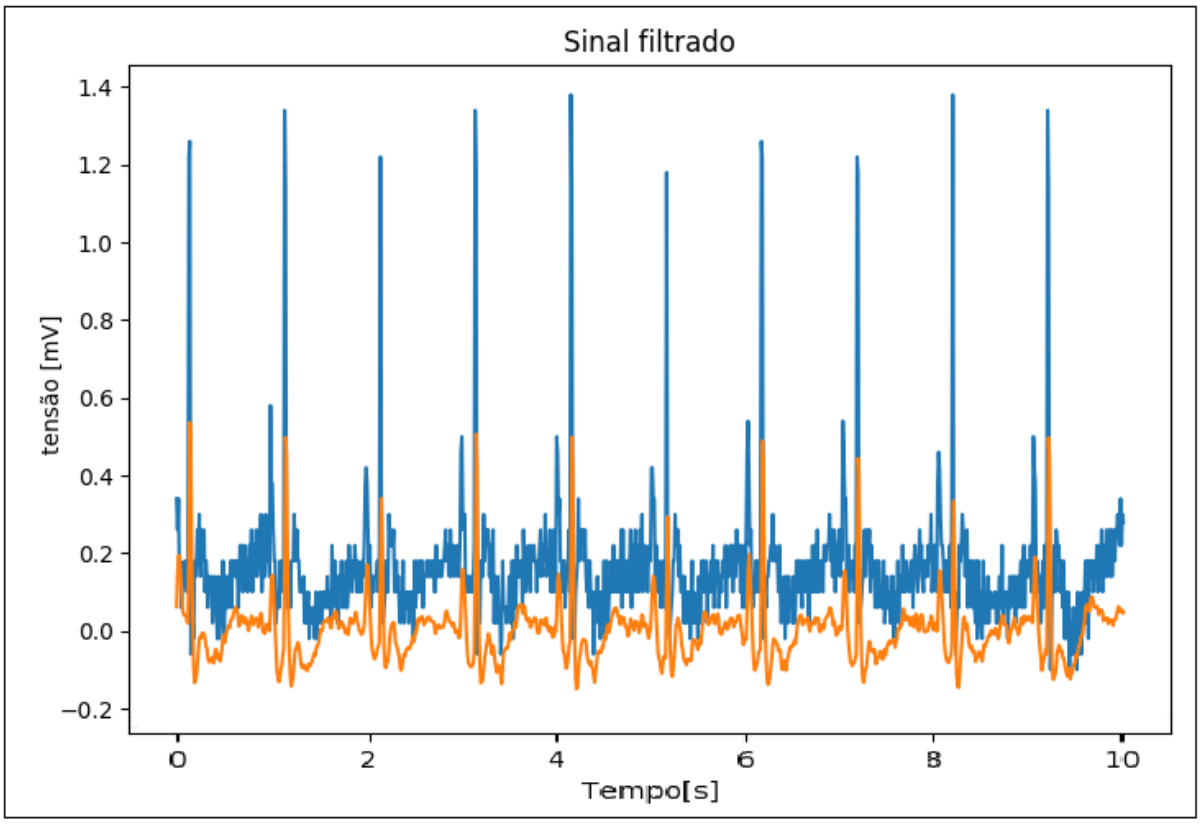
Com o auxilio da biblioteca SciPy e de suas sub-bibliotecas, é possível a programação do código, apêndice A, onde tal algoritmo tanto carrega os dados do sinal cardíaco, contidos no arquivo texto (.txt), como também os filtra com um filtro passa faixa, de segunda ordem. Os parâmetros ‘‘highcut’’ e ‘’Lowcut’ encontrados no

algoritmo definem respectivamente a frequência baixa de corte e a frequência alta de corte.

A variação da frequência alta de corte (highcut) no algoritmo para valores maiores ou iguais a 17HZ não se observa muita influência no sinal final, porém variando as frequências baixas de corte, se observa distorções no sinal filtrado em comparação com o sinal original. Variando a frequência de corte inicial (Lowcut) de 0 a 17 Hz, pode-se observar o comportamento do filtro e encontra um valor de corte que não cause deformação no final original. Tal valor é igual a 5Hz, encontrado com base em tentativa e erro. Sendo assim, as frequências de corte de baixa e alta foram definidas como 5Hz e 17Hz respectivamente.

Como resultado da filtragem se tem apenas as componentes presentes no complexo QRS, sem a presença de impurezas e das componentes presentes nas outras ondas que compõem o sinal, conforme se pode observar no gráfico que apresenta a resposta do filtro disponível na Figura 13, onde esta apresenta o sinal original em maior amplitude na cor azul e o de menor amplitude corresponde ao sinal filtro na cor laranja.

Figura 13 – Sinal filtrado

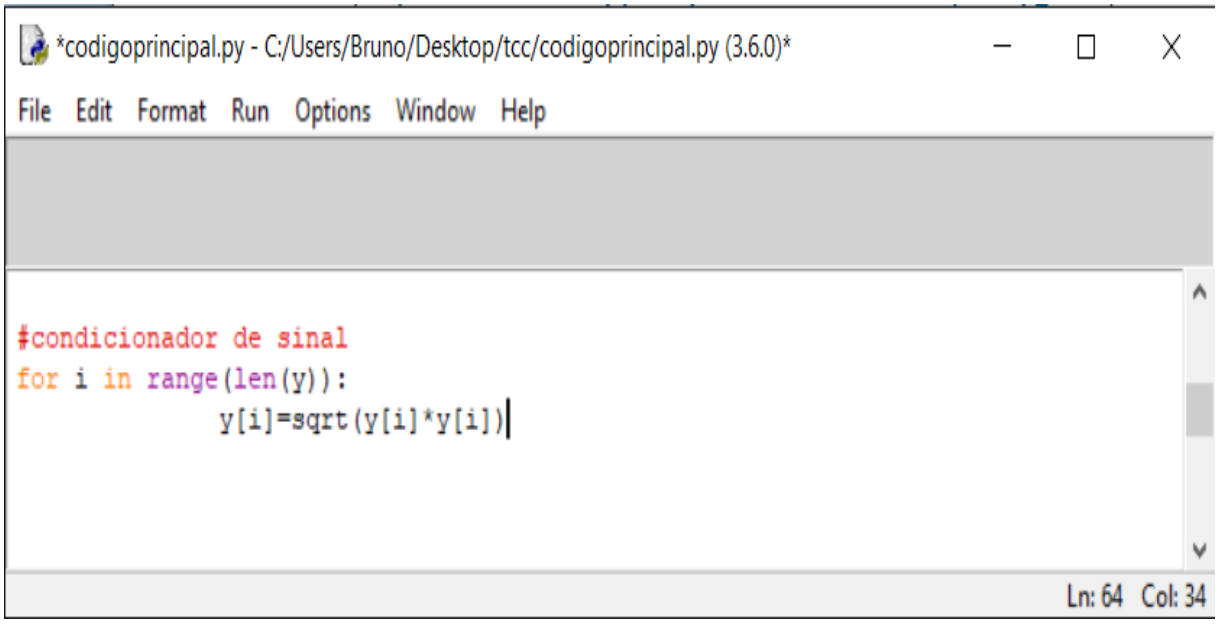


Fonte: Autoria própria.

4.2.3 Condicionamento do sinal

Para que se consiga fazer o condicionamento e necessário a implantação de um ‘’*for*’’ dentro do algoritmo que multiplica cada ponto por ele mesmo e se extrai a raiz quadrada, conforme demonstrado pela Figura 14, que contém a algoritmo necessário para o condicionamento do sinal.

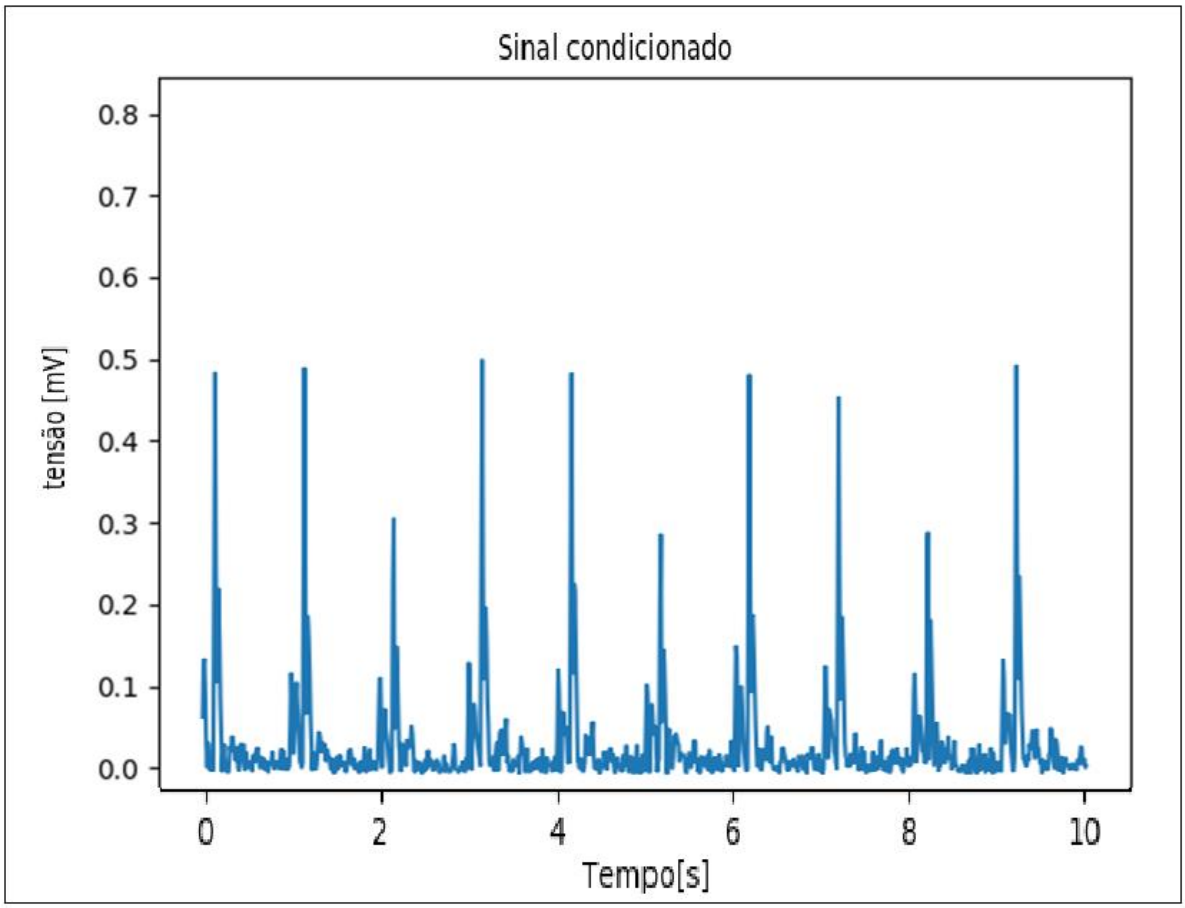
Figura 14 – Código para o condicionamento de sinal



Fonte: Autoria própria.

Como resultado do condicionamento do sinal pode-se observar valores não negativos como também aumentos nos valores das componentes do sinal de maiores amplitudes do complexo QRS em comparação com os demais componentes do eletrocardiograma, conforme se pode observar na Figura 15.

Figura 15 – Sinal condicionado



Fonte: Autoria própria.

4.2.4 Filtro do tipo média móvel (Integrador)

O uso do filtro do tipo média móvel faz com que os vários picos encontrados no Complexo QRS sejam transformados em um só pulso mais largo, facilitando a detecção dos picos R [24]. As amostras de saída dependem somente da entrada presente e de um número finito de suas amostras passadas. O número de amostras da janela móvel deve ser escolhido com cuidado, pois se este for muito grande, a janela pode agrupar dois complexos QRS no mesmo pulso. Por outro lado, se a janela for muito pequena, pode acontecer de haver mais de um pulso para o mesmo complexo QRS. A quantidade de amostras utilizadas no código é igual a 20, no qual apresentou bons resultados com diversos sinais eletrocardiográficos. A expressão das diferenças utilizada para a implementação deste filtro é mostrada na Equação 1.

$$y[n] = \frac{1}{N} \sum_{i=0}^{N-1} x[n-i] \tag{1}$$

Onde, N é o número de amostras.

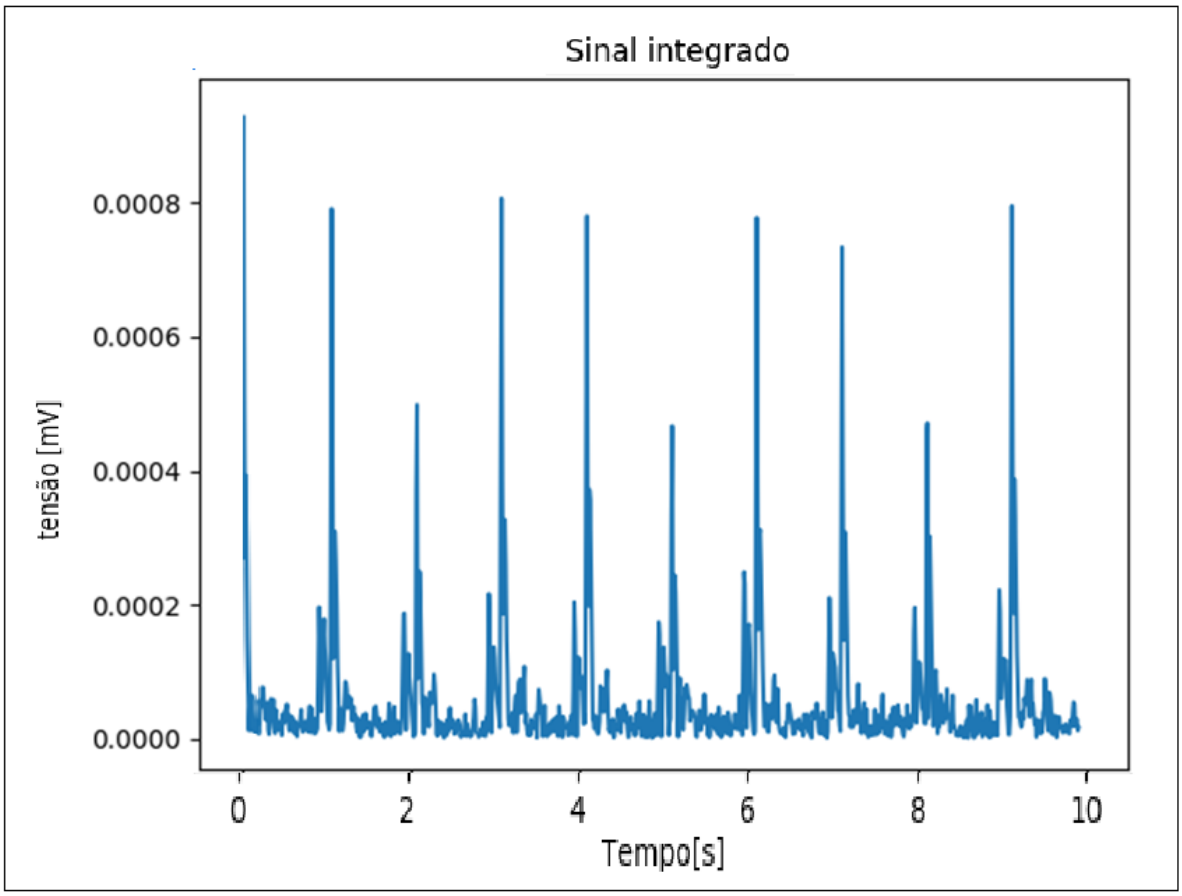
Para que se consiga fazer o condicionamento utilizando à linguagem de programação *Python* e necessário a implantação de um ‘’*for*’’ dentro do algoritmo, conforme demonstrado pela Figura 16, que contém a algoritmo necessário para o condicionamento do sinal.

Figura 16 – Código para o filtro do tipo média móvel

Fonte: Autoria própria.

Como resultado da passagem do sinal pelo filtro de media móvel se observa os vários picos encontrados no complexo QRS com pulsos mais largos, conforme se pode observar na Figura 17, facilitando posteriormente a detecção dos picos R.

Figura 17 – Sinal integrado filtro

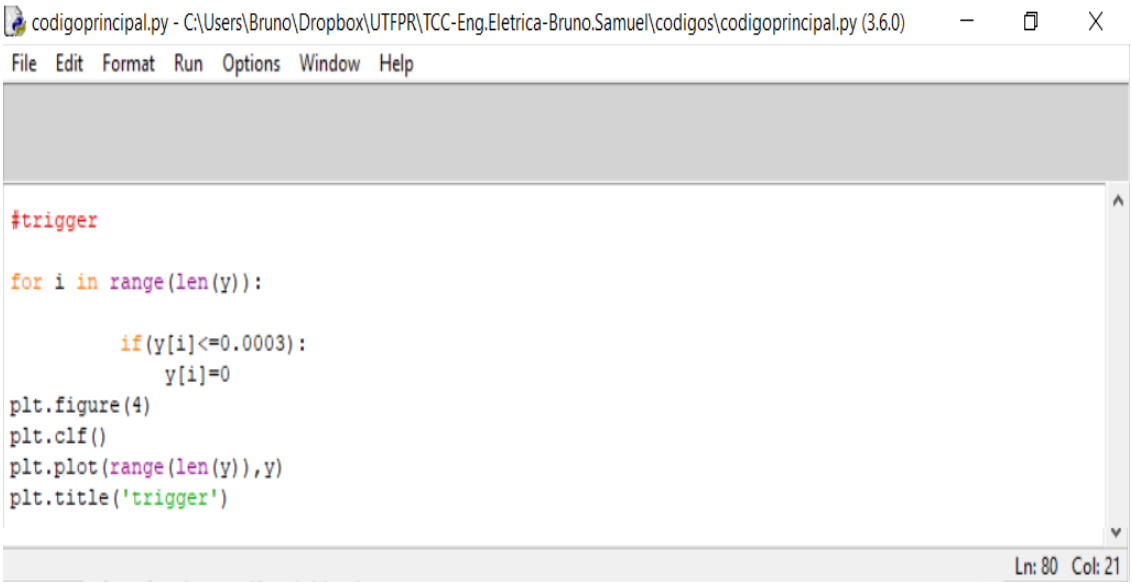


Fonte: Autoria própria.

4.2.5 *Trigger*

O *trigger,* que tem como objetivo criar um novo vetor com valores de amostras maiores que um dado valor, a partir de testes chegou se em um valor de 0.0003, conforme demonstrado pela Figura 18, que contém a algoritmo necessário para a implantação da função.

