UNIVERSIDADE DA INTEGRAÇÃO INTERNACIONAL DA LUSOFONIA AFRO-BRASILEIRA

INSTITUTO DE ENGENHARIAS E DESENVOLVIMENTO SUSTENTÁVEL

PROGRAMA DE PÓS-GRADUAÇÃO EM ENERGIA E AMBIENTE

MESTRADO ACADÊMICO EM ENERGIA E AMBIENTE

ADRIEL DE OLIVEIRA FREITAS

DESENVOLVIMENTO DE UMA PLATAFORMA IOT MÓVEL DE AQUISIÇÃO E PROCESSAMENTO DE SINAIS CARDIOLÓGICOS VIA APLICAÇÃO WEB

REDENÇÃO-CE

2022

ADRIEL DE OLIVEIRA FREITAS

DESENVOLVIMENTO DE UMA PLATAFORMA IOT MÓVEL DE AQUISIÇÃO E PROCESSAMENTO DE SINAIS CARDIOLÓGICOS VIA APLICAÇÃO WEB

Dissertação apresentada ao Curso de Mestrado Acadêmico em Energia e Ambiente do Programa de Pós-Graduação em Energia e Ambiente do Instituto de Engenharias e Desenvolvimento Sustentável da Universidade da Integração Internacional da Lusofonia Afro-Brasileira, como requisito parcial à obtenção do título de mestre em Energia e Ambiente. Área de Concentração: Processamento de Sinais

Orientador: Prof. Dr. João Paulo do Vale Madeiro

REDENÇÃO-CE

2022

Universidade da Integração Internacional da Lusofonia Afro-Brasileira
Sistema de Bibliotecas da UNILAB
Catalogação de Publicação na Fonte.

Freitas, Adriel de Oliveira.

F866d

Desenvolvimento de uma Plataforma IOT Móvel de Aquisição e Processamento de Sinais Cardiológicos via Aplicação WEB/ Adriel de Oliveira Freitas.
- Redenção, 2023.
81fl: il.

Dissertação - Curso de Mestrado Acadêmico Em Energia E Ambiente, Programa De Pós-graduação Em Energia E Ambiente, Universidade da Integração Internacional da Lusofonia Afro-Brasileira, Redenção, 2023.

Orientador: Prof.° Dr.° João Paulo do Vale Madeiro.

1. Arquitetura IoT. 2. Sinal de ECG. 3. Processamento de Sinais. 4. Monitoramento. 5. Aplicação WEB. I. Madeiro, João Paulo do Vale. II. Título.

CE/UF/BSCA CDD 006.6

ADRIEL DE OLIVEIRA FREITAS

DESENVOLVIMENTO DE UMA PLATAFORMA IOT MÓVEL DE AQUISIÇÃO E PROCESSAMENTO DE SINAIS CARDIOLÓGICOS VIA APLICAÇÃO WEB

Dissertação apresentada ao Curso de Mestrado Acadêmico em Energia e Ambiente do Programa de Pós-Graduação em Energia e Ambiente do Instituto de Engenharias e Desenvolvimento Sustentável da Universidade da Integração Internacional da Lusofonia Afro-Brasileira, como requisito parcial à obtenção do título de mestre em Energia e Ambiente. Área de Concentração: Processamento de Sinais

Aprovada em:

BANCA EXAMINADORA

Prof. Dr. João Paulo do Vale Madeiro (Orientador)
Universidade Federal do Ceará (UFC)

Prof. Dr. Fernando Antonio Mota Trinta
Universidade Federal do Ceará (UFC)

Prof. Dr. Cícero Saraiva Sobrinho
Universidade da Integração Internacional da
Lusofonia Afro-Brasileira (UNILAB)

AGRADECIMENTOS

A Deus, ser Supremo, ao qual devo a vida.

A minha mãe, pela compreensão nos momentos de anseio e ausência, e pelas palavras de incentivo.

Ao Prof. Dr. João Paulo do Vale Madeiro pelos conselhos e pela excelente orientação.

Aos meus professores do PGEA, que a cada momento contribuíram para o meu aprendizado com apoio e compreensão durante a realização do curso.

Aos demais familiares, pelo incentivo e carinho.

Aos amigos, que de alguma maneira me ajudaram a passar pelos momentos difíceis, com conversas e descontração

À Universidade da Integração Internacional da Lusofonia Afro-Brasileira- Aurora, pelo apoio financeiro e estrutura física oferecida.

À Universidade Federal do Ceará, pela parceria através do projeto SiMONE: Sensoriamento e MONitoramento remoto da vitalidade FEtal.