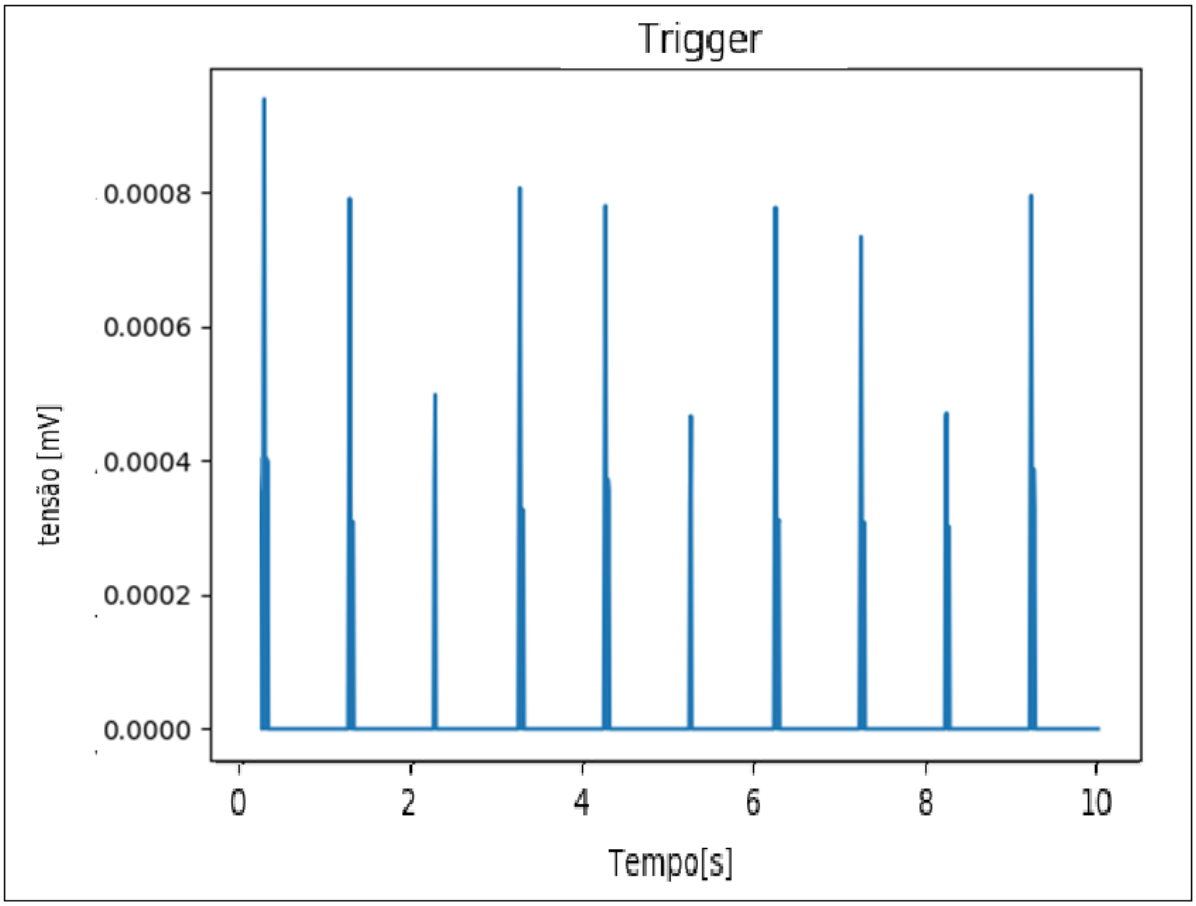
Figura 18 – Código para o *Trigger*



Fonte: Autoria própria.

Como resultado da passagem do sinal pela função *trigger* tem-se a geração de um novo sinal sem a presença de sinais ruidosos de pequena amplitude que poderiam interferir na detecção dos picos R, ou seja, o sinal de resposta após passagem pela função deve resultar apenas nos valores das amplitudes da onda R, conforme Figura 20 demostra.

Figura 20 – Sinal com *trigger*



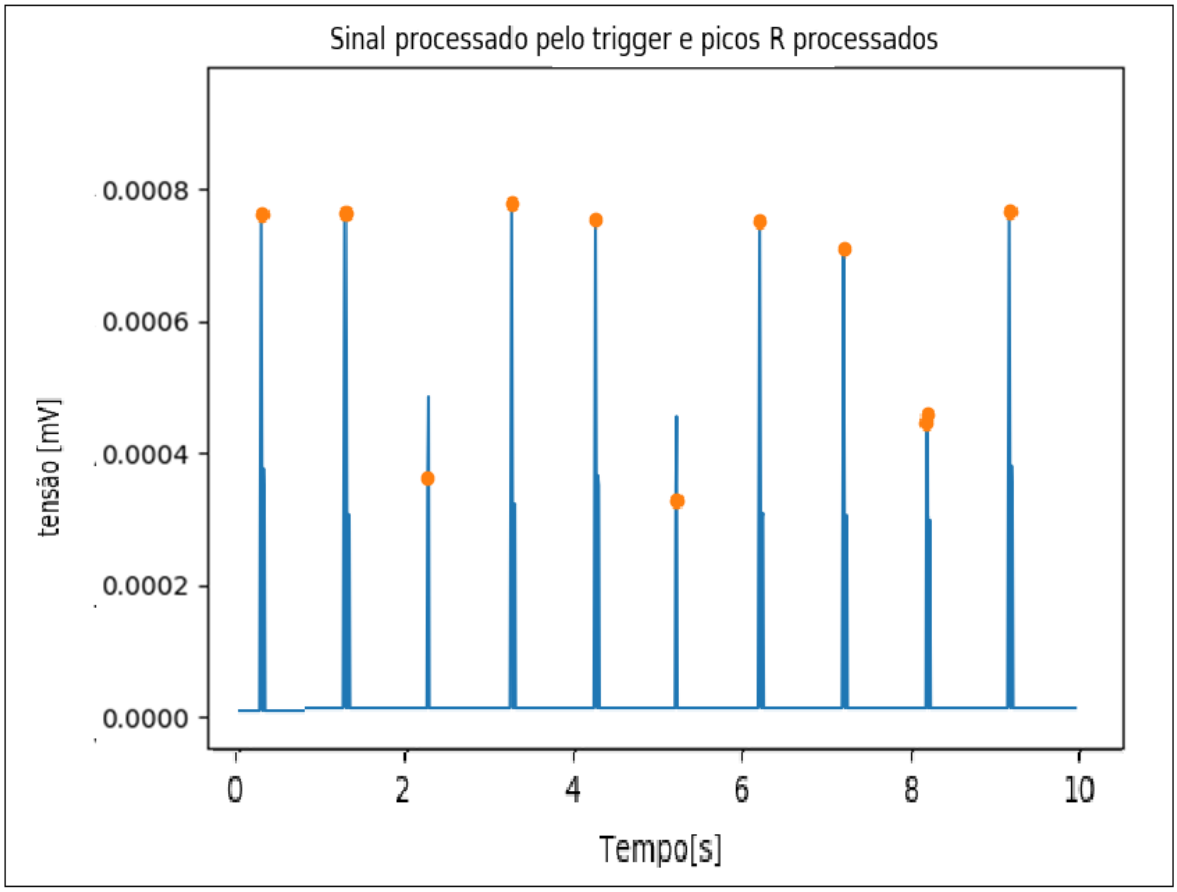
Fonte: Autoria própria.

4.2.6 Função *find_peaks_cwt*

A função *find_peaks_cwt* foi projetada para encontrar picos pontiagudos entre os dados, no entanto, com a seleção adequada de parâmetros, ela funciona bem para diferentes formatos de pico, sendo facilmente implementada usando uma única linha ‘’ *peakinds = find_peaks_cwt(y, np.arange(Ts,Ts*(len(y))*’’, onde os parâmetros são amplitude do sinal e tempo [25]

Após o sinal ser processado pelo *trigger*, pode se utilizar-se da ferramenta de detecção de picos locais, com isto obtém se como resultado os picos identificados na Figura 21, simbolizados por pequenos círculos laranja.

Figura 21 - Gráfico da detecção dos picos após utilização do *Trigger*



Fonte: Autoria própria.

Agora com estes valores pode se partir para o cálculo da frequência cardíaca, conforme será demonstrado no item a seguir.

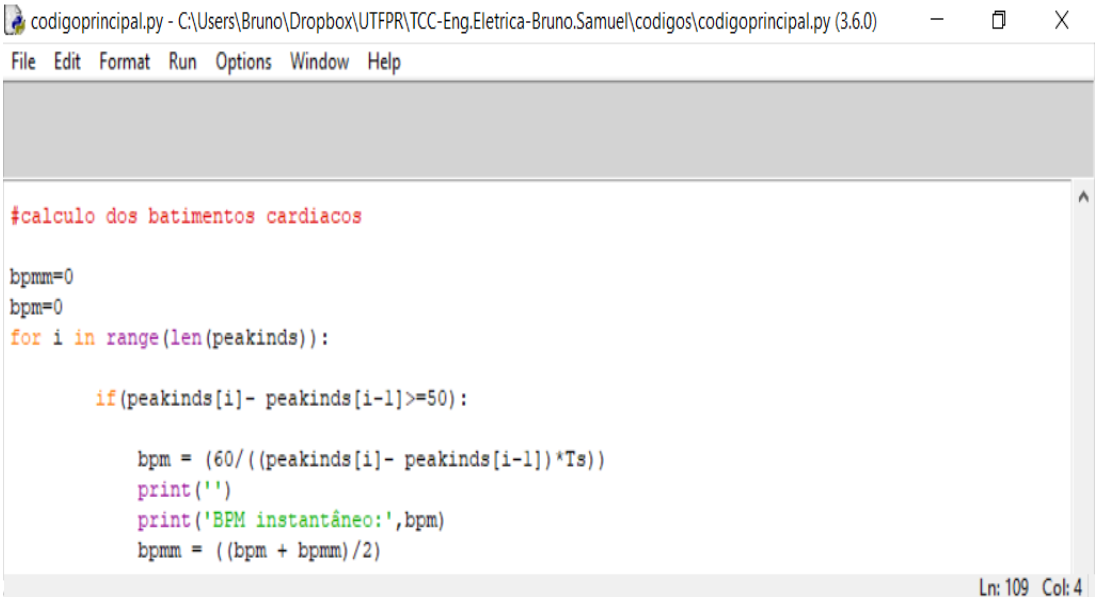
4.2.7 Cálculo frequência cardíaca

Após encontrar os picos R com as técnicas já descritas, é possível calcular o valor da frequência cardíaca em batimentos por minuto (BPM), encontrando a diferença entre os valores de posição entre cada pico R respectivo, identificados pela função ‘’*find_peaks_cwt*’’ e os multiplicando por uma constante de tempo, onde esta constante é dada por 60 divido pelo tempo de amostragem do sinal.

Como o sinal estudado possui mais que dois picos R, é plausível calcular uma média entre os valores de BPM, assim, minimizando os erros para o cálculo da

frequência cardíaca real e posteriormente ser possível a classificação das patologias presentes no sinal. A Figura 22 demonstra a parte do código proposto responsável pelo cálculo da frequência cardíaca.

Figura 22 - Cálculo da frequência cardíaca



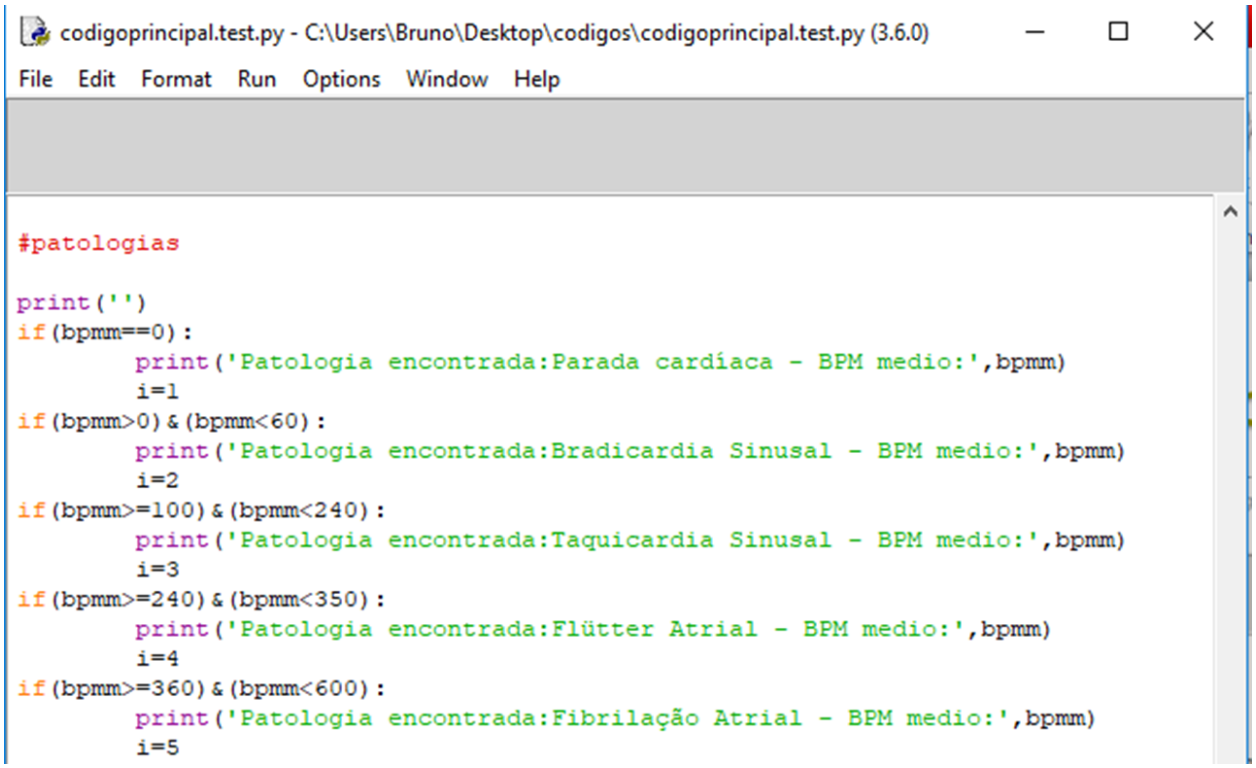
Fonte: Autoria própria.

4.2.8 Classificação de patologias

Após o cálculo da frequência cardíaca é possível a classificação das patologias cardíacas já descritas. Assim, é possível encontrar as patologias presentes no sinal, com aplicação de alguns ‘’if’’ para cada condição de patologia cardíaca existente, onde este relaciona a patologia cardíaca com o valor da frequência cardíaca encontrada.

Caso exista alguma patologia presente no sinal o firmware disponibiliza na janela Shell do Python uma mensagem contendo a patologia presente e valor do batimento por minuto, como também atribui um valor para a variável i de acordo com cada patologia encontrada, onde esta variável o firmware utiliza como referência para inicialização do servidor Web, conforme a Figura 23 demonstra.

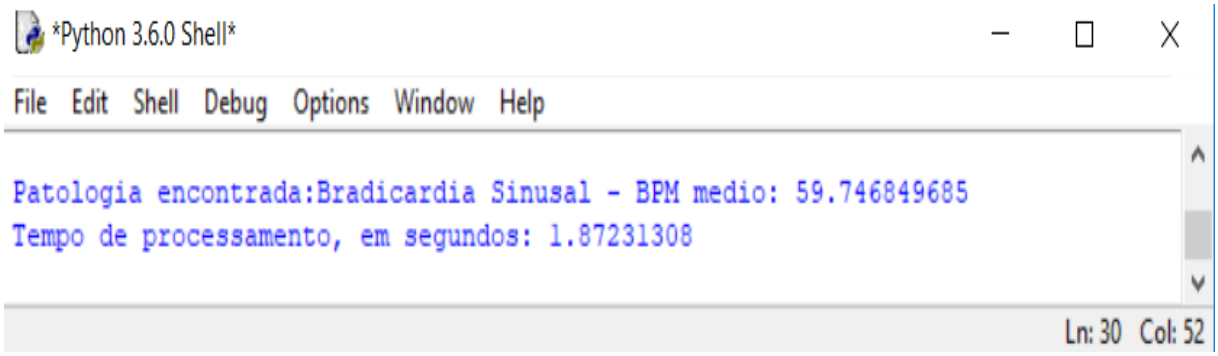
Figura 23 - Detecção de patologias cardíacas



Fonte: Autoria própria.

A Figura 24, demostra a resposta dada na janela Shell do Python, com a patologia presente e o valor do BPM encontrado apos processamento do *firmware*, que posteriormente é enviada ao servidor web.

Figura 24 - Resposta do *firmware*

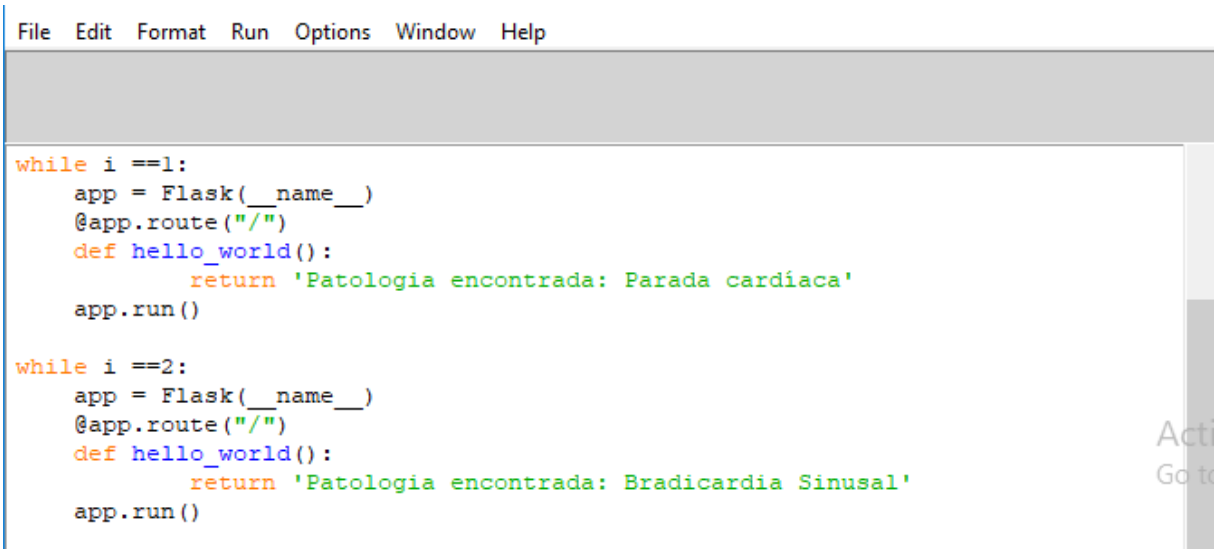


Fonte: Autoria própria.

4.2.9 Servidor Web flash

Para que o servidor responda de acordo com a patologia encontrada são utilizados funções “while”, onde a partir do valor da variável ‘’i’’ definido no item anterior se inicia um servidor Web, retornando uma mensagem contendo a patologia encontrada, conforme a Figura 25 demonstra.

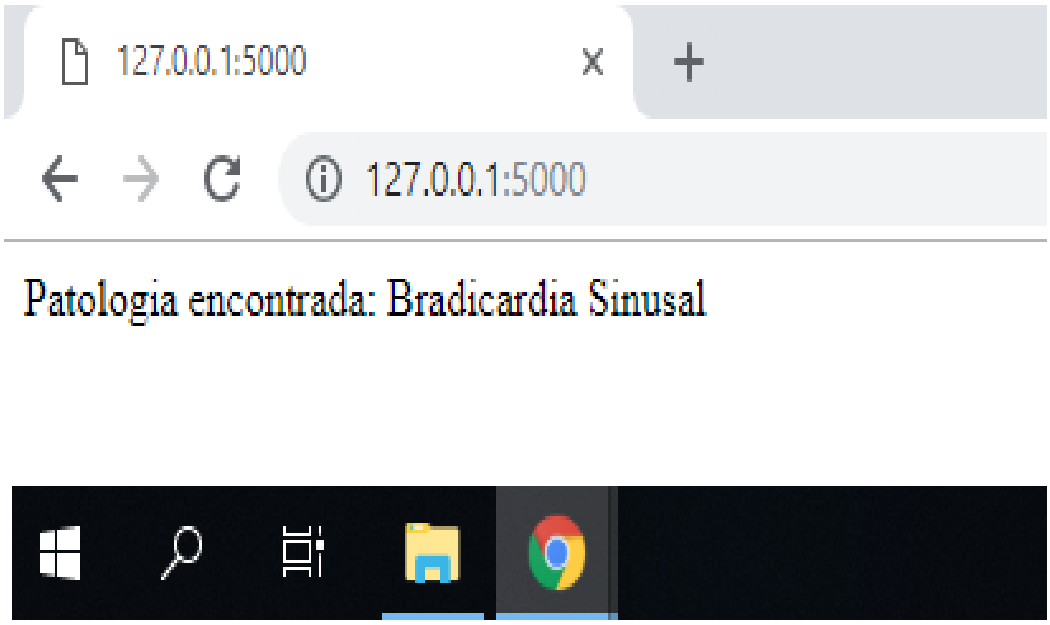
Figura 25 - Servidor Web



Fonte: Autoria própria.