Ao Doutorando em Engenharia Elétrica, Ednardo Moreira Rodrigues, e seu assistente, Alan Batista de Oliveira, aluno de graduação em Engenharia Elétrica, pela adequação do *template* utilizado neste trabalho para que o mesmo ficasse de acordo com as normas.

“A mente que se abre a uma nova ideia jamais voltará ao seu tamanho original”

(Albert Einstein)

RESUMO

Nos últimos anos, a aquisição e o monitoramento de sinais de ECG têm sido amplamente utilizados no campo da medicina cardiológica. Os sinais de ECG podem fornecer informações importantes sobre a atividade cardíaca de pacientes, permitindo que sua análise seja utilizada como ferramenta de auxílio ao diagnóstico médico. A presença de ruídos em sinais de ECG pode afetar o processo de detecção na aquisição de medições precisas e confiáveis do batimento cardíaco para o sistema de monitoramento de ECG, além da possibilidade de gerar alarmes falsos em classificadores automáticos de doenças cardiovasculares. O presente trabalho propõe o desenvolvimento e implementação de um sistema IoT móvel de baixo custo para aquisição, processamento, armazenamento em nuvem e monitoramento remoto de sinais de ECG. Neste sistema, o sinal de ECG é detectado e condicionado por um biossensor acoplado a um Arduíno UNO R3. Em seguida, o sinal coletado é enviado para uma aplicação *WEB* desenvolvida e hospedada com o auxílio dos softwares MATLAB *designer* ™ e MATLAB *WEB App Server* ™. A comunicação entre a aplicação *WEB* e o *hardware* do sistema pode ser estabelecida de três formas distintas: via comunicação serial, *bluetooth* ou *Wi-Fi*. Os testes de operação de todos os elementos do sistema proposto foram realizados em pacientes do Hospital Universitário Walter Cantídio da Universidade Federal do Ceará (UFC). Foram coletados 50 registros de 5 (cinco) min a uma frequência de amostragem de 256 Hz. Os resultados de desempenho da aplicação *WEB* foram satisfatórios, apresentando um tempo médio de apenas 1,7830 s com um desvio padrão de 0,0737 s para calcular a DFT, aplicar uma decomposição *wavelet* multinível em um *Buffer* de 7 (sete) segundos, gerar um novo sinal a partir das componentes de frequências desejadas e plotar os resultados. O monitoramento remoto e armazenamento em nuvem dos registros cardiográficos foram realizados com êxito.

Palavras-chave: Arquitetura IoT, Sinal de ECG, Monitoramento, Processamento de Sinais, Aplicação WEB.

ABSTRACT

In recent years, the acquisition and monitoring of ECG signals have been widely used in the field of cardiology. ECG signals can provide important information about the cardiac activity of patients, allowing their analysis to be used as a tool to aid medical diagnosis. The presence of noise in ECG signals can affect the detection process in acquiring accurate and reliable measurements of the heartbeat for the ECG monitoring system, as well as the possibility of generating false alarms in automatic classifiers of cardiovascular diseases. This work proposes the development and implementation of a low-cost mobile IoT system for the acquisition, processing, cloud storage, and remote monitoring of ECG signals. In this system, the ECG signal is detected and conditioned by a biosensor coupled with an Arduino UNO R3. Then, the collected signal is sent to a web application developed and hosted with the help of MATLAB designer ™ and MATLAB WEB App Server ™ software. The communication between the web application and the hardware of the system can be established in three different ways: via serial communication, Bluetooth, or Wi-Fi. Operation tests of all the elements of the proposed system were carried out on patients from the Walter Cantidio University Hospital of the Federal University of Ceará (UFC). Fifty records of 5 (five) minutes were collected at a sampling frequency of 256 Hz. The performance results of the web application were satisfactory, with an average time of only 1.7830 s and a standard deviation of 0.0737 s to calculate the DFT, apply a multilevel wavelet decomposition in a 7 (seven) seconds buffer, generate a new signal from the desired frequency components, and plot the results. The remote monitoring and cloud storage of the cardiographic records were successfully performed.

Keywords: IoT Architecture. ECG Signal. Monitoring. Signal Processing. WEB Application.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

AB	*Abrupt Change*
AWGN	*Additive White Gaussian Noise*
BW	*Baseline Wander*
cA	Componente de Aproximação
cD	Componente de Detalhe
CPU	Unidade Central de Processamento
ECG	Eletrocardiograma
FFT	*Fast Fourier Transform*
FIR	*Finite Impulse Response*
FL	*Flat Line*
GUI	Interface Gráfica do Usuário
ICSP	*In Circuit Serial Programmer*
IDE	Ambiente de Desenvolvimento Integrado
IoT	*Internet of Things*
ISM	*ndustrial, Scientific, and Medical*
MA	*Muscle Artifacts*
MIT	*Massachusetts Institute of Technology*
MQTT	*Message Queuing Telemetry Transport*
PLI	*Power Line Interference*
PWM	*Pulse Width Modulation*
SPP	*Serial Port Protocol*
TIW	Transformada inversa de Wavelet
TVN	*Time-Varying Noise or Pause*
TWD	Transformada Wavelet Discreta

SUMÁRIO

1 INTRODUÇÃO

Segundo a Organização Mundial da Saúde (OMS), em 2019 as doenças cardiovasculares foram a principal causa de mortes no mundo, sendo responsáveis por 16% dos falecimentos em todo o planeta. As doenças cardíacas têm se tornado uma preocupação séria, pois afetam pessoas de todas as faixas etárias. De 2000 a 2019, o maior aumento de mortes também foi por doenças cardiovasculares: A quantidade de pessoas afetadas por estas doenças passou de 6,8 milhões para 8,9 milhões(OMS, 2020), onde diversos fatores de risco contribuem para o aparecimento de doenças cardiovasculares como obesidade, hipertensão, tabagismo, sedentarismo, colesterol alto, estresse e diabetes mellitus (JANVEJA; TRIVEDI, 2022).

Uma das principais formas de combate a estas doenças é por meio de exames preventivos. Dentre eles destaca-se o eletrocardiograma (ECG), exame que monitora e registra o sinal formado pela diferença de potencial gerada pela atividade elétrica cardíaca. O monitoramento contínuo do Sinal de ECG pode ser usado para identificar diversos problemas cardiovasculares, como arritmias, isquemia miocárdica, infarto agudo do miocárdio, doenças do músculo cardíaco e outras condições (Li *et al.*, 2017). Profissionais da área da medicina utilizam o monitoramento do sinal de ECG como uma prática importante na prevenção e diagnóstico precoce de doenças cardiovasculares, permitindo o tratamento adequado e a redução do risco de complicações e morte (JANVEJA; TRIVEDI, 2022). É recomendado que o exame de ECG seja realizado regularmente, especialmente em pessoas com fatores de risco para doenças cardiovasculares (HARSKAMP, 2019).

Para realizar a aquisição, monitoramento e processamento de sinais de ECG é necessário ter equipamentos específicos para captar os sinais que caracterizam a atividade elétrica do coração e convertê-los em sinais visuais ou digitais. Esses equipamentos podem ser caros e difíceis de adquirir em algumas regiões ou países, especialmente em locais com sistemas de saúde precários ou com recursos financeiros limitados (AHAMED *et al.*, 2015). No entanto, com o avanço da tecnologia, atualmente é possível desenvolver dispositivos mais acessíveis e móveis, que podem ser utilizados em ambientes diversos, incluindo ambulâncias, clínicas e hospitais. Além disso, algumas iniciativas têm buscado desenvolver soluções tecnológicas que proporcionam o monitoramento e compartilhamento em nuvem de sinais de ECG coletados em tempo real, como softwares e aplicativos que se integram com dispositivos móveis utilizados no processo de aquisição dos sinais de ECG (ZACARIAS *et al.*, 2014).

1.1 Problema/Justificativa

O custo associado à realização de ECGs, bem como a impossibilidade de se transportar tais equipamentos para monitoramento fora de locais específicos dificulta o acesso de pacientes, especialmente os mais carentes, à um tratamento adequado. Com isso, alternativas precisam ser propostas.

Um caminho interessante é o uso de estratégias baseadas em dispositivos embarcados no modelo da chamada Internet das Coisas (IoT), que se refere a interconexão de dispositivos físicos (sensores, atuadores, dispositivos de processamento, entre outros) por meio de redes de comunicação, permitindo que esses dispositivos coletem, transmitam e processem dados em tempo real. Na área médica, essa tecnologia pode ser utilizada para monitoramento remoto de pacientes, coleta de dados de sinais vitais, diagnóstico remoto, entre outras aplicações. Hoje, já existem diversos sensores de baixo custo capazes de monitorar sinais cardíacos que podem ser utilizados para prototipar soluções mais baratas, leves e de fácil transporte, como por exemplo, biossensor. Estes sensores capturam os sinais de ECG brutos, que podem ser usados para criação de diversas aplicações.

Porém, estes sinais ainda sofrem com ruídos quando processados por dispositivos baseados em microcontroladores. Esses ruídos podem ser causados por diversos fatores, como interferências eletromagnéticas, movimentos do paciente, variações de impedância elétrica, entre outros. Para minimizar ou remover esses ruídos, é necessário aplicar técnicas de processamento de sinais que permitam separar o sinal de ECG dos ruídos presentes. Existem diversas técnicas de processamento de sinais que podem ser utilizadas, tais como: Aplicação de filtros digitais, decomposição em componentes independentes (do inglês: *Independent Component Analysis*), técnicas de aprendizado de máquina e algoritmos de classificação.