Para acessar a aplicação Web criada e verificar a patologia encontrada deve-se acessar o endereço: <http://localhost:5000> de um navegador Web, conforme a Figura 26 demostra.

Figura 26 - Resposta do servido *Web*



Fonte: Autoria própria.

Como pode-se observado após passagem do sinal pelo *firmware* proposto descrito nos itens anteriores, vemos que o firmware se comporta da forma como previsto, pois, era esperado encontrar como resposta para o sinal uma patologia cardíaca do tipo Bradicardia Sinusal, como também dar o inicio no servidor *Web* contendo a patologia encontrada, sendo, assim o sofware cumpriu seu objetivo quanto a analise de patoligas do tipo citado, comparando o valor real da frequência cardíaca de valor igual 60 BPM com o encontrado pelo *firmware* de valor 59,75 BPM, obtem-se um erro de aproximadamente 0,42%.

5 Resultados

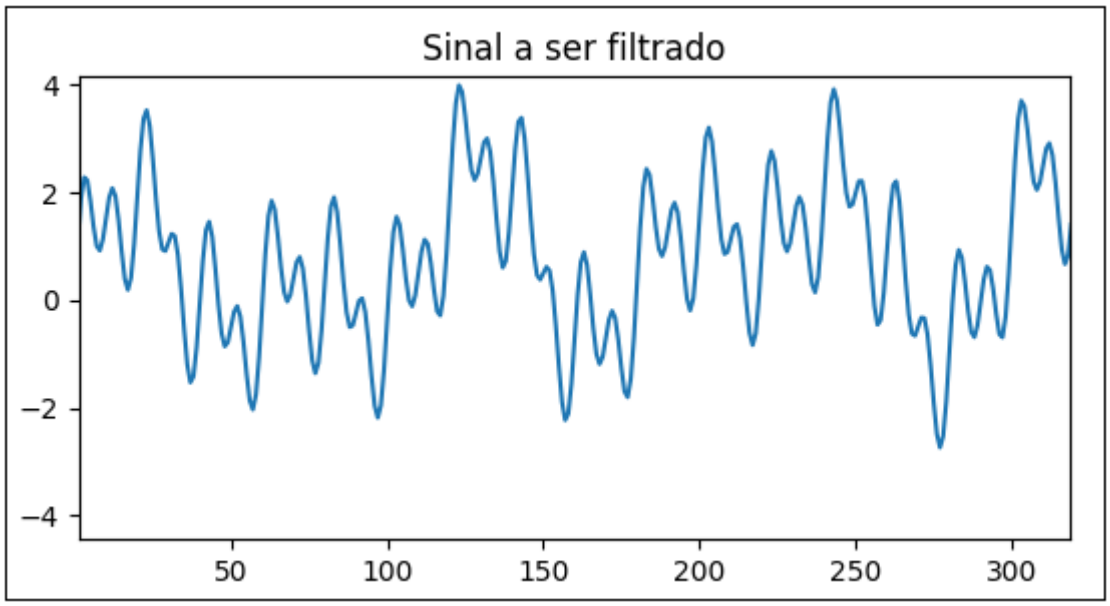
Este capítulo apresenta os testes realizados no firmware desenvolvido.

5.1 Teste do filtro em *Python (Butter_Bandpass_filter)*

Com o intuito de avaliar a ferramenta de filtragem, como um filtro passa faixa, programou-se um algoritmo disponibilizado no apêndice C. O algoritmo tem como função somar ondas senoidais e posteriormente filtrá-las na frequência desejada, que no caso o projeto exige que sejam centrados em 17Hz para melhor detecção dos sinais cardíacos.

Para realizar os testes, simulou se cinco ondas senoidais nos valores de 100Hz, 50Hz, 17Hz, 10Hz e 1Hz, posteriormente se somam as cinco ondas senoidais geradas, resultando no sinal apresentado pela Figura 27.

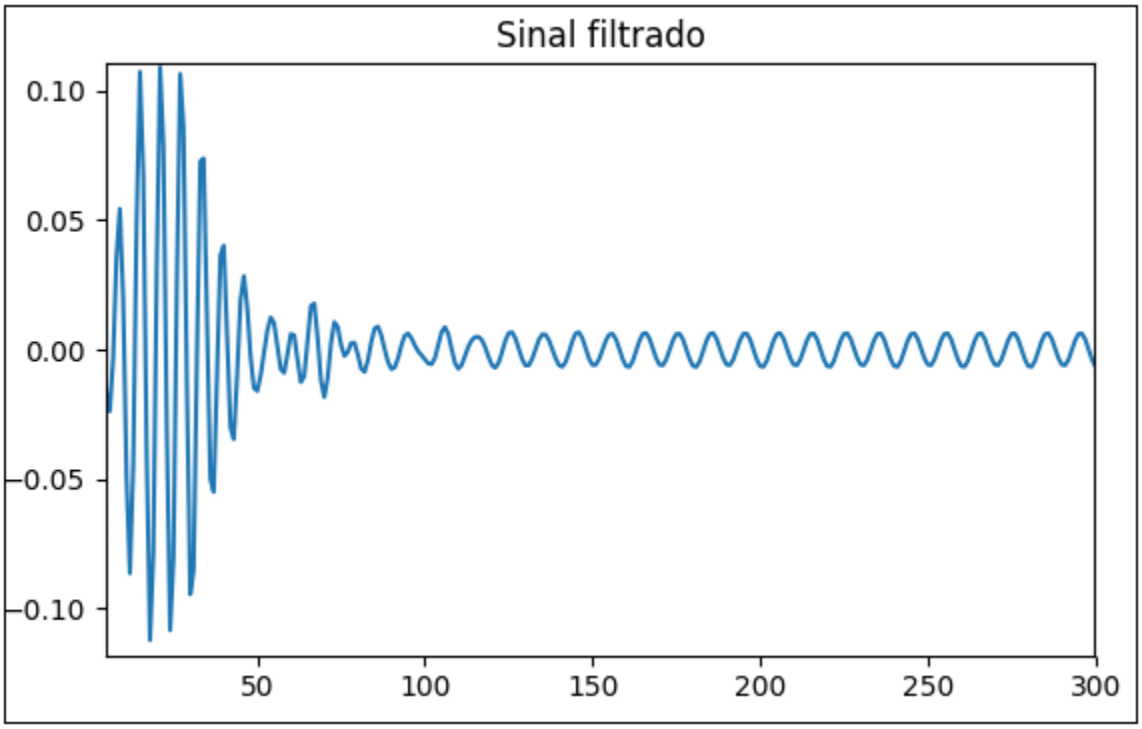
Figura 27 - Onda resultante



Fonte: Autoria própria.

Logo em seguida, com o auxilio das bibliotecas de filtragem já citadas para linguagem Python, onde a partir destas programou se um filtro passa-faixa de terceira ordem, com frequência de corte inicial igual a 15 Hz e final igual a 18 Hz, onde tal filtro teria por objetivo selecionar apenas a frequência de 17 Hz, os dados filtrados estão demonstrados na Figura 28.

Figura 28 - Sinal filtrado



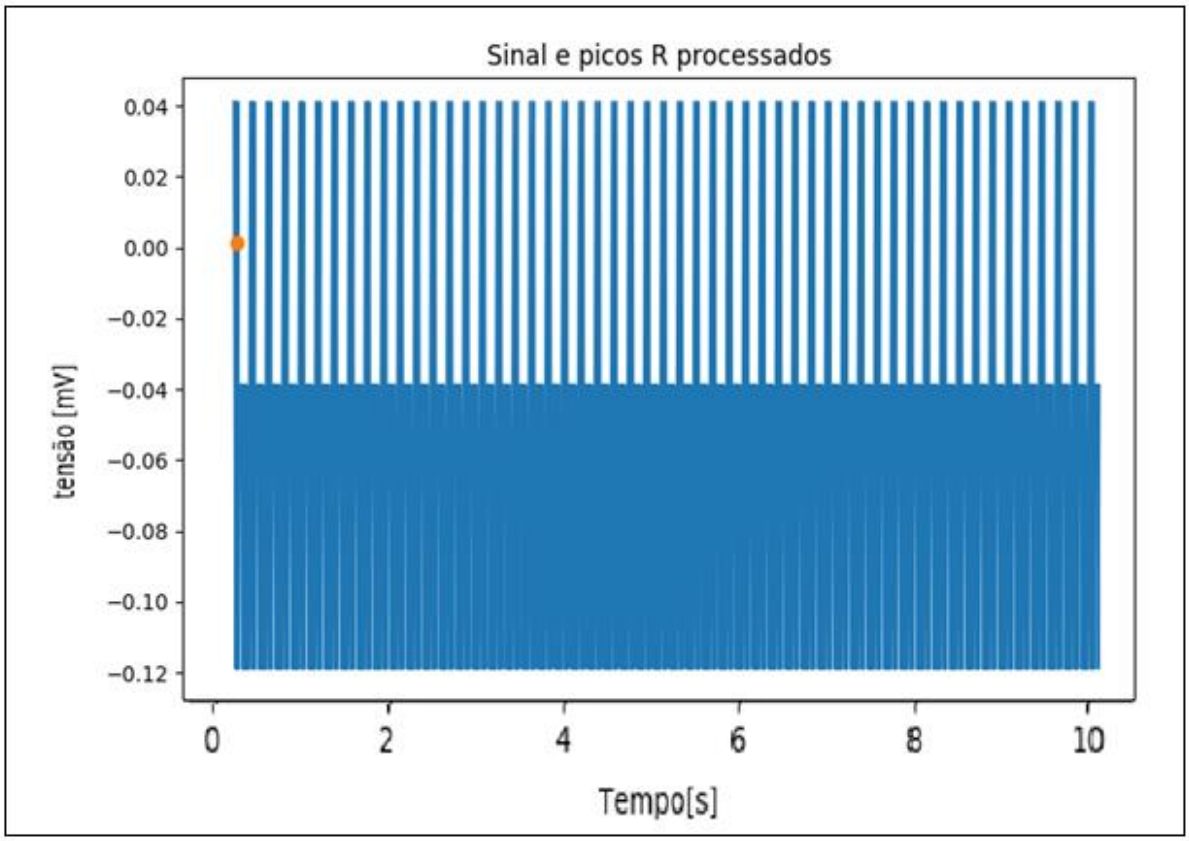
Fonte: Autoria própria.

Após aplicação do filtro pode-se concluir como resultado que o filtro operou de forma adequada, selecionando apenas a frequência desejada, justificando a deformação no inicia do sinal pela pouca quantidade de valores para cálculo do sinal resultante, conforme o número de amostras aumenta o filtro se estabiliza.

5.2 Teste *firmware* parada cardíaca

Com intuito de se averiguar a capacidade de processamento do *firmware* proposto, desenvolvido e apresentado no item 4.2 é utilizado um sinal com frequência de amostragem igual a 100 Hz, com mil amostras e contendo como patologia uma parada cardíaca e com frequência cardíaca de 0 bpm, conforme o sinal representado pela Figura 29, onde tal figura representa um sinal ruidoso, no entanto sem a presença de picos R. Após o processamento podemos observar que o f*irmware* encontra apenas um pico, que condiz com inicio do processamento.

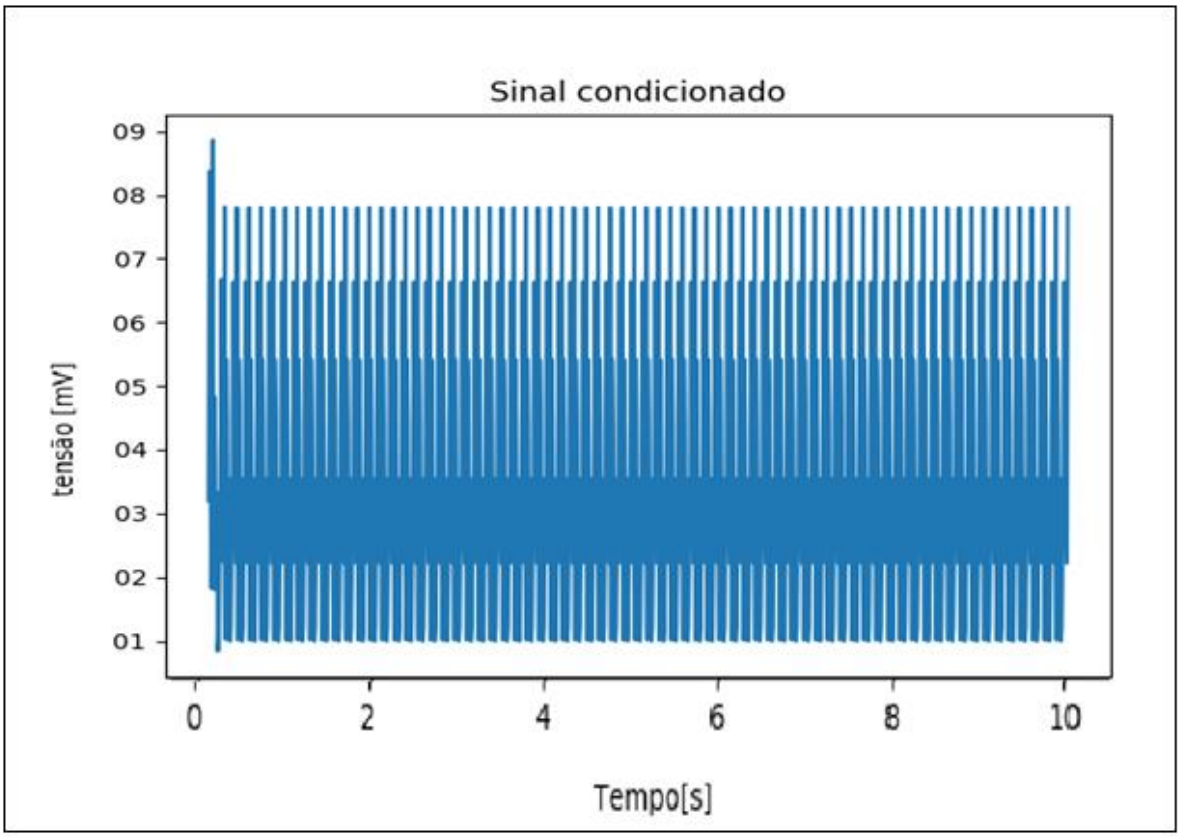
Figura 29 - Sinal cardíaco a ser analisado contendo como patologia parada cardíaca



Fonte: Autoria própria.

A Figura 30 representa a etapa de condicionamento do sinal para o sinal de teste contendo como patologia parada cardíaca.

Figura 30 - Sinal condicionado

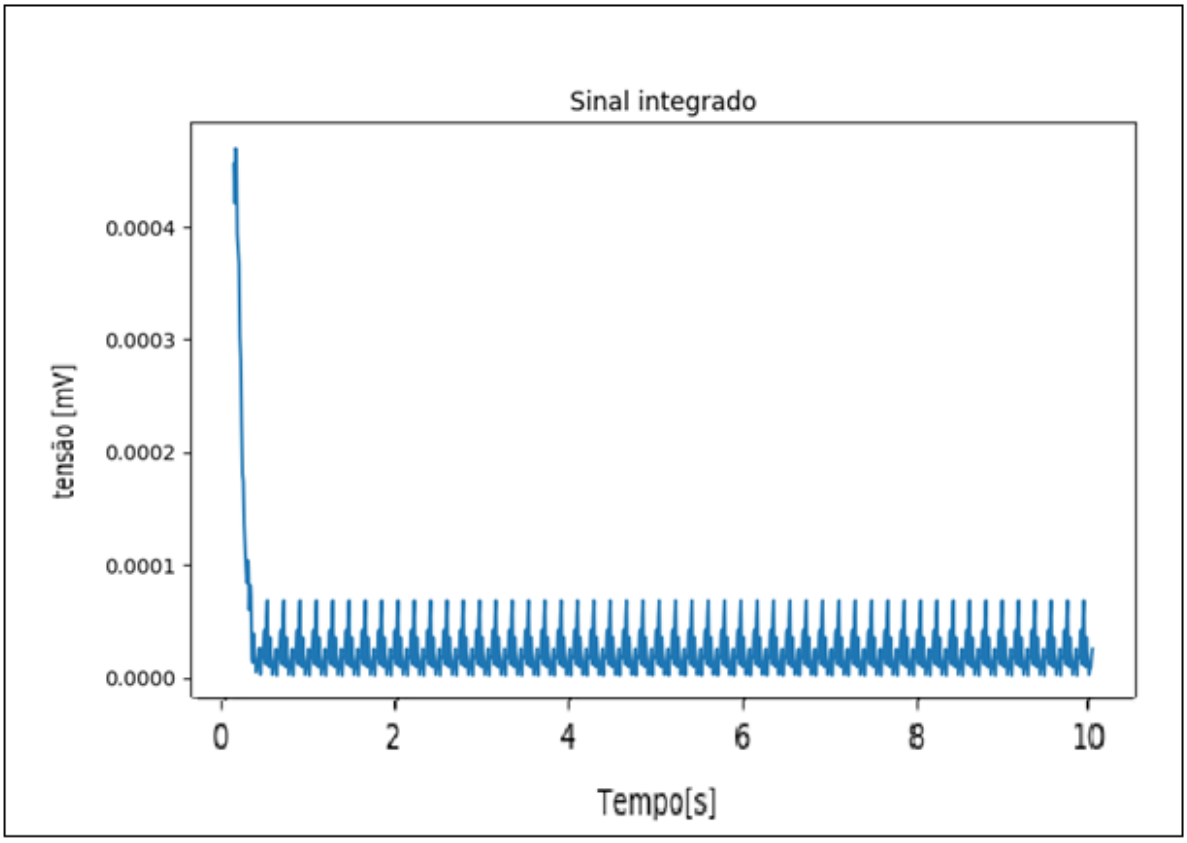


Fonte: Autoria própria.

Após o condicionamento pode-se observar, que existe uma elevação no inicio do sinal, que condiz com o ponto de pico encontrado no final do processamento, conforme a Figura 29 demostra, este pico é gerado devido a pouca quantidade de amostra do sinal, conforme o numero de amostras aumenta o sinal se estabiliza.

A Figura 31 representa à resposta do *firmware* a passagem pelo filtro integrador de sinal, para o sinal de teste contendo como patologia parada cardíaca.

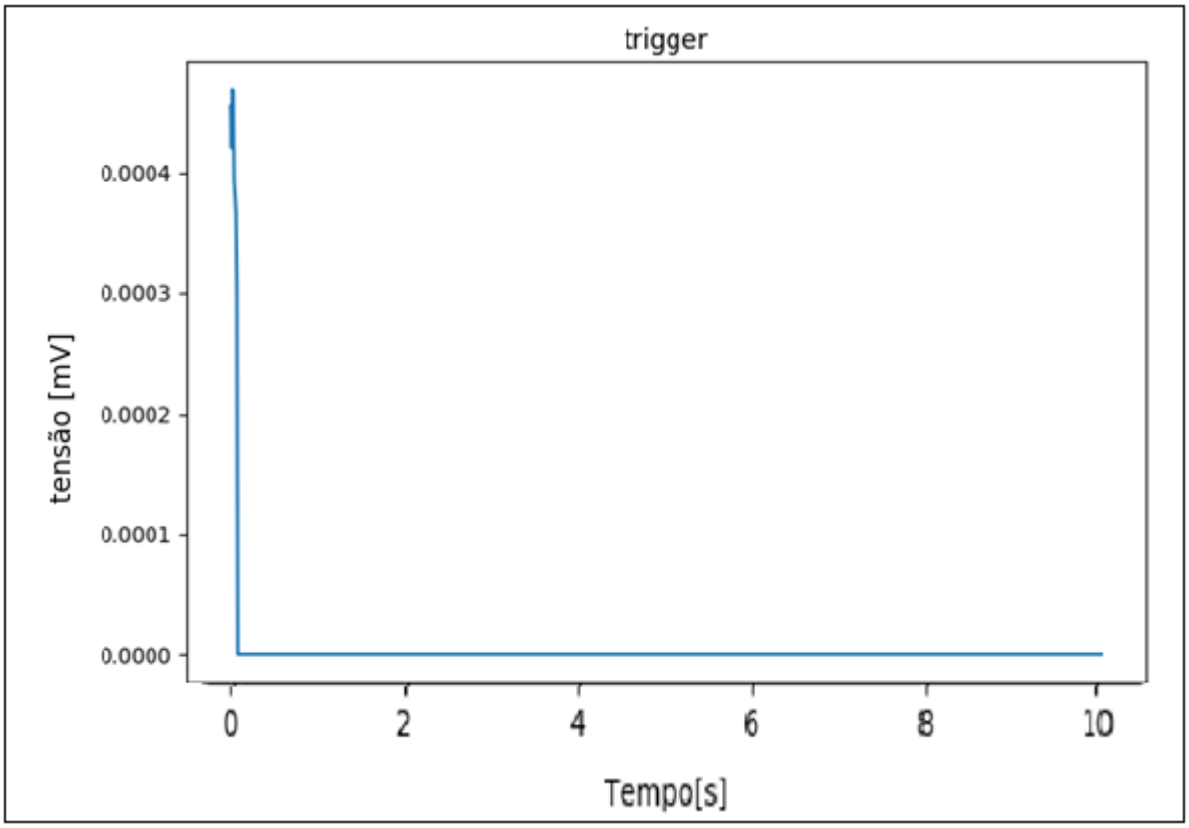
Figura 31 - Sinal integrado



Fonte: Autoria própria.

A Figura 32 representa à resposta do *firmware* a passagem pela função *trigger*, para o sinal de teste contendo como patologia parada cardíaca.

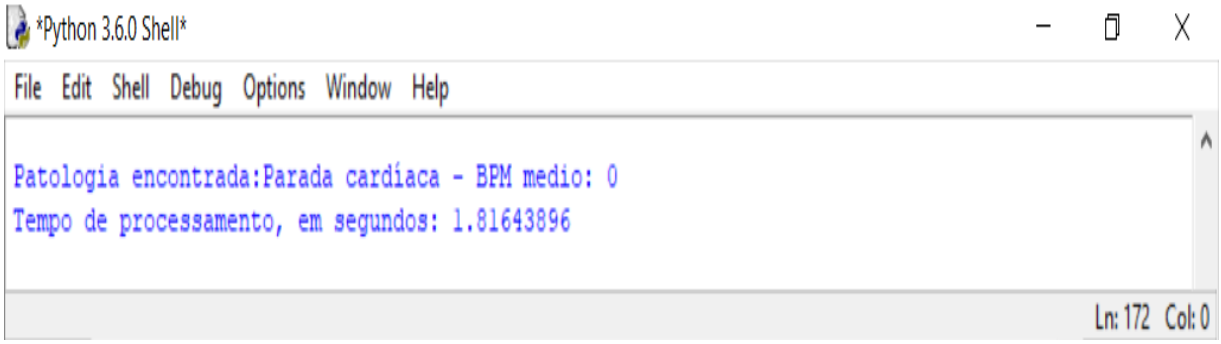
Figura 32 - Sinal com *trigger*



Fonte: Autoria própria.

A Figura 33 representa à resposta dada na janela *Shell* do *Python* com a patologia presente, que posteriormente é enviada ao servidor web, nela é possível a obter o valor da frequência cardíaca por minuto media, com valor igual 0 BPM.

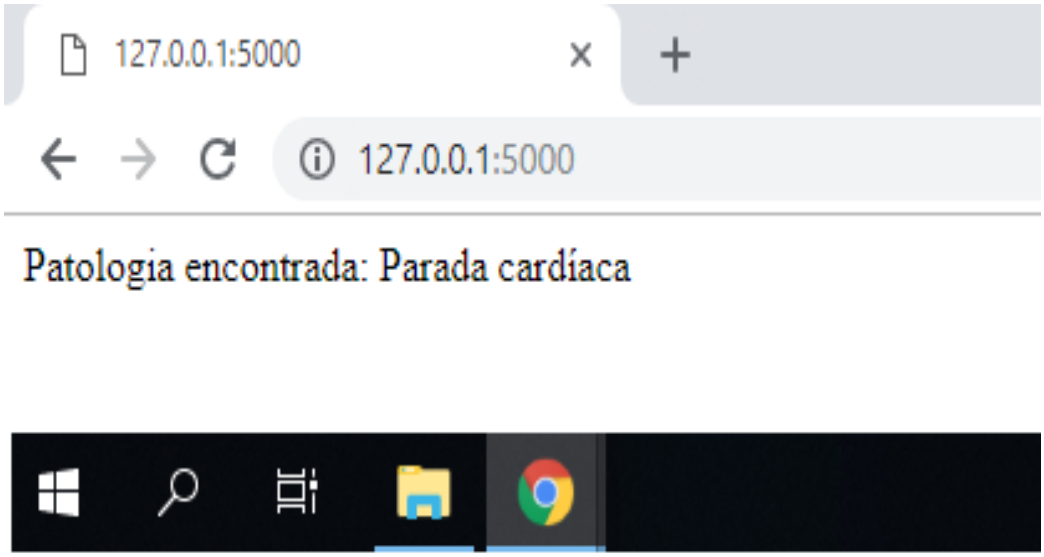
Figura 33 - Resposta do *firmware*



Fonte: Autoria própria.

A Figura 34 demostra o alerta apresentado via página *Web*, onde tal apresenta a patologia cardíaca encontrada no sinal analisado via *firmware.*

Figura 34 - Resposta do servido *Web*



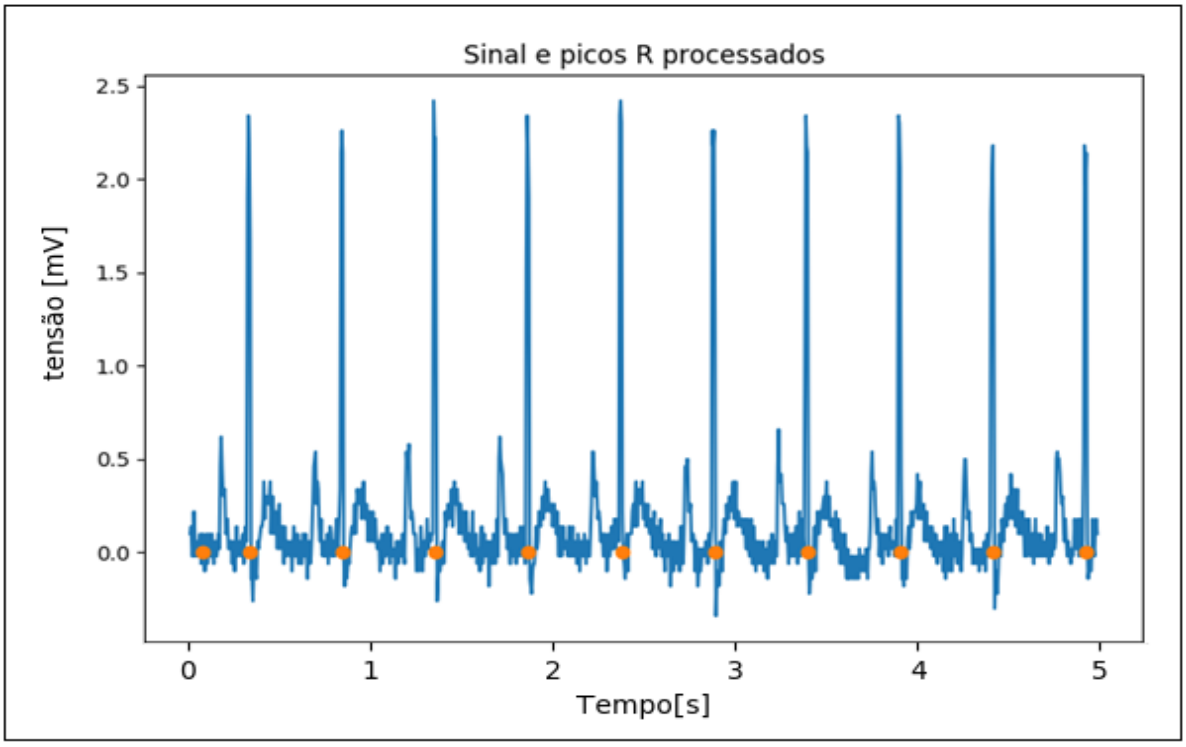
Fonte: Autoria própria.

Como pode-se observado após passagem do sinal pelo *firmware* proposto descrito nos itens anteriores, vemos que o firmware se comporta da forma como previsto, pois, era esperado encontrar como resposta para o sinal uma patologia cardíaca do tipo Parada Cardíaca, como também dar o inicio no servidor *Web* contendo a patologia encontrada, sendo, assim o *firmware* cumpriu seu objetivo quanto a análise de patoligas do tipo citado, comparando o valor real da frequência cardíaca de valor igual 0 BPM com o encontrado pelo *firmware* de valor 0 BPM, obtem-se um erro de 0%.

5.3 Teste firmware Taquicardia Sinusal

Com intuito de se averiguar a capacidade de processamento do *firmware* proposto, desenvolvido e apresentado no item 4.2 é utilizado um sinal com frequência de amostragem igual a 200 Hz, com mil amostras e contendo como patologia Taquicardia Sinusal e com frequência cardíaca de 120 BPM, conforme o sinal representado pela Figura 35. Após processamento, obtém se como resultado os picos simbolizados por pequenos círculos laranja, onde estes representa a posição dos picos R encontrados.

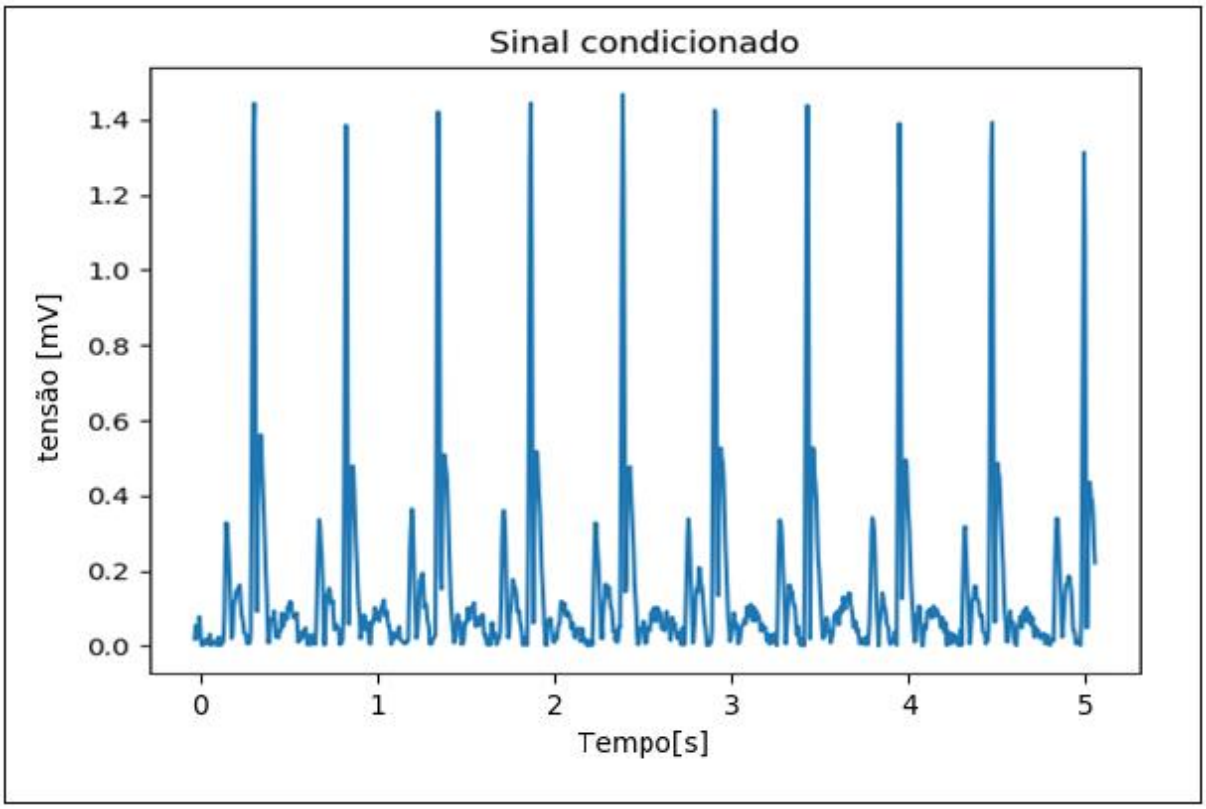
Figura 35 - Sinal cardíaco a ser analisado contendo como patologia Taquicardia Sinusal



Fonte: Autoria própria.

A Figura 36 representa a etapa de condicionamento do sinal para o sinal de teste contendo como patologia Taquicardia Sinusal.

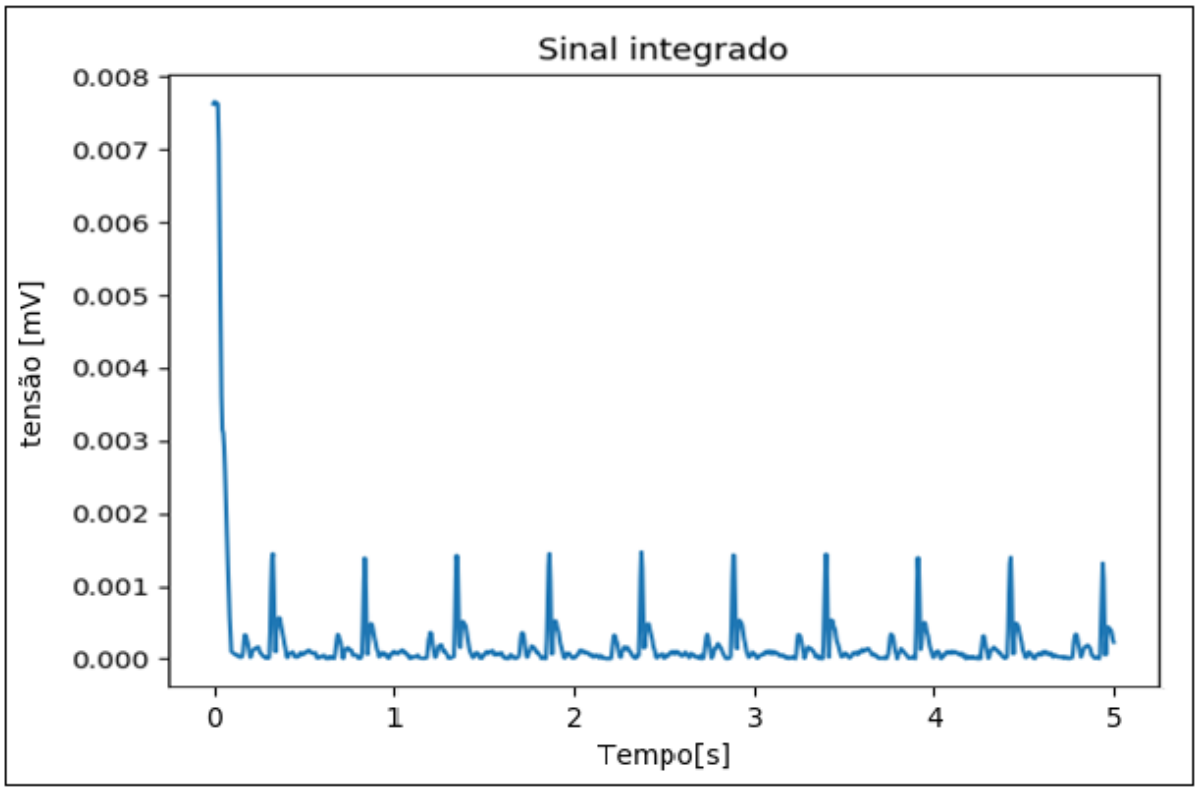
Figura 36 - Sinal condicionado



Fonte: Autoria própria.

A Figura 37 representa à resposta do *firmware* a passagem pelo filtro integrador de sinal, para o sinal de teste contendo como patologia Taquicardia Sinusal.

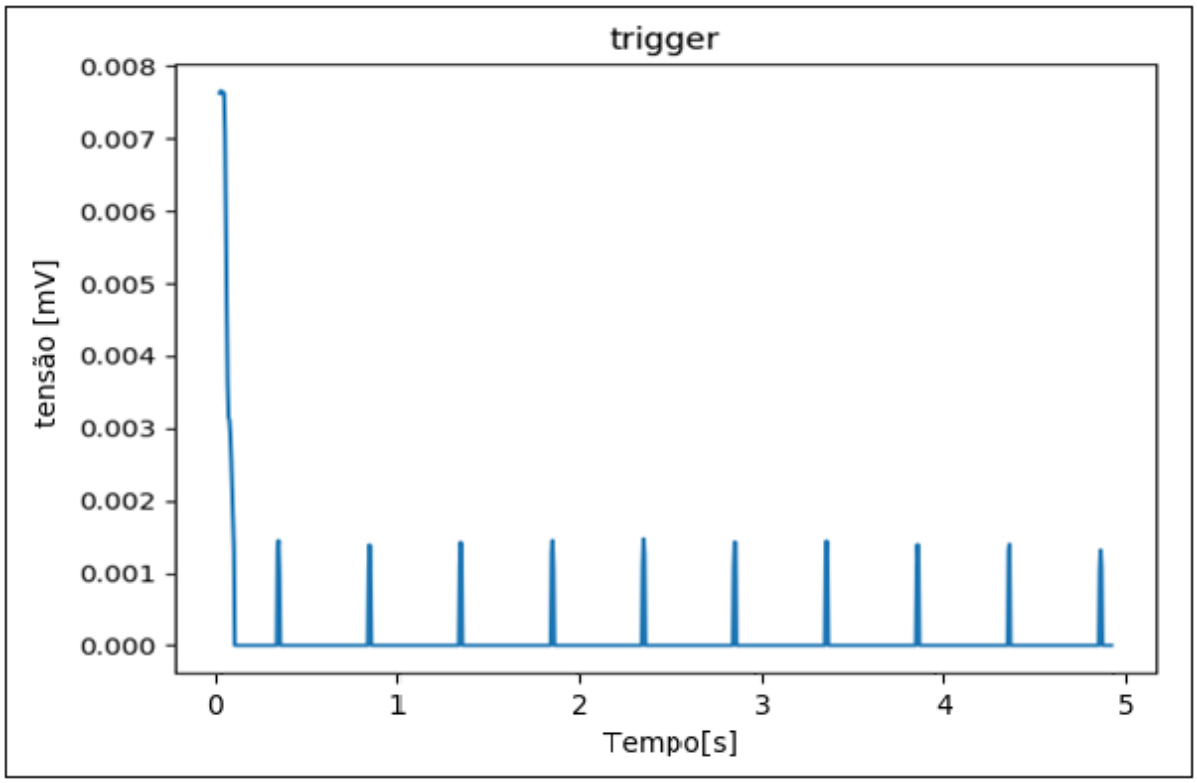
Figura 37 - Sinal integrado



Fonte: Autoria própria.

A Figura 38 representa à resposta do *firmware* a passagem pela função *trigger*, para o sinal de teste contendo como patologia Taquicardia Sinusal.

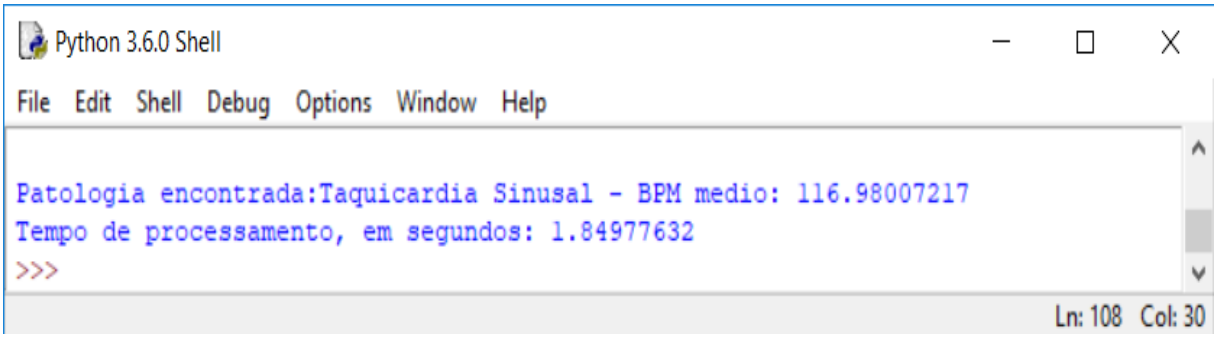
Figura 38 - Sinal com *trigger*



Fonte: Autoria própria.