Outro fator importante é a capacidade de compartilhar ou facilitar o acesso de exames entre especialistas de saúde. Neste sentido, têm-se tornado padrão o uso de aplicações WEB para acesso a diversos domínios, como aplicações bancárias, comércio eletrônico, dentre outros. Com isso, a forma de acesso a diversos por meio de metáforas como botões, links, listas de seleção têm se tornado padrão para o desenvolvimento de sistemas atualmente. Seguir este modelo facilitaria o engajamento de profissionais, uma vez que eles já estão acostumados a usar tais aplicações fora do domínio da saúde.

Por fim, ao usar os dados obtidos dos sensores, seria também interessante auxiliar os especialistas na detecção de problemas cardiovasculares de acordo com padrões já conhecidos na

literatura médica. Para isso, o uso de algoritmos computacionais aplicada a técnicas clássicas e modernas de processamento digital aparecem como possível solução para a extração automática de informações sobre a morfologia e o comportamento do sinal de ECG analisado.

Neste sentido, é necessário pensar em uma proposta que permita resolver de modo sistemático os problemas supracitados em uma solução viável para coleta de ECG de modo rápido, com fácil transporte, baixo custo e fácil uso para seus usuários finais.

1.2 Objetivo

Com base no cenário e problemas citados anteriormente, esta dissertação de mestrado têm por objetivo principal propor uma arquitetura de sistema baseada em dispositivos móveis e sensores IoT para obter informações de ECG de pacientes, e uma aplicação WEB capaz de processar, apresentar e compartilhar tais informações, em modelo colaborativo.

Para se conseguir este objetivo principal, outras metas secundárias também precisaram ser alcançadas. Dentre elas, cita-se:

1. Desenvolver um protótipo de baixo custo baseado em IoT capaz de obter dados de ECG de modo fácil, rápido e confiável;
2. Desenvolver uma solução WEB capaz de permitir que diversos usuários sejam capazes de acessar a dados de pacientes remotamente;
3. Utilizar algoritmos para detectar ruídos;
4. Utilizar algoritmos para análise espectral e aplicação de técnicas modernas de processamento digital em sinais de ECG, com o propósito de auxiliar no diagnóstico médico;
5. Validar a solução com pacientes reais.

1.3 Metodologia

Para se alcançar o objetivo final, este trabalho seguiu um conjunto de etapas

1. Levantamento de trabalhos relacionados, onde foram encontrados 5 trabalhos voltados ao desenvolvimento de dispositivos baseados em microcontroladores para coletar, monitorar e processar sinais de ECG. Esse levantamento é importante para ajudar a identificar possíveis oportunidades de inovação e aprimoramento da tecnologia existente, bem como para entender as principais limitações e desafios a serem enfrentados no desenvolvimento de uma nova solução.

2. Levantamento dos requisitos de sua solução, tanto de hardware quanto software. No que diz respeito ao hardware, foram levados em consideração critérios como baixo consumo de energia, baixo custo, possuir hardware livre e proporcionar uma prototipagem rápida. Esses critérios são importantes para garantir que o dispositivo possa ser utilizado por longos períodos de tempo, sem a necessidade de troca frequente de baterias, além de permitir a produção em larga escala a um baixo custo. O hardware livre e a prototipagem rápida permitem maior flexibilidade e agilidade no desenvolvimento do projeto. Já para a plataforma de desenvolvimento de aplicações WEB, foram levados em consideração requisitos como facilidade de criação de interface gráfica, possibilidade de integrar algoritmos de processamento de sinais e comunicação via Serial UART, bluetooth e WIFI utilizando protocolo MQTT.
3. Modelagem da Arquitetura de Solução. A arquitetura da solução é composta por três principais camadas: a camada de sensoriamento, a camada de processamento e a camada de aplicação. A camada de sensoriamento é responsável pela aquisição dos dados e consiste no biossensor ECG-EMG Olimex, que é utilizado para detectar o sinal elétrico produzido pelo coração e convertê-lo em um sinal elétrico analógico. O sinal analógico é então enviado para o Arduíno UNO R3, que é responsável por converter o sinal em uma representação digital que pode ser processada ou enviada para outros dispositivos. A camada de processamento é responsável pelo processamento dos dados e consiste no Arduíno UNO R3, que é responsável por receber o sinal digital do biossensor e processá-lo. Os dados podem ser processados localmente ou enviados para outras camadas para processamento adicional. A camada de aplicação é responsável pela apresentação dos dados ao usuário e consiste em um aplicativo web criado usando o ambiente de desenvolvimento MATLAB App Designer. O aplicativo permite a visualização dos dados do sinal elétrico do coração em tempo real e fornece informações adicionais, como batimentos cardíacos por minuto e análise da frequência cardíaca.
4. Desenvolvimento da Solução de Hardware e do Software. Para prototipar o hardware do dispositivo móvel, usamos um biossensor ECG-EMG Olimex, um Arduíno UNO R3, um módulo WI-FI ESP8266 e um módulo bluetooth HC-05. O biossensor detecta o sinal elétrico produzido pelo coração e o Arduíno o converte em sinal digital que pode ser processado ou enviado para outros dispositivos. O módulo WI-FI e o módulo bluetooth permitem a conexão sem fio do dispositivo móvel com outros dispositivos,