A Figura 39 representa à resposta dada na janela Shell do Python com a patologia presente, que posteriormente é enviada ao servidor web, nela é possível à obtenção do valor da frequência cardíaca por minuto media, com valor igual 116,18 BPM.

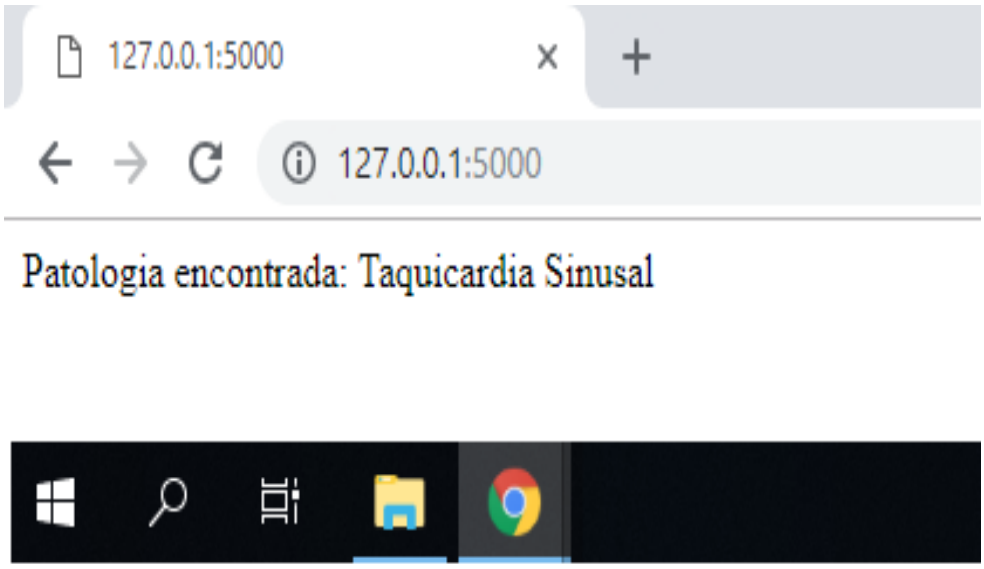
Figura 39 - Resposta do *firmware*



Fonte: Autoria própria.

A Figura 40 demostra o alerta apresentado via página *Web*, onde tal apresenta à patologia cardíaca encontrada no sinal analisado via *firmware.*

Figura 40 - Resposta do servido *Web*



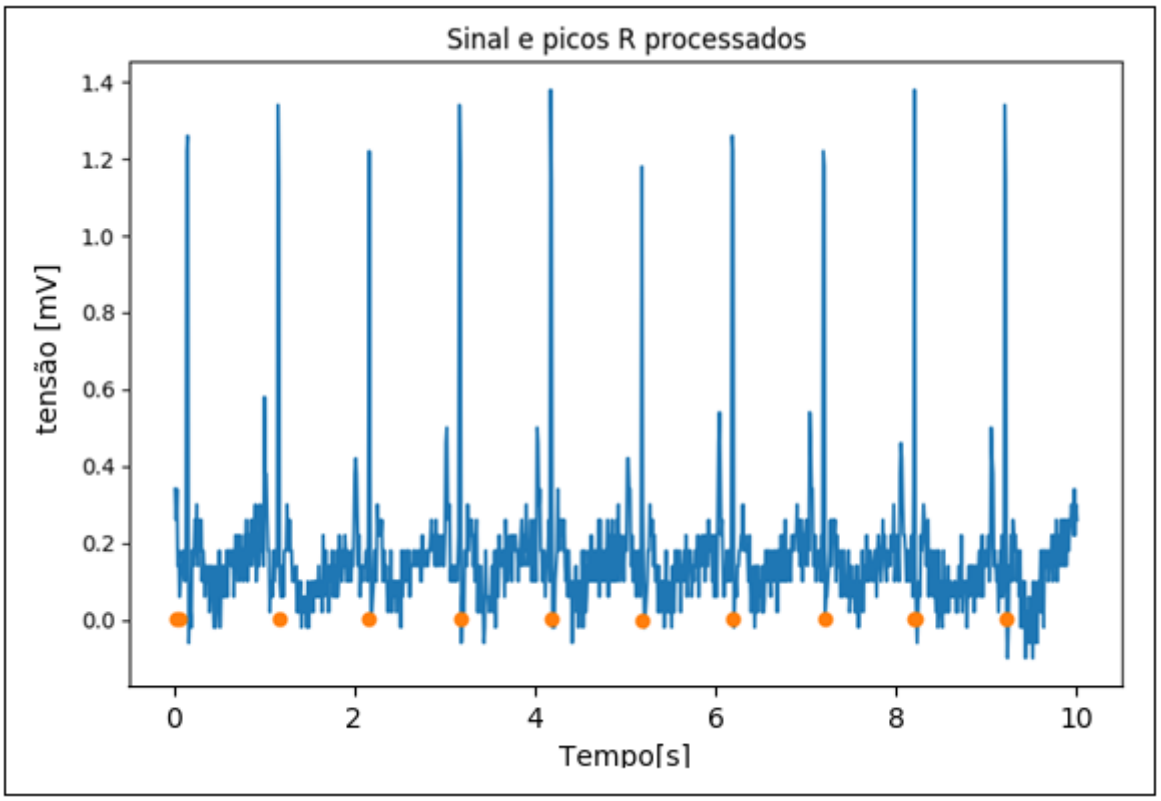
Fonte: Autoria própria.

Como observado após passagem do sinal pelo *firmware* proposto descrito nos itens anteriores, vemos que o firmware se comporta da forma como previsto, pois, era esperado encontrar como resposta para o sinal uma patologia cardíaca do tipo Taquicardia Sinusal, como também dar o inicio no servidor *Web* contendo a patologia encontrada, sendo, assim o sofware cumpriu seu objetivo quanto a analise de patoligas do tipo citado, comparando o valor real da frequência cardíaca de valor igual 120 BPM com o encontrado pelo *firmware* de valor 116,18 BPM, obtem-se um erro de aproximadamente 3,18%.

5.4 Teste firmware Bradicardia Sinusal

Com intuito de se averiguar a capacidade de processamento do *firmware* proposto, desenvolvido e apresentado no item 4.2 é utilizado um sinal com frequência de amostragem igual a 100 Hz, com mil amostras e contendo como patologia Bradicardia Sinusal e com frequência cardíaca de 60 BPM, conforme o sinal representado pela Figura 41. Após processamento, obtém se como resultado os picos simbolizados por pequenos círculos laranja, onde estes representa a posição dos picos R encontrados.

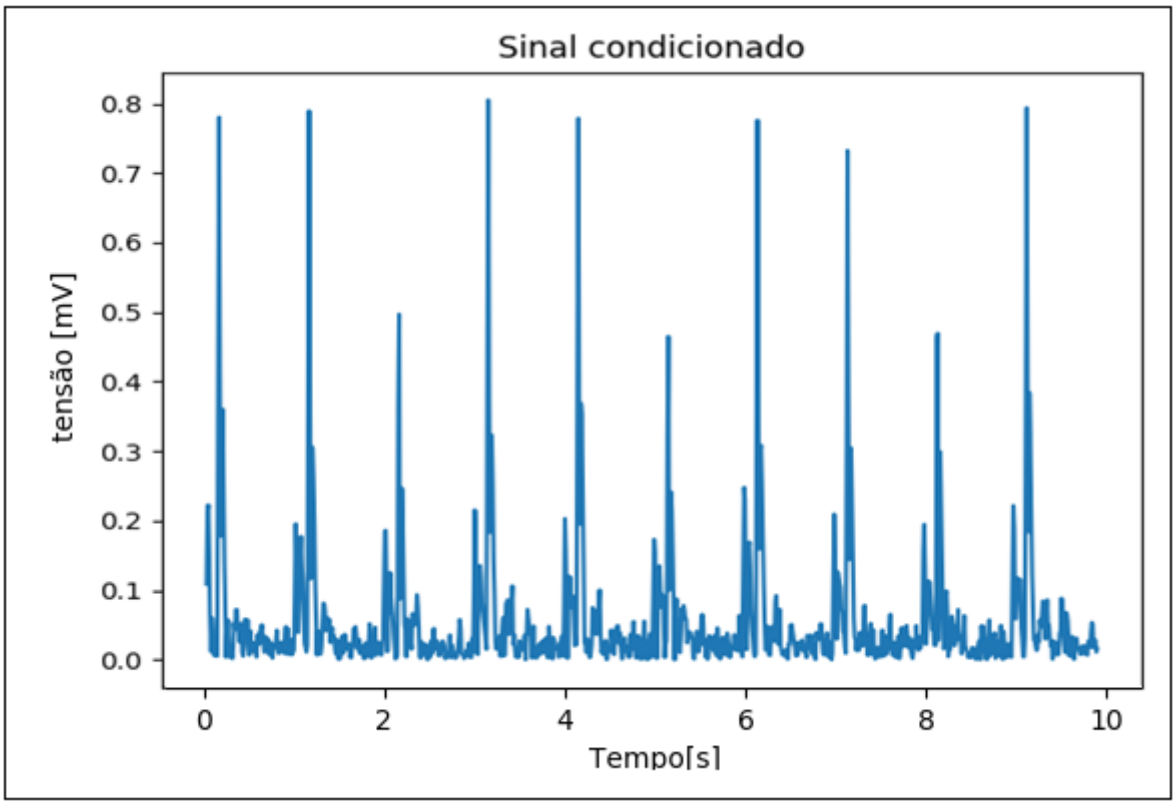
Figura 41- Sinal cardíaco a ser analisado contendo como patologia Bradicardia Sinusal



Fonte: Autoria própria.

A Figura 42 representa a etapa de condicionamento do sinal para o sinal de teste contendo como patologia Bradicardia Sinusal.

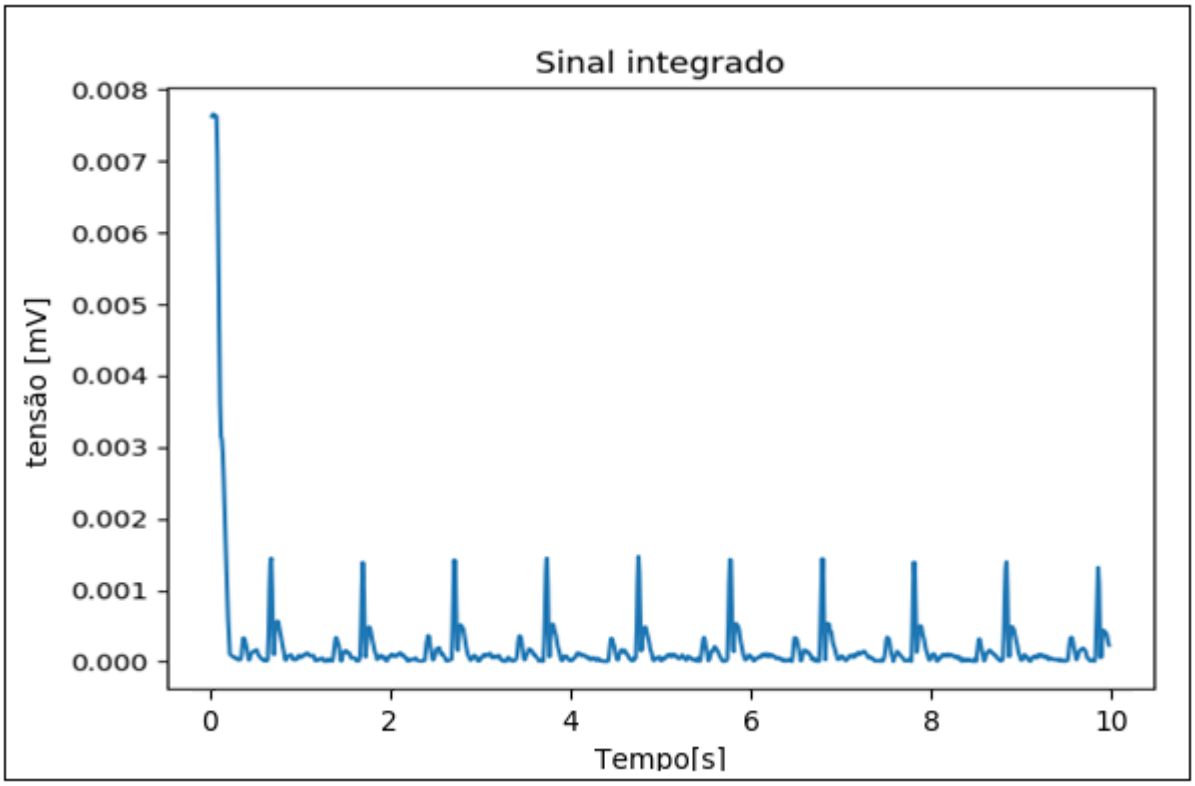
Figura 42 - Sinal condicionado



Fonte: Autoria própria.

A Figura 43 representa à resposta do *firmware* a passagem pelo filtro integrador de sinal, para o sinal de teste contendo como patologia Bradicardia Sinusal.

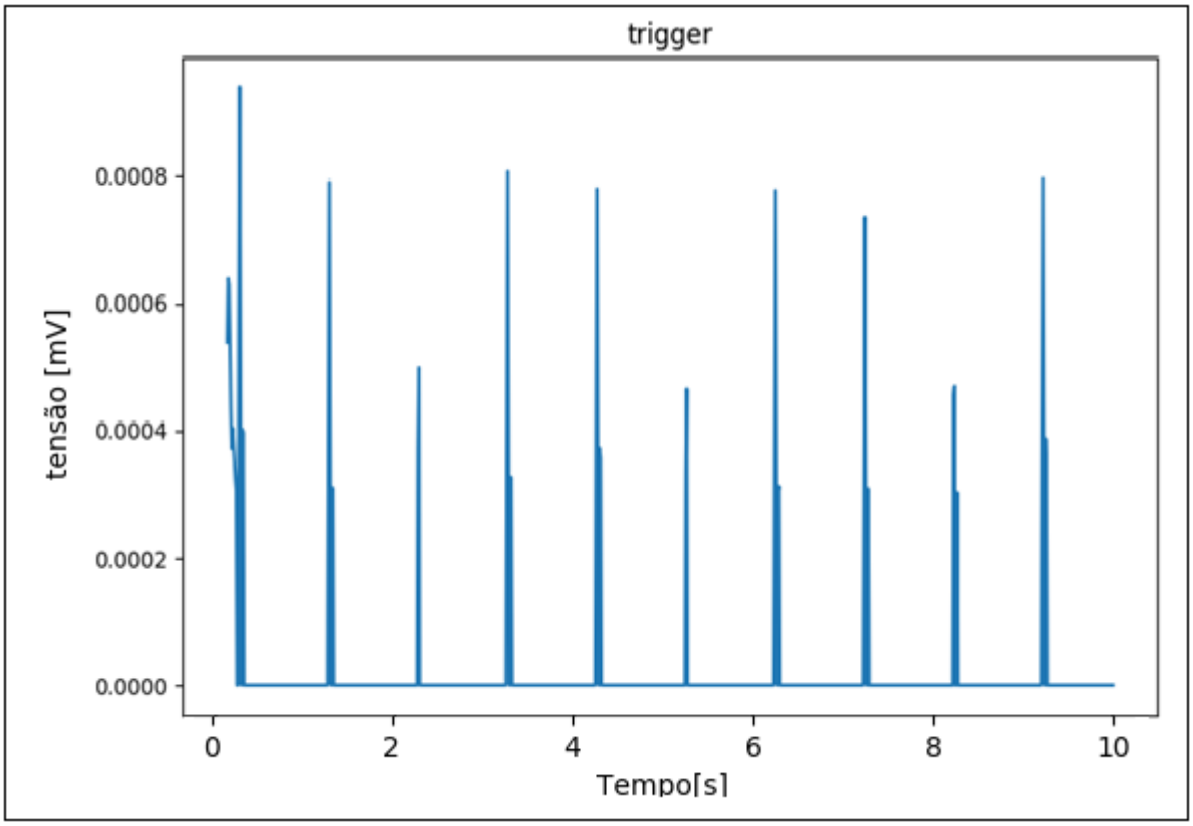
Figura 43 - Sinal integrado



Fonte: Autoria própria.

A Figura 44 representa à resposta do *firmware* a passagem pela função *trigger*, para o sinal de teste contendo como patologia Bradicardia Sinusal.

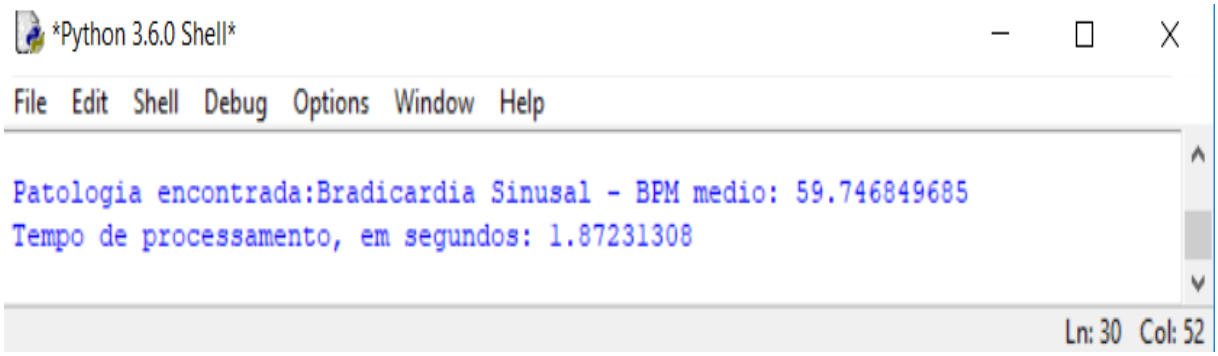
Figura 44 - Sinal com *trigger*



Fonte: Autoria própria.

A Figura 45 representa à resposta dada na janela Shell do Python com a patologia presente, que posteriormente é enviada ao servidor web, nela é possível a obtenção do valor da frequência cardíaca por minuto media, com valor igual 59,74 BPM.

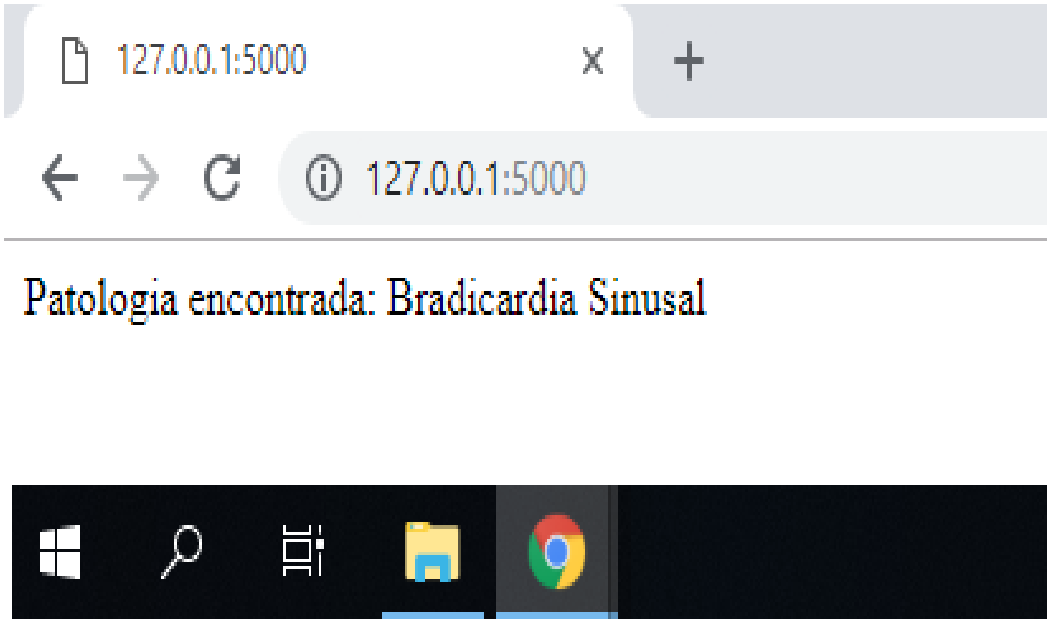
Figura 45 - Resposta do *firmware*



Fonte: Autoria própria.

A Figura 46 demostra o alerta apresentado via página *Web*, onde tal apresenta a patologia cardíaca encontrada no sinal analisado via *firmware.*

Figura 46 - Resposta do servido *Web*



Fonte: Autoria própria.

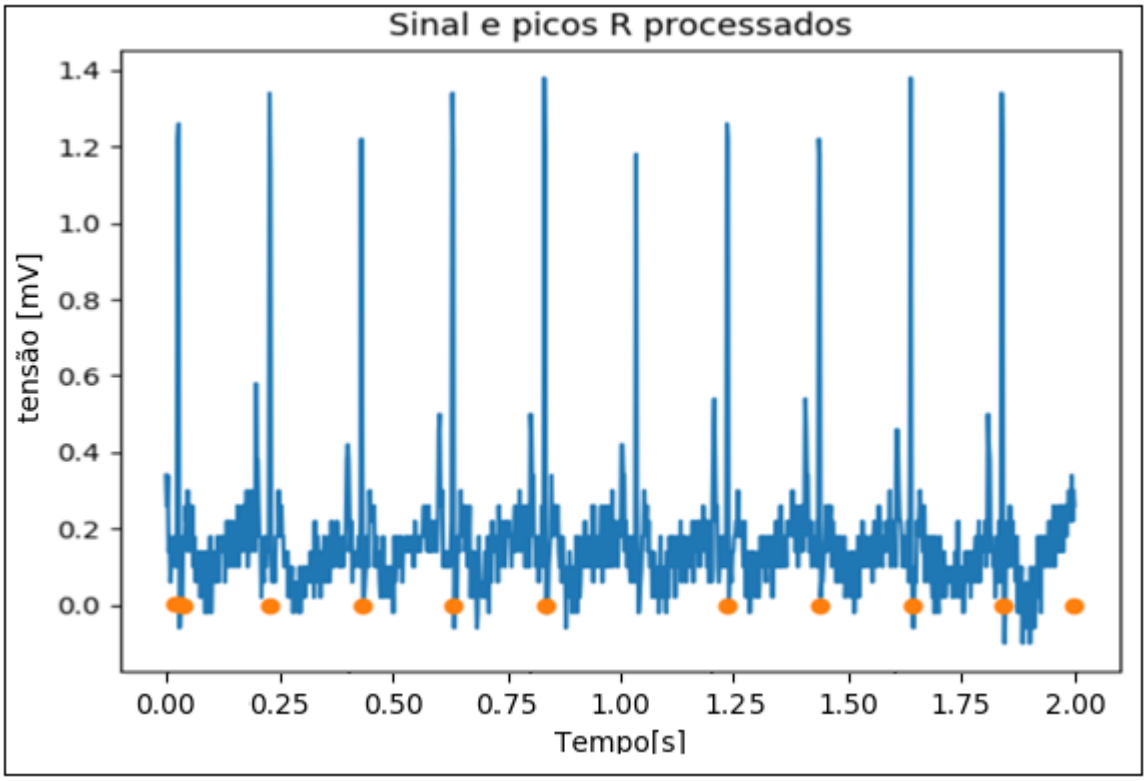
Como observado após passagem do sinal pelo *firmware* proposto descrito nos itens anteriores, vemos que o firmware se comporta da forma como previsto, pois, era esperado encontrar como resposta para o sinal uma patologia cardíaca do tipo Bradicardia Sinusal, como também dar o inicio no servidor *Web* contendo a patologia encontrada, sendo, assim o sofware cumpriu seu objetivo quanto a análise de patoligas do tipo citado, comparando o valor real da frequência cardíaca de valor igual 60 BPM com o encontrado pelo *firmware* de valor 59,74 BPM, obtem-se um erro de aproximadamente 0,43%.

5.5 Teste firmware Flutter Atrial

Com intuito de se averiguar a capacidade de processamento do *firmware* proposto, desenvolvido e apresentado no item 4.2 é utilizado um sinal com frequência de amostragem igual a 500 Hz, com mil amostras e contendo como patologia Flütter Atrial e com frequência cardíaca de 340 BPM, conforme o sinal representado pela Figura 47. Após processamento, obtém se como resultado os

picos simbolizados por pequenos círculos laranja, onde estes representa a posição dos picos R encontrados.

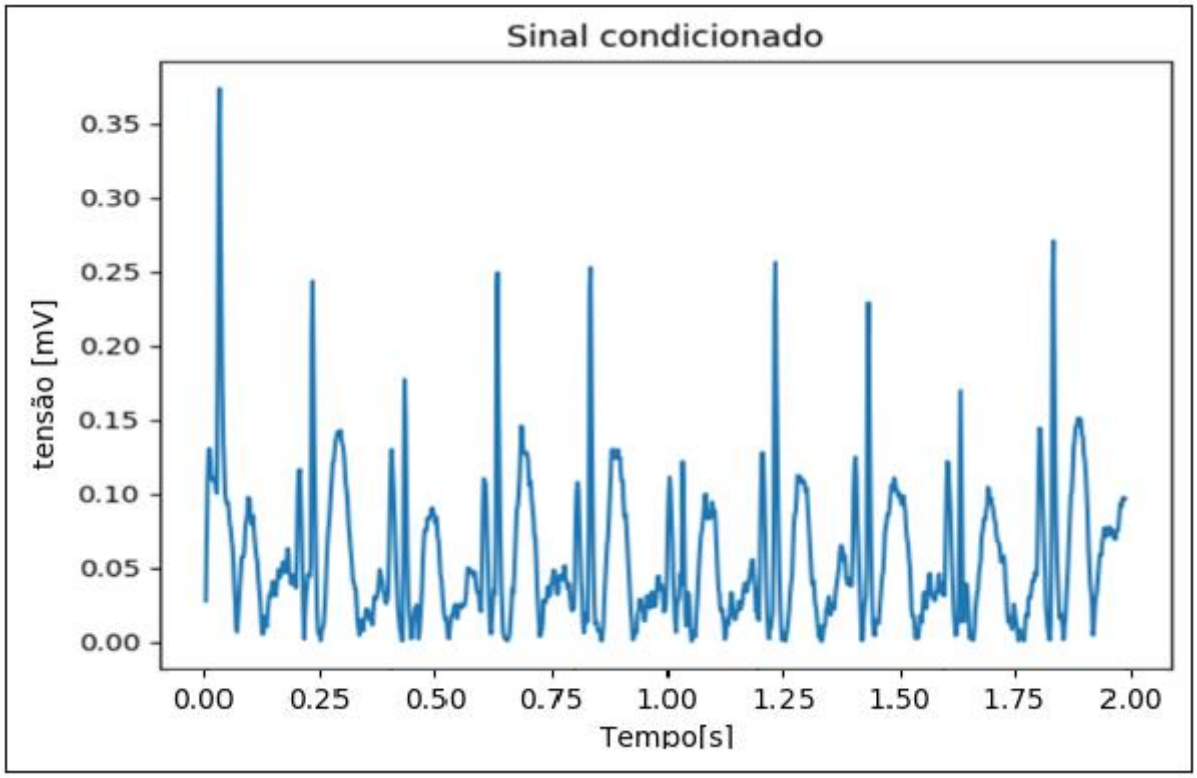
Figura 47 - Sinal cardíaco a ser analisado contendo como patologia Flutter Atrial



Fonte: Autoria própria.

A Figura 48 representa a etapa de condicionamento do sinal para o sinal de teste contendo como patologia Flütter Atrial.

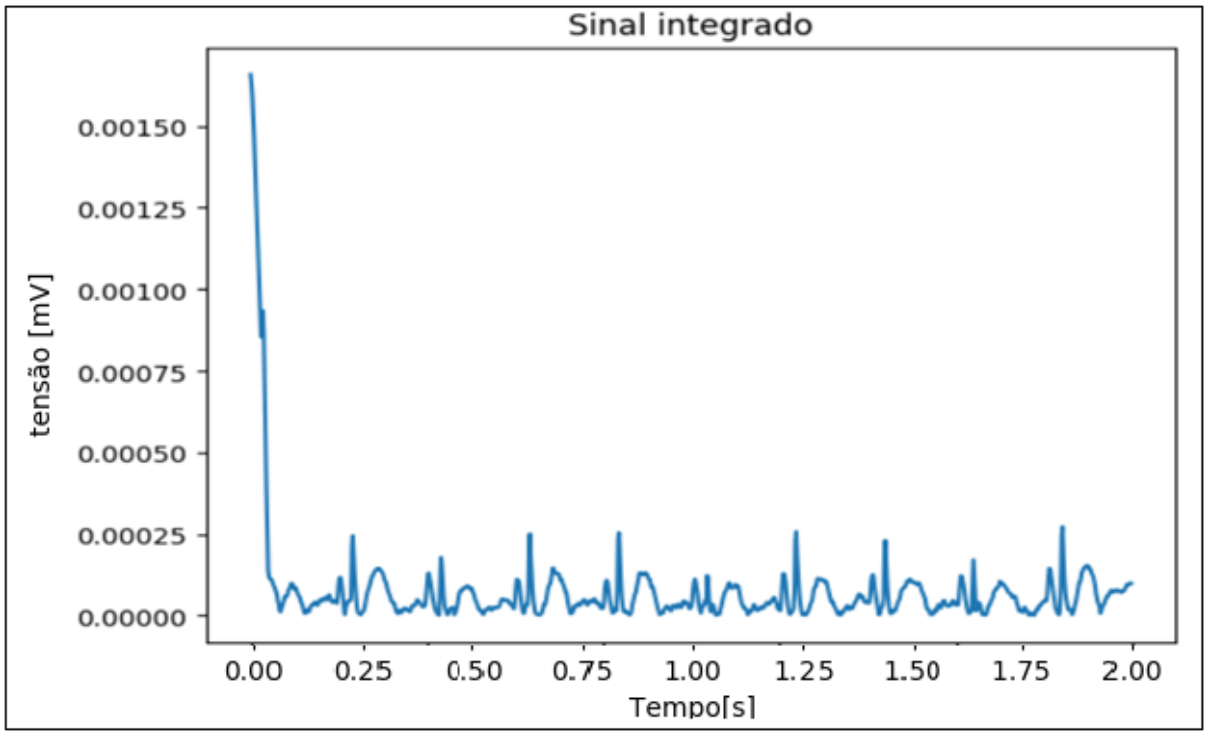
Figura 48 - Sinal condicionado



Fonte: Autoria própria.

A Figura 49 representa à resposta do *firmware* a passagem pelo filtro integrador de sinal, para o sinal de teste contendo como patologia Flütter Atrial.

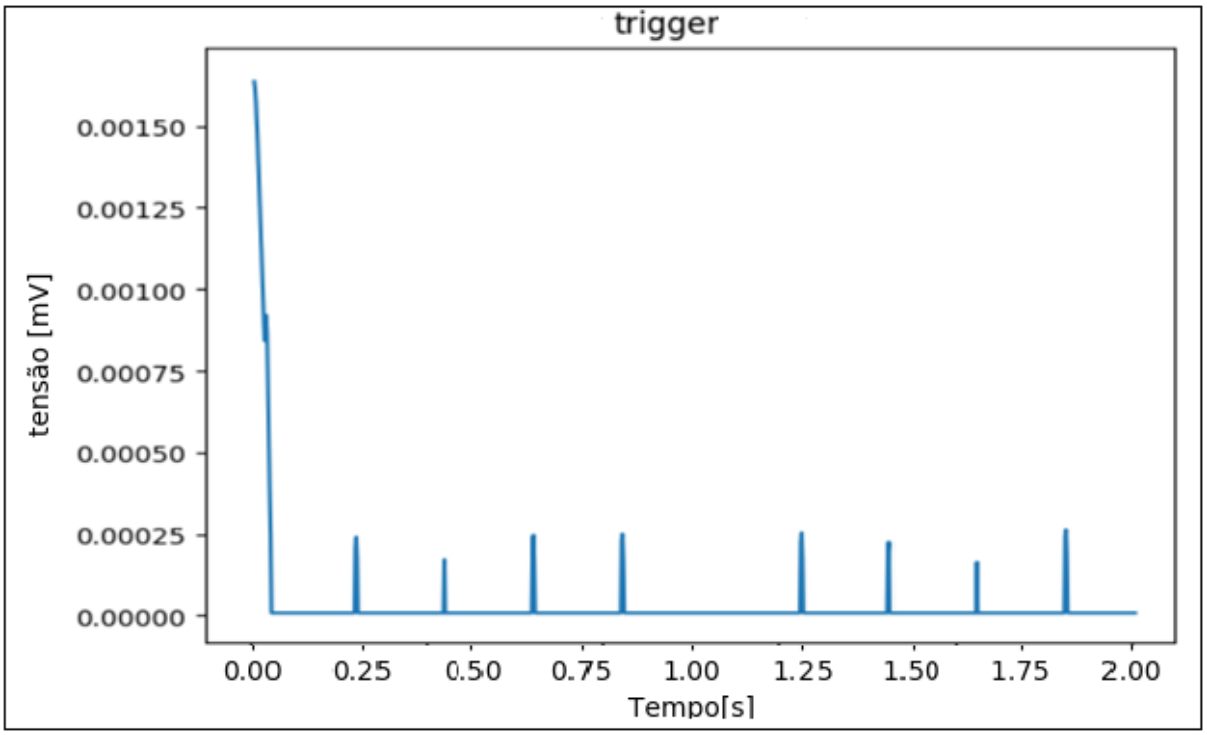
Figura 49 - Sinal integrado



Fonte: Autoria própria.

A Figura 50 representa à resposta do *firmware* a passagem pela função *trigger*, para o sinal de teste contendo como patologia Flütter Atrial.

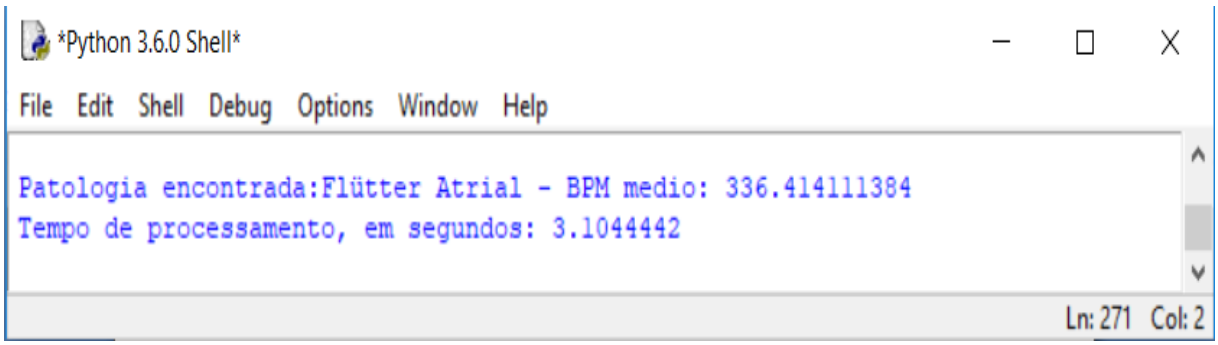
Figura 50 - Sinal com trigger



Fonte: Autoria própria.

A Figura 51 representa a resposta dada na janela Shell do Python com a patologia presente, que posteriormente é enviada ao servidor web, nela é possível à obtenção do valor da frequência cardíaca por minuto media, com valor igual 336,41 BPM.

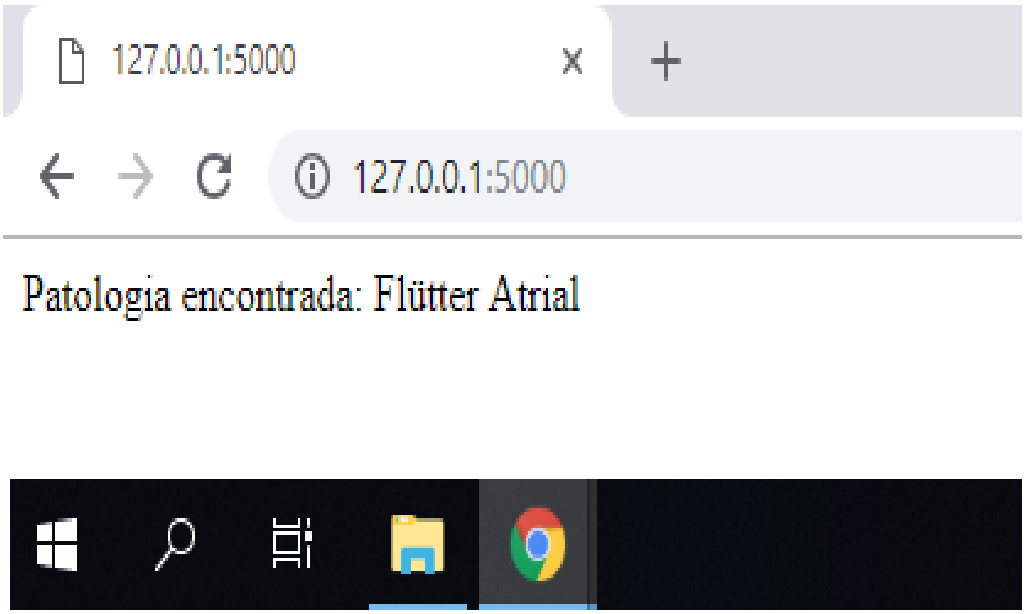
Figura 51 - Resposta do *firmware*



Fonte: Autoria própria.

A Figura 52 demostra o alerta apresentado via página Web, onde tal apresenta a patologia cardíaca encontrada no sinal analisado via *firmware*.

Figura 52 - Resposta do servido *Web*



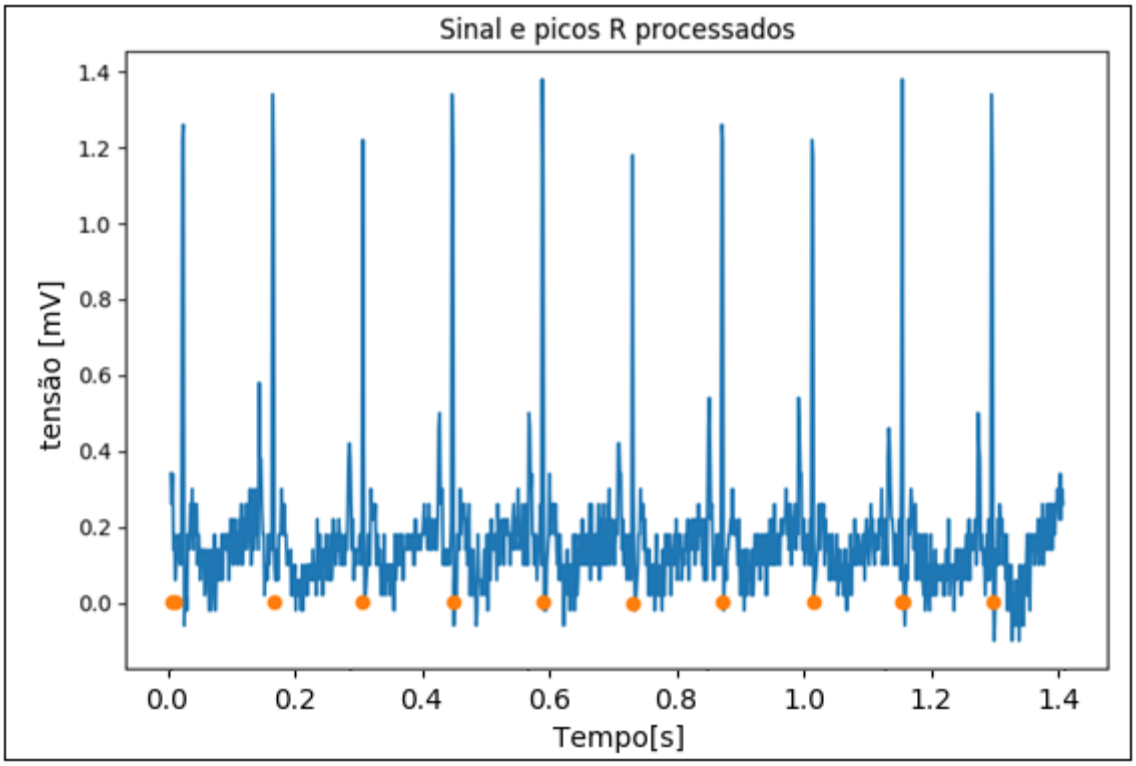
Fonte: Autoria própria.