como smartphones ou computadores, para enviar e receber dados. Esses módulos também permitem a integração do dispositivo com plataformas de IoT, possibilitando a análise e compartilhamento dos dados coletados pelo dispositivo. Para desenvolvimento da aplicação aplicação WEB, foi utilizado o ambiente de desenvolvimento de aplicativos executáveis e aplicativos WEB do MATLAB ®. Este ambiente de desenvolvimento é denominado de MATLAB App Designer. Nele, é possível desenvolver aplicativos de alta qualidade apenas movendo os componentes visuais para definir o design da sua interface gráfica com o usuário (GUI), utilizando o editor integrado para programar rapidamente seu comportamento.

5. Testes de desempenho, testes com pacientes. Para os testes de desempenho na aplicação WEB, foi utilizado o MATLAB profile, que permitiu avaliar o tempo de execução de cada função implementada. Foram realizados 30 ensaios, em que a aplicação WEB foi submetida a diversas operações, como a decomposição multinível wavelet, o cálculo da DFT e a plotagem dos sinais coletados. Os resultados desses testes foram utilizados para otimizar os algoritmos implementados e melhorar a eficiência da aplicação. Já nos testes com pacientes, foram coletados registros eletrocardiográficos de 50 pacientes hospitalizados em fase ativa da doença Covid-19. Cada registro coletado possui uma duração de 5 minutos e juntos formam um banco de dados para estudos colaborativos com o objetivo de identificar a existência de um padrão de alteração na variabilidade da frequência cardíaca. Esses testes permitiram verificar a qualidade do sinal adquirido pelo dispositivo de aquisição de sinais ECG e a estabilidade do dispositivo durante a coleta de dados.

1.4 Estrutura da Dissertação

Esta dissertação está dividida em seis capítulos. Neste primeiro capítulo, apresenta-se a motivação por detrás da sua formulação, bem como os objetivos, a metodologia adotada para alcançar o objetivo final e o modo como a dissertação está organizada.

Já o Capítulo 2, apresentará um levantamento dos trabalhos relacionados encontrados e discutir suas principais contribuições para a área de coleta de sinais de ECG baseada em microcontrolador. Essa análise dos trabalhos relacionados servirá para contextualizar a proposta em relação ao conhecimento existente, identificar possíveis melhorias e definir as lacunas de conhecimento a serem preenchidas com o estudo em questão.

No Capítulo 3, é exposto, inicialmente, uma introdução à eletrocardiografia, seguida da discussão de conceitos fisiológicos e clínicos aplicados ao exame de eletrocardiograma. Em seguida, são apresentados os tipos de derivação padrão utilizados no exame de ECG, incluindo as derivações bipolares (DI, DII e DIII), as derivações unipolares aumentadas (aVR, aVL e aVF) e as derivações precordiais (V1 a V6).

O Capítulo 4, aborda a metodologia adotada. Inicialmente, é apresentado um detalhamento sobre os componentes escolhidos para compor o Hardware do sistema proposto, bem como os esquemáticos elétricos e procedimentos de montagem. Em seguida, são descritos os procedimentos utilizados para desenvolvimento da aplicação WEB e dos algoritmos de processamento de sinais. Por fim, é realizado um detalhamento sobre os algoritmos desenvolvidos para detectar de forma não supervisionada os principais ruídos que corrompem os sinais de ECG.

No Capítulo 5, são apresentados os resultados alcançados fazendo uma análise deles e gerando discussões. Inicialmente, são apresentados os parâmetros de orçamento do sistema, que foram definidos como parte do processo de projeto. Em seguida, são apresentados os ensaios realizados para levantar os parâmetros de energia consumida pelo hardware do sistema. Os resultados dos testes de operação do sistema também são apresentados neste capítulo. Por fim, é apresentada uma comparação do sistema proposto com trabalhos correlatos, a fim de avaliar sua eficiência e relevância em relação a outras soluções existentes.