Como observado após passagem do sinal pelo *firmware* proposto descrito nos itens anteriores, vemos que o firmware se comporta da forma como previsto, pois, era esperado encontrar como resposta para o sinal uma patologia cardíaca do tipo Flütter Atrial, como também dar o inicio no servidor *Web* contendo a patologia encontrada, sendo, assim o sofware cumpriu seu objetivo quanto a análise de patoliga do tipo citado, comparando o valor real da frequência cardíaca de valor igual 340 BPM com o encontrado pelo *firmware* de valor 336,41 BPM, obtem-se um erro de aproximadamente 1%.

5.6 Teste firmware Fibrilação Atrial

Com intuito de se averiguar a capacidade de processamento do *firmware* proposto, desenvolvido e apresentado no item 4.2 é utilizado um sinal com frequência de amostragem igual a 700 Hz, com mil amostras e contendo como patologia Fibrilação Atrial e com frequência cardíaca de 400 BPM, conforme o sinal representado pela Figura 53. Após processamento, obtém se como resultado os picos simbolizados por pequenos círculos laranja, onde estes representa a posição dos picos R encontrados.

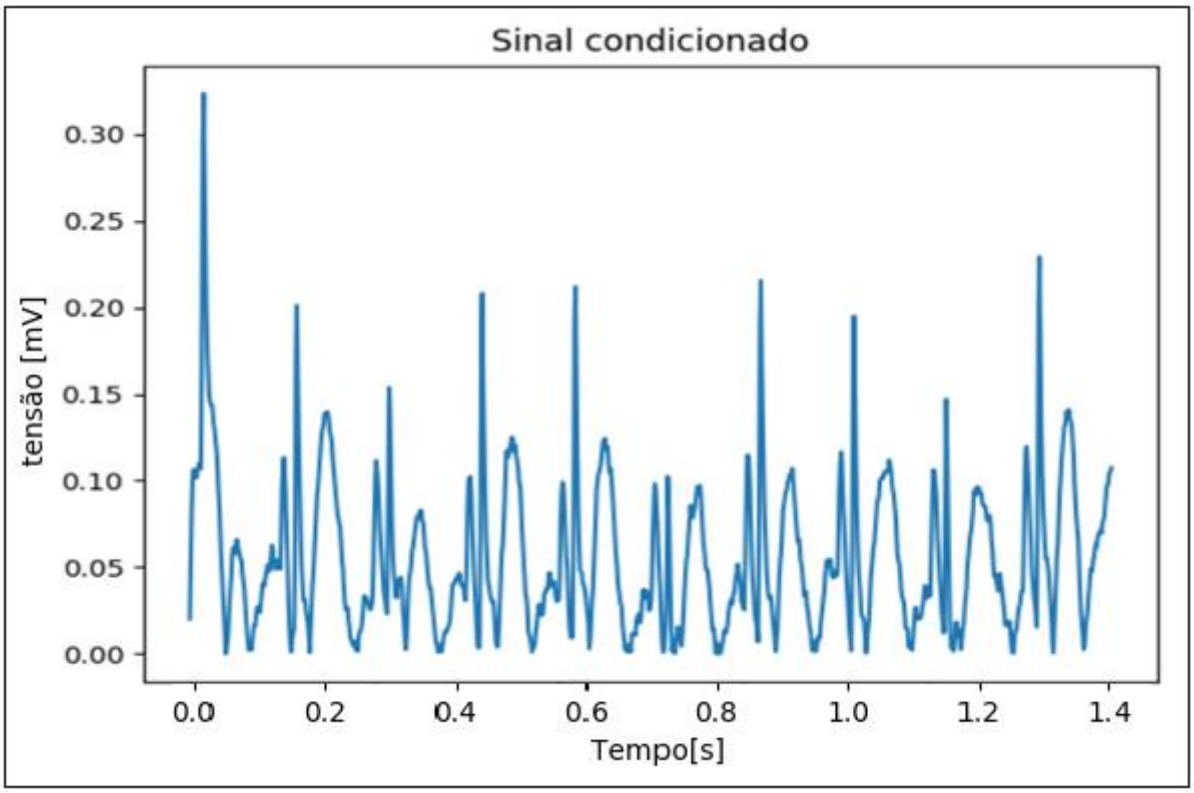
Figura 53 - Sinal cardíaco a ser analisado contendo como patologia Fibrilação Atrial



Fonte: Autoria própria.

A Figura 54 representa a etapa de condicionamento do sinal para o sinal de teste contendo como patologia Fibrilação Atrial.

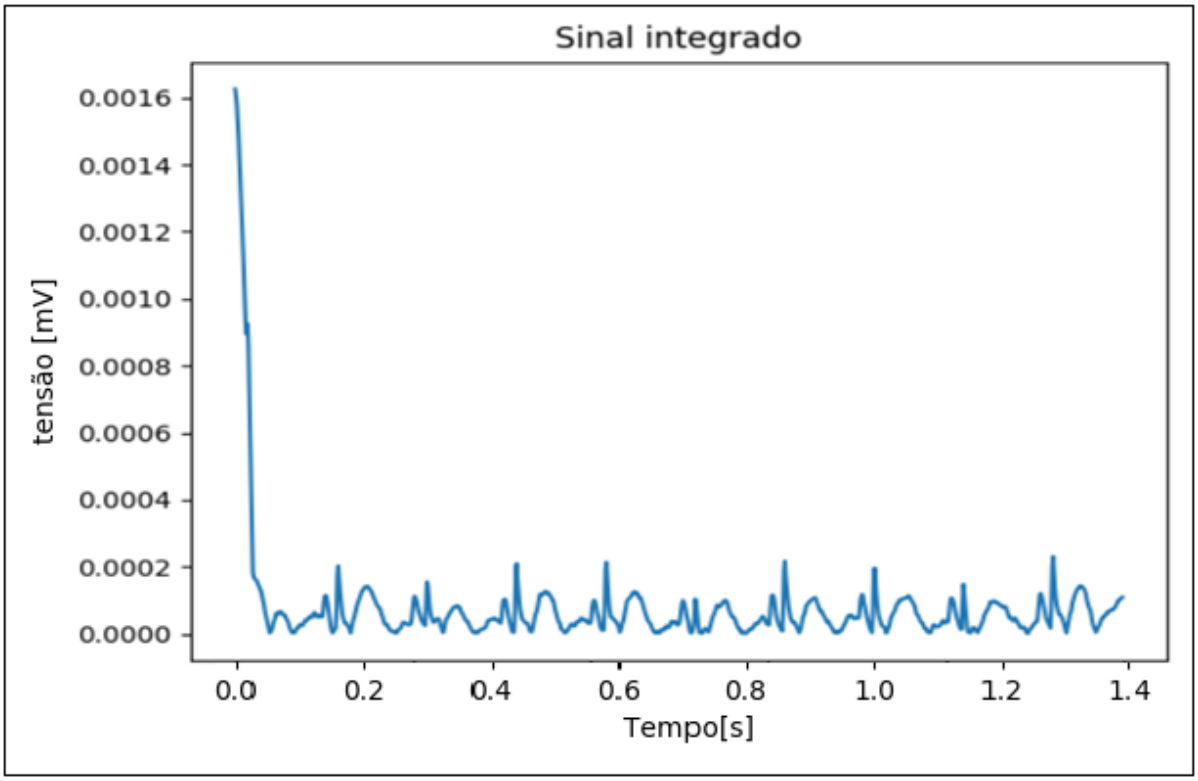
Figura 54: Sinal condicionado



Fonte: Autoria própria.

A Figura 55 representa à resposta do *firmware* a passagem pelo filtro integrador de sinal, para o sinal de teste contendo como patologia Fibrilação Atrial.

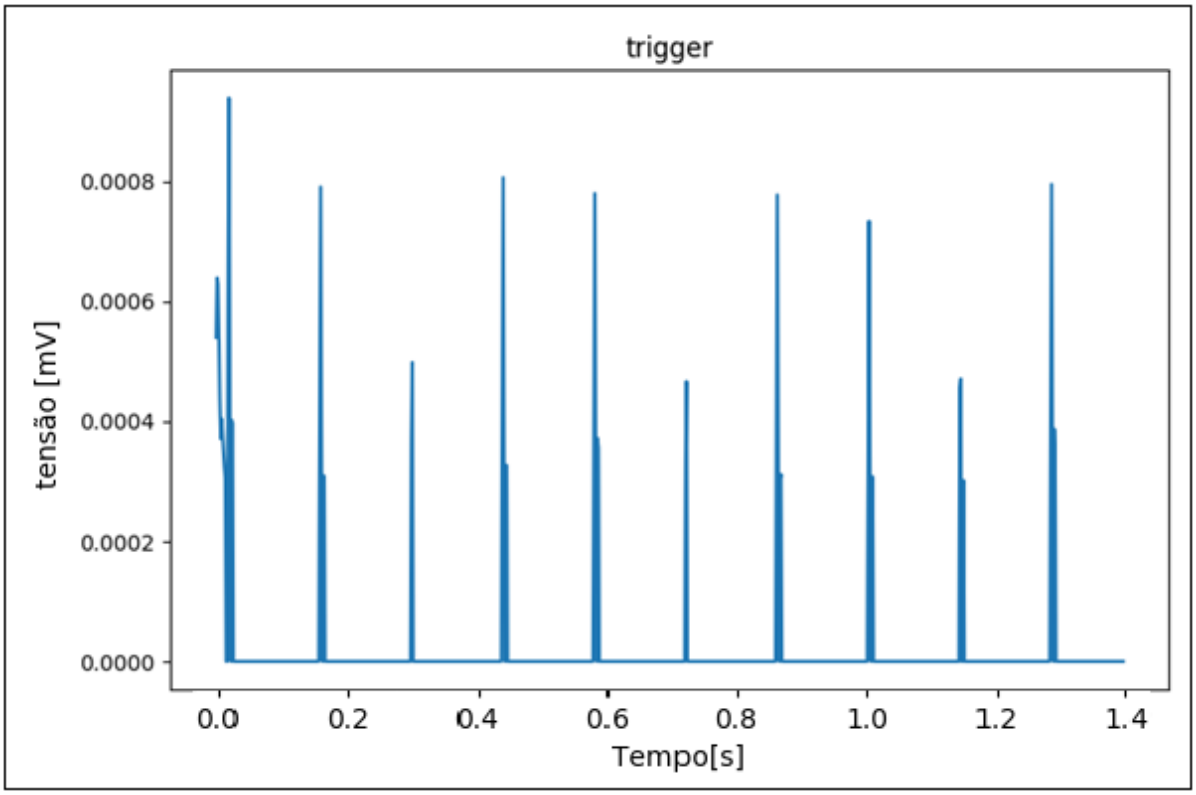
Figura 55: Sinal integrado



Fonte: Autoria própria.

A Figura 56 representa à resposta do *firmware* a passagem pela função trigger, para o sinal de teste contendo como patologia Fibrilação Atrial.

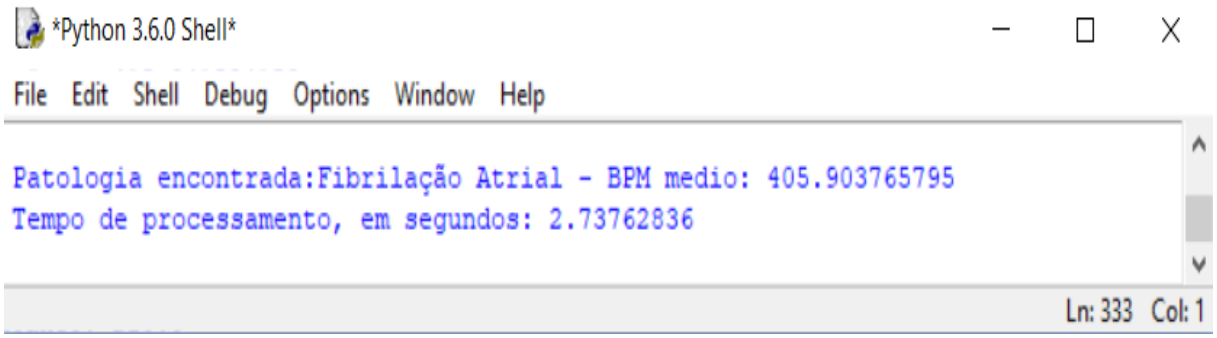
Figura 56 - Sinal com trigger



Fonte: Autoria própria.

A Figura 57 representa à resposta dada na janela Shell do Python com a patologia presente, que posteriormente é enviada ao servidor web, nela é possível à obtenção do valor da frequência cardíaca por minuto media, com valor igual 405,90 BPM.

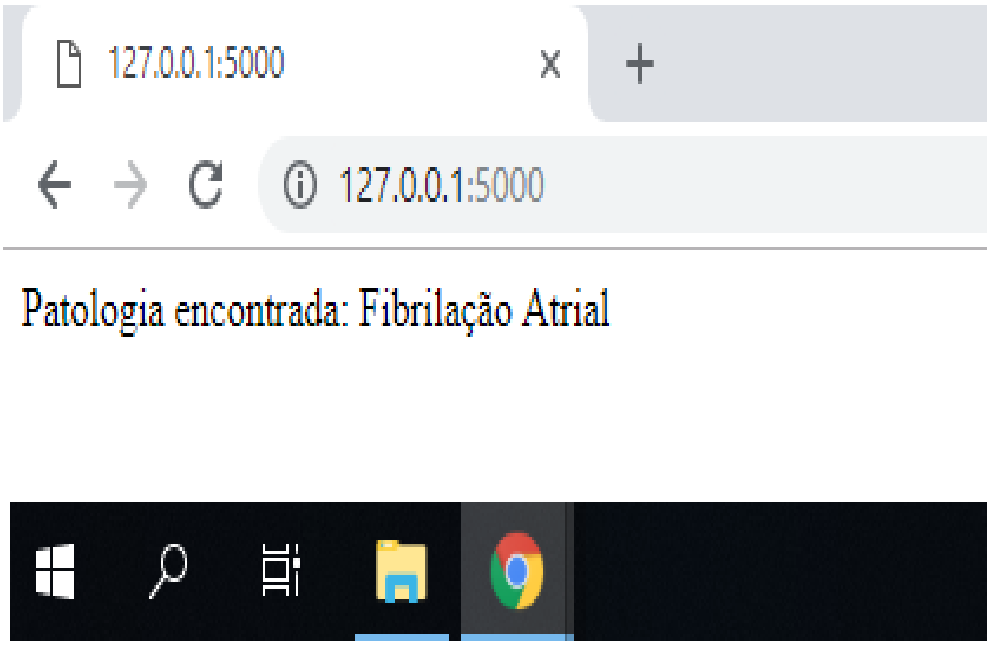
Figura 57 - Resposta do *firmware*



Fonte: Autoria própria.

A Figura 58 demostra o alerta apresentado via página Web, onde tal apresenta a patologia cardíaca encontrada no sinal analisado via *firmware.*

Figura 58 - Resposta do servido *Web*



Fonte: Autoria própria.

Como observado após passagem do sinal pelo *firmware* proposto descrito nos itens anteriores, vemos que o firmware se comporta da forma como previsto, pois, era esperado encontrar como resposta para o sinal uma patologia cardíaca do tipo Fibrilação Atrial, como também dar o inicio no servidor *Web* contendo a patologia encontrada, sendo, assim o sofware cumpriu seu objetivo quanto a análise de patoligas do tipo citado, comparando o valor real da frequência cardíaca de valor igual 400 BPM com o encontrado pelo *firmware* de valor 405,90 BPM, obtem-se um erro de aproximadamente 1,4%.

6 CONSIDERAÇÕES FINAIS

O maior interesse na elaboração deste trabalho foi possibilitar o estudo de meios para que o firmware de analises de patologias cardíacas opere no mundo real e realize o monitoramento remoto de sinais cardíacos, visando à criação de projetos que contribuam com o bem-estar da população.

No desenvolvimento deste projeto e na construção da proposta de trabalho, foi possível a verificação da implementação dos tratamentos matemáticos em sinais cardíacos, a partir do banco de dados contendo amostras de sinais cardíacos, com presença de patologias cardíacas, disponibilizado pelo professor e orientador André Sanches Fonseca Sobrinho, sendo estes processados pela linguagem Python de programação, viabilizando assim o desenvolvimento de um f*irmware* embarcado em um sistema eletrônico, sendo este o Raspberry Pi.

A contribuição deste trabalho consiste em estabelece uma base informacional, apresentar uma análise quantitativa dos dados oferecidos e no desenvolvimento do firmware embarcado em um sistema eletrônico capaz de realizar a análise de sinais elétricos cardíacos e disponibilizar, via servidor *Web,* um alerta sobre possíveis patologias encontradas nos sinais analisados, com base nos resultados foi se possível observar que o firmware é capaz de operar e encontrar patologias presentes nos sinais, com erro entre o valor real e obtido menores que 2%.

Com os resultados obtidos, mesmo que por simulação, espera-se ter contribuído em apresentar uma nova metodologia de aquisição e processamento de sinais eletrocardiográficos, visando a detecção de patologias cardíacas de maneira eficiente e rápida. Algumas propostas para trabalhos futuros são listadas a seguir:

- Integrar um circuito para aquisição de eletrocardiogramas, possibilitando desta forma a aplicação prática do sistema como um todo;
- Modificar os tipos de sinais bioelétricos de entrada;
- Incrementar novas análises para outras patologias;

- Utilizar de redes neurais artificiais que permitam a análise morfológica dos sinais eletrocardiográficos, possibilitando o estudo de uma nova variedade de patologias cardíacas que podem ser detectadas;
- Criar de uma interface gráfica para usuário para o acompanhamento eficiente de todo o processo de aquisição e detectar dos sinais eletrocardiográficos.

REFERÊNCIAS

[1] BORN, R. S. **Filtros Adaptativos aplicados a Sinais Bioelétricos.** Trabalho apresentado como requisito parcial à obtenção do título de Bacharel em Informática, UFPel, Pelotas, 2000. p. 1.

[2] FORTI, F. **Análise do Sinal Eletromiográfico em Diferentes Posicionamentos, Tipos de Eletrodos, Ângulos Articulares e Intencidades de Contração**. Universidade Metodista de Piracicaba. Piracicaba, 2005. p. 134.

[3] GUYTON; HALL. **Tratado de fisiologia médica.** Editora, 12. ed. 2011, cap.9, p. 107-135.

[4] MALCOLM S. THALER. **The only EKG BOOK You´ll Ever Need.** Editora MIK Medical Library. 5. ed. 2007.

[5] PAN, J., TOMPKINS, W. **A Real-Time QRS Detection Algorithm**. IEEE Transactions on biomedical Enginnering, Vol. BME-32, No. 3, March 1988.