No Capítulo 6, por fim, são apresentadas as contribuições finais deste trabalho e as indicações de trabalhos que podem dar continuidade a esta pesquisa e desenvolvimento.

2 TRABALHOS RELACIONADOS

Recentemente, muitos dispositivos programáveis foram propostos para realizar a coleta do sinal de ECG. O trabalho realizado por Pereira *et al.* (2022) propos uma arquitetura IoT para aquisição, armazenamento e visualização de sinais de ECG. Neste trabalho, foi utilizado uma *shield* ECG-EMG para detectar o sinal eletrocardiográfico e em seguida convertê-lo em um sinal elétrico analógico. A representação discreta do sinal analógico proveniente da *shield* é obtido com o auxílio de um conversor analógico-digital de um microcontrolador ESP 32, que por sua vez processa o sinal e o envia para uma *Google Spreadsheet*. O ESP 32 é programado utilizando linguagem *microphyton*.

Em Al-Busaidi e Khriji (2013), os autores propuseram o projeto de uma arquitetura para coleta de sinais de ECG baseada em microcontrolador Atmega32. Esta arquitetura conta com filtros digitais integrados do tipo passa-baixa, *notch* e passa-banda. A coleta das amostras do sinal de ECG é realizada a partir da conexão por cabo entre um dispositivo e um computador.

O trabalho realizado por Gupta *et al.* (2010) propos o desenvolvimento de um sistema (*hardware* e *software*) de baixo custo para aquisição, armazenamento e processamento de sinais de ECG. O *hardware* deste sistema foi desenvolvido com base em microcontrolador, amplificador de instrumentação, ADC, conversores de nível RS 232 com seus CIs periféricos. Já o *software* consiste em uma interface gráfica de usuário (GUI) desenvolvida com o auxilio do *software* MATLAB. Neste sistema, o sinal de ECG é amostrado a uma taxa de 1 (um) kHz e, após a digitalização, é alimentado por um sistema embarcado baseado em microcontrolador para em seguida converter os dados de ECG em um fluxo de *bits serial* formatado em RS232.

Em Bansal *et al.* (2009), é proposto um protótipo de sistema de aquisição e processamento de sinais de ECG formado por um amplificador *front-end*, um transceptor sem fio e uma interface com dispositivo de exibição de saída e filtros digitais para remoção de ruído. O amplificador *front-end* é responsável por detectar e condicionar os sinais de ECG. Ele é formado por eletrodos, amplificadores de ganho de *buffer* e unidade, circuito de restauração DC, circuito de acionamento da perna direita, circuito de filtro ativo e unidade de fonte de alimentação. O transceptor utilizado é composto por um transmissor e uma unidade receptora. Para envio dos sinais coletados sem a necessidade de cabos, o transmissor fica conectado ao amplificador *front-end* enquanto a unidade receptora fica conectada à porta de som do computador usando um conector compatível. A partir de uma Interface Gráfica, o usuário pode visualizar os sinais coletados e os resultados da aplicação de filtros digitais FIR (do inglês: *Finite Impulse Response*).

Os autores em Christ *et al.* (2020) propuseram um sistema IoT para monitoramento contínuo de sinais de ECG. Na implementação deste sistema foi utilizado o computador *single-board Raspberry-Pi* para processar o sinal proveniente de múltiplos biossensores. Caso seja detectado inconformidades no sinal coletado, o sistema envia uma mensagem para o médico por meio de um módulo GSM conectado na porta seria do *Raspberry*. Os sinais coletados são encaminhados para um *dashboard* desenvolvido na plataforma de análise IoT *ThingSpeak*.

Em relação aos trabalhos descritos anteriormente, muitos projetos usaram Arduíno, ESP 32 e *Raspberry-Pi* por serem dispositivos destinados à prototipagem rápida e por apresentarem um ambiente de desenvolvimento de programação de fácil utilização. A maioria dos sistemas de aquisição de ECG que acompanham uma interface gráfica de usuário necessita de uma CPU para executar a interface e estabelecer uma comunicação com o *hardware* do sistema via comunicação serial. Uma abordagem pouco utilizada é a aplicação *WEB* para mediar a interação entre usuários e arquiteturas IoT de aquisição, processamento, armazenamento e monitoramento de sinais de ECG.