[6] PAN, J., TOMPKINS, W. **Biomedical Digital Signal Processing**. Editora Prentice-Hall Vol. 1993.

[7] PORTAL Science museum. **Willem Einthoven.** Disponível em <http://www.sciencemuseum.org.uk/broughttolife/people/willemeinthoven>. Acesso em: 30 Abril 2017.

[8] GUYTON; HALL. **Tratado de fisiologia médica.** Editora Elsevier conhecimento sem fronteiras. Rio de janeiro, 12. ed. 2011, cap.13, p. 153-163.

[9] GUYTON; HALL. **Tratado de fisiologia médica.** Editora Elsevier conhecimento sem fronteiras. Rio de janeiro, 12. ed. 2011, cap.11, p. 129-135.

[10] CARNEIRO, E. F. **O Eletrocardiograma**. Editora Ateneu, Rio de Janeiro, 1987.

[11] WEBSTER, J. G. **Medical Intrumentation – Application and Design**. John Wiley & Sons, 1995.

[12] GUYTON; HALL. **Tratado de fisiologia médica.** Editora Elsevier conhecimento sem fronteiras. Rio de janeiro, 12. ed. 2011, cap.12, p. 139-150.

[13] PORTAL Lorver Pi**. Raspberry pi Banana pi Orange pi Odroid differences and chart**. Disponível em < https://www.loverpi.com/blogs/news/85588545-raspberry-pi-banana-pi-orange-pi-odroid-differences-and-chart>Acesso em: 30 Abril 2017.

[14] PORTAL Raspberry Pi. **About Us.** Disponível em <https://www.raspberrypi.org/about/ Acesso em: 30 Abril 2017.

[15] PORTAL Raspberry Pi. **Raspberry Pi 3 model b.** Disponível em <https://www.raspberrypi.org/products/raspberry-pi-3-model-b/>. Acesso em: 6 Maio 2017.

[16] PORTAL Raspberry Pi. **Downloads Raspberry Pi.** Disponível em <https://www.raspberrypi.org/downloads/raspbian//>. Acesso em: 30 Maio 2017.

[17] PORTAL Python. **Python.** Disponível em < https://www.python.org/about/help/>. Acesso em: 23 Abril 2017.

[18] Eric Hideki. **Resenha Python Cookbook**. Disponível em < https://ericstk.wordpress.com/about/>. Acesso em: 25 Abril 2017.

[19] PORTAL Flask. **Flask.** Disponível em < http://flask.pocoo.org/docs/0.12/>. Acesso em: 30 Abril 2017.

[20] PORTAL Scipy. **Scipy.** Disponível em < http://flask.pocoo.org/docs/0.12/>. Acesso em: 02 julho 2017.

[21] PORTAL Numpy. **Numpy.** Disponível em < http://www.numpy.org**/**>. Acesso em: 09 julho 2017.

[22] PORTAL Matplotlib. **Matplotlib.** Disponível em < https://matplotlib.org**/**>. Acesso em: 09 julho 2017.

[23] PORTAL Pandas. **Pandas.** Disponível em http://pandas.pydata.org/ >. Acesso em: 09 julho 2017.

[24] Analog. **Datasheet: ADAS1000**. Publicação eletrônica 2012-2015 Disponível em < http://www.analog.com/media/en/technical-documentation/data-sheets/ADAS1000-3_1000-4.pdf >. Acesso em: 6 Maio 2017.

[25] CARVALHO, J. L. A. **Análise Temporal e Espectral de Sinais de Variabilidade Cardíaca Usando Matlab.** Departamento de Engenharia Elétrica – UNB**.** Brasília, 2001.

[26] Analog. **Datasheet: ADAS1000**. Publicação eletrônica 2012-2015 Disponível em < https://docs.scipy.org/doc/scipy-0.14.0/reference/generated/scipy.signal.find_peaks_cwt.html>. Acesso em: 6 Maio 2017.

Brand	Raspberry Pi	Raspberry Pi	Raspberry Pi	Orange Pi	Orange Pi	Orange Pi	Orange Pi	Orange Pi	Orange Pi	Orange Pi
Board	Zero	2 Model B	3 Model B	One	Lite	PC	PC Plus	Plus	Plus 2e	Plus 2
SoC Vendor	Broadcom	Broadcom	Broadcom	Allwinner	Allwinner	Allwinner	Allwinner	Allwinner	Allwinner	Allwinner
SoC Chip	BCM2835	BCM2836	BCM2837	H3	H3	H3	H3	H3	H3	H3
SoC Process	40nm	40nm	40nm	40nm	40nm	40nm	40nm	40nm	40nm	40nm
CPU Cores	1	4	4	4	4	4	4	4	4	4
CPU Design	ARM1176JZF-S	Cortex-A7	Cortex-A53	Cortex-A7	Cortex-A7	Cortex-A7	Cortex-A7	Cortex-A7	Cortex-A7	Cortex-A7
CPU Freq	1GHz	0.9GHz	1.2GHz	1.2GHz	1.2GHz	1.3GHz	1.3GHz	1.3GHz	1.3GHz	1.3GHz
CPU Instruction	ARMv6	ARMv7	ARMv8	ARMv7	ARMv7	ARMv7	ARMv7	ARMv7	ARMv7	ARMv7
GPU Vendor	Broadcom	Broadcom	Broadcom	ARM	ARM	ARM	ARM	ARM	ARM	ARM
GPU Design	VideoCore IV	VideoCore IV	VideoCore IV	Mali-400 MP2	Mali-400 MP2	Mali-400 MP2	Mali-400 MP2	Mali-400 MP2	Mali-400 MP2	Mali-400 MP2
GPU Freq	250MHz	250MHz	400MHz	600MHz	600MHz	600MHz	600MHz	600MHz	600MHz	600MHz
H264 Dec	1080P30	1080P30	1080P60	1080P60	1080P60	1080P60	1080P60	1080P60	1080P60	1080P60
H264 Enc	1080P30	1080P30	1080P30	1080P30	1080P30	1080P30	1080P30	1080P30	1080P30	1080P30
H265 Dec	None	None	None	4K30	4K30	4K30	4K30	4K30	4K30	4K30
H265 Enc	None	None	None	None	None	None	None	None	None	None
VP9	None	None	None	None	None	None	None	None	None	None
Memory	512MB DDR2	1GB DDR2	1GB DDR2	512MB DDR3	512MB DDR3	1GB DDR3	1GB DDR3	1GB DDR3	2GB DDR3	2GB DDR3
Memory Freq	400MHz	400MHz	450MHz	672MHz	672MHz	672MHz	672MHz	672MHz	672MHz	672MHz
Storage Expandable	MicroSD	MicroSD	MicroSD	MicroSD	MicroSD	MicroSD	MicroSD	MicroSD/USB SATA 2.0	MicroSD	MicroSD/USB SATA 2.0
Storage Onboard	None	None	None	None	None	None	8GB eMMC	8GB eMMC	16GB eMMC	16GB eMMC
USB 2.0	1 OTG	4 (Shared BW)	4 (Shared BW)	1 + 1 OTG	2 + 1 OTG	3 + 1 OTG	3 + 1 OTG	4 + 1 OTG (Shared BW)	4 + 1 OTG	4 + 1 OTG (Shared BW)
USB 3.0	0	0	0	0	0	0	0	0	0	0
Ethernet	None	100Mb	100Mb	100Mb	100Mb	100Mb	100Mb	1Gb Realtek RTL8211E	1Gb Realtek RTL8211E	1Gb Realtek RTL8211E
Wireless	None	None	802.11N BCM43438	None	802.11N RTL8189FTV	None	802.11N RTL8189FTV	802.11N RTL8189ETV	802.11N RTL8189FTV	802.11N RTL8189ETV
Bluetooth	None	None	Bluetooth 4.1	None	None	None	None	None	None	None
IR	None	None	None	None	Yes	Yes	Yes	Yes	Yes	Yes
Microphone	None	None	None	None	Yes	Yes	Yes	Yes	Yes	Yes
HDMI	1080P60	1080P60	1080P60	4K30	4K30	4K30	4K30	4K30	4K30	4K30
RTC	No	No	No	No	No	No	No	No	No	No
Power Req	5V1A	5V2A	5V2.4A	5V2A	5V2A	5V2A	5V2A	5V2A	5V2A	5V2A
Power Plug uUSB	Yes	Yes	Yes	No	No	No	No	No	No	No
Power Plug Barrel	No	No	No	4/1.7mm	4/1.7mm	4/1.7mm	4/1.7mm	4/1.7mm	4/1.7mm	4/1.7mm
Android	Maybe	Maybe	Maybe	5.1	5.1	5.1	5.1	5.1	5.1	5.1
Linux Mainline	Yes	Yes	Yes	In Progress	In Progress	In Progress	In Progress	In Progress	In Progress	In Progress
Distro	Raspbian	Raspbian	Raspbian	Armbian	Armbian	Armbian	Armbian	Armbian	Armbian	Armbian
US Price	Vaporware	$37.99	$37.99	$16.99	$18.99	$22.99	$28.99	$44.99	$44.99	$59.99
The MicroUSB connector was designed to support maximum of 1.8A of current. 2.4A exceeds this specification and poses a safety hazard.										
Raspberry Pi and Orange Pi boards are not drop in replacements for each other. Operating system images contain different bootloaders and kernels along with different drivers.										

Apêndice A

```
import numpy as np

import pandas as pd
import matplotlib.pyplot as plt

import matplotlib.pyplot as plt

import matplotlib.animation as animation

import time

from   flask import Flask

from   array import array

from   scipy import signal

from   numpy import cos, sin, pi, absolute, arange, array, append, abs, angle,\

                fft, linspace, zeros, log10, unwrap

from   scipy.signal  import kaiserord, lfilter, firwin, freqz, butter, lfilter,find_peaks_cwt

from   pylab import figure, clf, plot, xlabel, ylabel, xlim, ylim, title, grid, \

                axes, show, subplot, axis, plot

from math import sqrt



def butter_bandpass(lowcut, highcut, fs, order=2):

    nyq = 0.5 * fs

    low = lowcut / nyq

    high = highcut / nyq

    b, a = butter(order, [low, high], btype='band')
```

```
    return b, a


def butter_bandpass_filter(data, lowcut, highcut, fs, order=2):
    b, a = butter_bandpass(lowcut, highcut, fs, order=order)
    y = lfilter(b, a, data)
    return y


def load(nome):                    # Função de leitura arquivo .txt
  arq = open(nome, 'r')            # Abrir o arquivo
  x = array([])                      # Cria vetor vazio
  for linha in arq:                  # Leitura linha a linha de todo o arquivo
      x = append(x,float(linha))   # Atribui os valores da linha a cada ponto do vetor
  arq.close()                        # Fecha o arquivo


  return x                  # retorna x


###CODIGO PRINCIPAL
fs = int(input("Qual Frequencia de amostragem do sinal ECG(Hz)? "))  # Frequência de amostragem do sinal ECG (Hz)
sine = load('BradicardiaSinusal1000.amostras100Hz.txt')  # Chama a função de leitura e carrega o vetor x com as amostras do arquivo teste.txt
Ts =1./fs                          # Tempo de amostragem
lowcut = 3                         # Frequência baixa de corte
```

```
highcut = 17                        # Frequência alta de corte



#filtro passa faixa

y = butter_bandpass_filter(sine.data, lowcut, highcut, fs, order=1)



plt.figure(1)

plt.clf()

plt.plot(range(len(sine)),sine, range(len(y)),y)  # Plota o sinal original e filtrado

plt.title('Sinal filtrado')

plt.show()
```

Apêndice B

```
import numpy as np

import pandas as pd

from scipy import signal

import matplotlib.pyplot as plt

import matplotlib.pyplot as plt

import matplotlib.animation as animation



from numpy import cos, sin, pi, absolute, arange, array, append, abs, angle,\

                fft, linspace, zeros, log10, unwrap

from scipy.signal  import kaiserord, lfilter, firwin, freqz, butter, lfilter

from pylab import figure, clf, plot, xlabel, ylabel, xlim, ylim, title, grid, \

                axes, show, subplot, axis, plot



def sine_generator(fs, sinefreq, duration):

    T = duration

    nsamples = fs * T*10

    w = 2. * np.pi * sinefreq

    t_sine = np.linspace(0, T, nsamples, endpoint=False)

    y_sine = np.sin(w * t_sine)

    result = pd.DataFrame({

        'data' : y_sine} ,index=t_sine)
```

```
    return result


def butter_bandpass(lowcut, highcut, fs, order=3):
    nyq = 0.5 * fs
    low = lowcut / nyq
    high = highcut / nyq
    b, a = butter(order, [low, high], btype='band')
    return b, a



def butter_bandpass_filter(data, lowcut, highcut, fs, order=3):
    b, a = butter_bandpass(lowcut, highcut, fs, order=order)
    y = lfilter(b, a, data)

    return y


fps = 100


#Gerando sinal de 100Hz
sine_fq = 100 #Hz
duration = 100 #seconds
sine_100Hz = sine_generator(fps,sine_fq,duration)
```

```
#Gerando sinal de 50Hz

sine_fq = 50 #Hz

duration = 100 #seconds

sine_50Hz = sine_generator(fps,sine_fq,duration)


#Gerando sinal de 17Hz

sine_fq = 17 #Hz

duration = 100 #seconds

sine_17Hz = sine_generator(fps,sine_fq,duration)


#Gerando sinal de 10Hz

sine_fq = 10 #Hz

duration = 100 #seconds

sine_10Hz = sine_generator(fps,sine_fq,duration)


#Gerando sinal de 1Hz

sine_fq = 1 #Hz

duration = 100 #seconds

sine_1Hz = sine_generator(fps,sine_fq,duration)


sine = sine_100Hz + sine_50Hz + sine_17Hz + sine_10Hz + sine_1Hz
```

```
lowcut = 15

highcut = 18


y = butter_bandpass_filter(sine.data, lowcut, highcut, fps, order=3)

plt.figure(1)

plt.clf()

plt.plot(range(len(sine_17Hz)),sine_17Hz)

plt.title('Sinal de 17Hz gerado')


plt.figure(2)

plt.clf()

plt.plot(range(len(sine)),sine)

plt.title('Sinal a ser filtrado')


plt.figure(3)

plt.clf()

plt.plot(range(len(y)),y)

plt.title('Sinal filtrado')

plt.show()
```

Apêndice C

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib.pyplot as plt
import matplotlib.animation as animation
import time
import io
import base64
from   array import array
from   scipy import signal
from   numpy import cos, sin, pi, absolute, arange, array, append, abs, angle,\
                fft, linspace, zeros, log10, unwrap
from   scipy.signal  import kaiserord, lfilter, firwin, freqz, butter,
lfilter,find_peaks_cwt
from   pylab import figure, clf, plot, xlabel, ylabel, xlim, ylim, title, grid, \
                axes, show, subplot, axis, plot
from math import sqrt
from flask import Flask
from flask import render_template
import matplotlib.pyplot as plt
```

```
app = Flask(__name__)



def butter_bandpass(lowcut, highcut, fs, order=2):
    nyq = 0.5 * fs
    low = lowcut / nyq
    high = highcut / nyq
    b, a = butter(order, [low, high], btype='band')
    return b, a


def butter_bandpass_filter(data, lowcut, highcut, fs, order=2):
    b, a = butter_bandpass(lowcut, highcut, fs, order=order)
    y = lfilter(b, a, data)
    return y


def load(nome):                # Funçao de leitura arquivo .txt
  arq = open(nome, 'r')        # Abrir o arquivo
  x = array([])                # Cria vetor vazio
  for linha in arq:            # Leitura linha a linha de todo o arquivo
     x = append(x,float(linha))# Atribui os valoes da linha a cada ponto do
vetor
  arq.close()                  # Fecha o arquivo
```

```
        return x                # retorna x


    ###CODIGO PRINCIPAL

    fs = int(input("Qual Frequencia de amostragem do sinal ECG(Hz)? "))# Frequencia de amostragem do sinal ECG (Hz)

    t0 = time.clock()                # Inicia contagem de tempo

    sine = load('DI60-1000.amostras-.txt')# Chama a funçao de leitura e carrega o vetor x

                                # Com as amostras do arquivo teste.txt

    Ts =1./fs                    # Tempo de amostragem

    lowcut = 3                    # Frequencia baixa de corte

    highcut = 17                    # Frequencia alta de corte


    #filtro passa faixa


    y = butter_bandpass_filter(sine.data, lowcut, highcut, fs, order=1)


    #condicionador de sinal


    for i in range(len(y)):

            y[i]=sqrt(y[i]*y[i])

    plt.figure(2)

    plt.clf()
```

```
        plt.plot(range(len(y)),y)
        plt.title('Sinal condicionado')


        #integrador


        for i in range(len(y)):
                y[i]=(((y[i])+(y[i-1])+(y[i-2])+(y[i-3])+(y[i-4])+(y[i-5])+(y[i-6])+(y[i-7])+(y[i-8])+(y[i-9])+(y[i-10])+(y[i-11])+(y[i-12])+(y[i-13])+(y[i-14])+(y[i-15])+(y[i-16])+(y[i-17])+(y[i-18])+(y[i-19])+(y[i-20]))/len(y))
        plt.figure(3)
        plt.clf()
        plt.plot(range(len(y)),y)
        plt.title('Sinal integrado')


        #trigger


        for i in range(len(y)):


            if(y[i]<=0.0003):
                y[i]=0
        plt.figure(4)
        plt.clf()
        plt.plot(range(len(y)),y)
```

```
plt.title('trigger')


#detecção dos picos R


peakinds = find_peaks_cwt(y, np.arange(Ts,Ts*(len(y))))


#calculo dos batimentos cardiacos


bpmm=0

bpm=0

for i in range(len(peakinds)):


    if(peakinds[i]- peakinds[i-1]>=50):


        bpm = (60/((peakinds[i]- peakinds[i-1])*Ts))

        print('')

        print('BPM instantâneo:',bpm)

        bpmm = ((bpm + bpmm)/2)


plt.figure(1)

plt.clf()

plt.plot(range(len(sine)),sine)
```

```
plt.title('Sinal cardiaco e picos R processados')
plt.plot(peakinds, y[peakinds], 'o')
plt.savefig('Sinal.png')


#patologias


print('')
if(bpmm==0):
    print('Patologia encontrada:Parada cardíaca - BPM medio:',bpmm)
    i=1
if(bpmm>0)&(bpmm<60):
    print('Patologia encontrada:Bradicardia Sinusal - BPM medio:',bpmm)
    i=2
if(bpmm>=100)&(bpmm<240):
    print('Patologia encontrada:Taquicardia Sinusal - BPM medio:',bpmm)
    i=3
if(bpmm>=240)&(bpmm<350):
    print('Patologia encontrada:Flütter Atrial - BPM medio:',bpmm)
    i=4
if(bpmm>=360)&(bpmm<600):
    print('Patologia encontrada:Fibrilação Atrial - BPM medio:',bpmm)
    i=5
```

```
t1 = time.clock()



print('Tempo de processamento, em segundos:',t1 - t0)

#plt.show()


#FIM DO CODIGO PRINCIPAL


#SERVIDOR WEB FLASK

while i ==1:

    app = Flask(__name__)

    @app.route("/")

    def hello_world():

        return 'Patologia encontrada: Parada cardíaca'

    app.run()

while i ==2:

    app = Flask(__name__)

    @app.route("/")

    def hello_world():

        return 'Patologia encontrada: Bradicardia Sinusal'

    app.run()

while i ==3:
```

```
        app = Flask(__name__)
        @app.route("/")
        def hello_world():
            return 'Patologia encontrada: Taquicardia Sinusal'
        app.run()
    while i ==4:
        app = Flask(__name__)
        @app.route("/")
        def hello_world():
            return 'Patologia encontrada: Flütter Atrial'
        app.run()
    while i ==5:
        app = Flask(__name__)
        @app.route("/")
        def hello_world():
            return 'Patologia encontrada: Fibrilação Atrial'
        app.run()
    #FIM DO ALGORITIMO
    #GLÓRIA A DEUS _/\_
```