3 FUNDAMENTAÇÃO TEÓRICA

3.1 Eletrocardiografia

A eletrocardiografia é o estudo dos registros gráficos da atividade elétrica do coração, representada pelo eletrocardiograma (ECG). O ECG registra os sinais elétricos do coração por meio dos eletrodos posicionados na superfície do corpo (GOLDBERGER, 2006). A eletrocardiografia é considerada a principal técnica usada para estudos acerca da dinâmica dos desenvolvimentos de doenças cardiovasculares, sendo considerada uma técnica menos invasiva e que, por isto, acaba por deixar o paciente mais relaxado (OGANISYAN *et al.*, 2020).

O primeiro ECG foi desenvolvido por Willem Einthoven em 1902. Embora enfrentasse oposição na época para ser utilizada como uma ferramenta de diagnóstico, a técnica se tornou posteriormente uma das principais ferramentas de diagnóstico de doenças cardíacas. Apesar de ter passado por evoluções ao longo dos anos, sua mecânica é a mesma criada por Einthoven e, mesmo após mais de 100 anos de sua criação, o ECG é o aparelho de cardiologia mais reconhecido e utilizado por médicos em todo o mundo (CAJAVILCA; VARON, 2008).

O ECG teve avanços contínuos nos últimos dois séculos, tornando-se a primeira e a mais comum técnica para o diagnóstico cardíaco. O eletrocardiograma é ainda considerado importante na avaliação de terapias para pacientes que sofrem com insuficiência cardíaca, bem como na identificação e avaliação de pacientes com doenças genéticas propensas à arritmia (MIRVIS; GOLDBERGER, 2001).

Desta forma, a eletrocardiografia é fundamental na compreensão das doenças cardíacas como também na avaliação de pacientes com problemas cardíacos, principalmente por ser uma ferramenta não invasiva e com custo acessível para avaliar doenças cardíacas, tais como a arritmias e cardiopatias isquêmicas. Além disto, possui extrema importância no fornecimento de dados objetivos sobre a função e a estrutura do corpo humano. Por tudo isto, a eletrocardiografia é considerada por médicos(as) como um instrumento clínico essencial (ALGHATRIF; LINDSAY, 2012).

A eletrocardiografia é baseada em três técnicas: O eletrocardiograma clínico padrão (ECG), o vetorcardiograma (VCG) e o Monitoramento de ECG. A primeira técnica, o ECG clínico padrão, ou eletrocardiograma, de repouso registra os sinais elétricos do coração, incluindo as 12 derivações. Ou seja, são registrados 12 potenciais diferentes na superfície do corpo (TOMPKINS, 2000). A segunda técnica, o VCG, consiste na representação da força eletromotriz gerada pelo

coração durante as atividades cardíacas. Esta representação ocorre por meio de um único vetor em que os vetores sucessíveis possuem pontos de origem comuns (RIERA *et al.*, 2007).

A terceira técnica consiste no monitoramento de ECG ou eletrocardiografia dinâmica. Esta técnica baseia-se no registro de uma ou duas derivações das 12 do eletrocardiograma clínico padrão, para ser realizado o monitoramento destas derivações. Isto é feito para que seja possível verificar a ocorrência de distúrbios ou alguma anomalia no ritmo dos batimentos cardíacos do paciente (TOMPKINS, 2000).

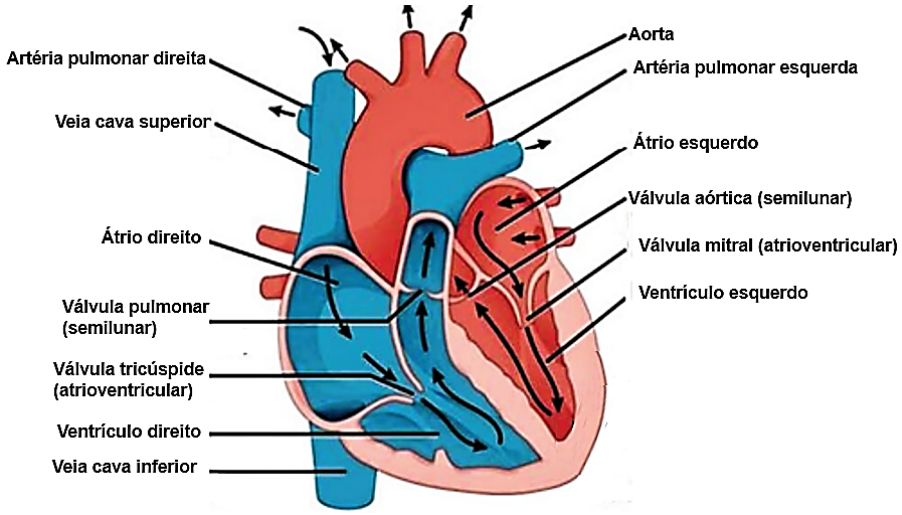
3.2 Anatomia e fisiologia do coração

O coração é o principal órgão do sistema circulatório, atuando como uma bomba eletricamente temporária com a função de contrair ritmicamente e bombear o sangue primeiramente para os pulmões (promovendo sua oxigenação) e depois bombear este sangue oxigenado por todo o organismo (GOLDBERGER, 2006).

O coração é composto por quatro câmaras (Figura 1): Duas delas superiores chamadas de átrios e duas inferiores denominadas de ventrículos. Além destas câmaras, o órgão é composto por quatro válvulas que abrem e fecham de acordo com as mudanças de pressão provocadas pelos movimentos de contração e relaxamento do coração. Durante estes movimentos, o sangue é ejetado para dentro dos ventrículos ou para fora do coração. As válvulas servem para impedir o refluxo do sangue (TORTORA; DERRICKSON, 2014). Conforme este autor, as quatro válvulas são:

- Válvulas atrioventriculares (AV): Situadas entre os átrios e o ventrículo;
- Válvulas tricúspides (VT): Situadas entre o átrio direito e o ventrículo direito. Esta válvula é composta por outras três (cúspides);
- Válvulas bicúspides ou mitral: Situadas entre o átrio esquerdo e o ventrículo esquerdo;
- Válvulas semilunares ou válvula da aorta: Situadas no óstio, entre o ventrículo esquerdo e a aorta.

Figura 1 – Representação do corte frontal do coração humano.

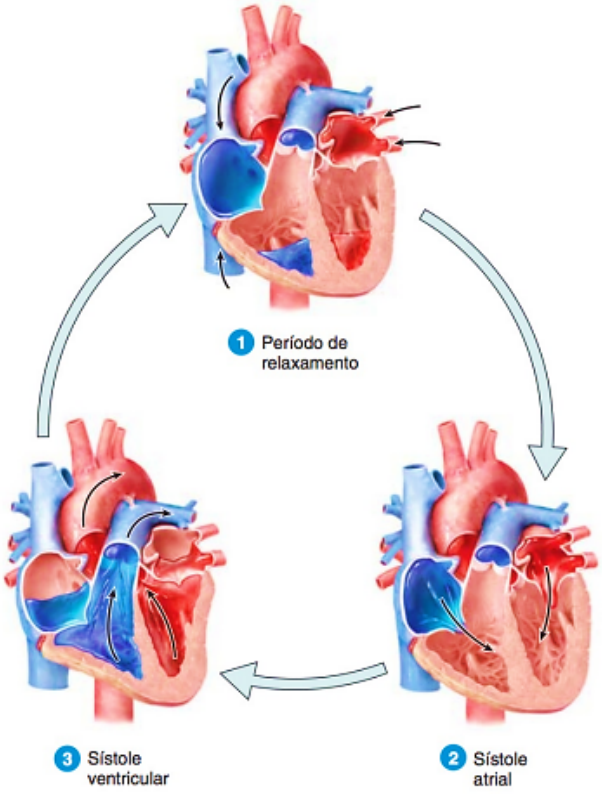


Fonte: Adaptado de Zhong *et al.* (2019)

O processo de bombeamento é iniciado quando o sangue desoxigenado do corpo entra no coração por meio da veia cava superior e inferior, fundindo-se ao entrar no átrio direito, o qual faz fluir este sangue para o ventrículo direito, passando pelas válvulas tricúspides e impedindo o seu retorno durante a contração do ventrículo. Do ventrículo direito, o sangue é bombeado para os pulmões por meio da válvula pulmonar para a artéria pulmonar que, em seguida, bombeia o sangue para os pulmões, ocorrendo o retorno do sangue para o átrio esquerdo por meio das veias pulmonares. No pulmão, o sangue é oxigenado e segue para o átrio esquerdo onde é bombeado para o ventrículo esquerdo passando pela válvula mitral e impedindo o retorno do sangue com a contração do ventrículo, o qual ao contrair impulsiona o bombeamento do sangue para fora do coração por meio da válvula aórtica (ZHONG *et al.*, 2019).

Além disto, é por meio dos batimentos cardíacos que ocorre a normalização da pressão arterial, bem como a oxigenação dos tecidos e do sistema nervoso, garantindo que o coração seja capaz de atender às demandas das diferentes situações apresentadas pelo organismo. A prova disto é que se houver alguma alteração na pressão arterial e no nível de oxigênio, haverá por consequência alterações na frequência cardíaca (SCANLON; SANDERS, 2018).

Figura 2 – Ciclo cardíaco.



Fonte: (TORTORA; DERRICKSON, 2014, p. 352)

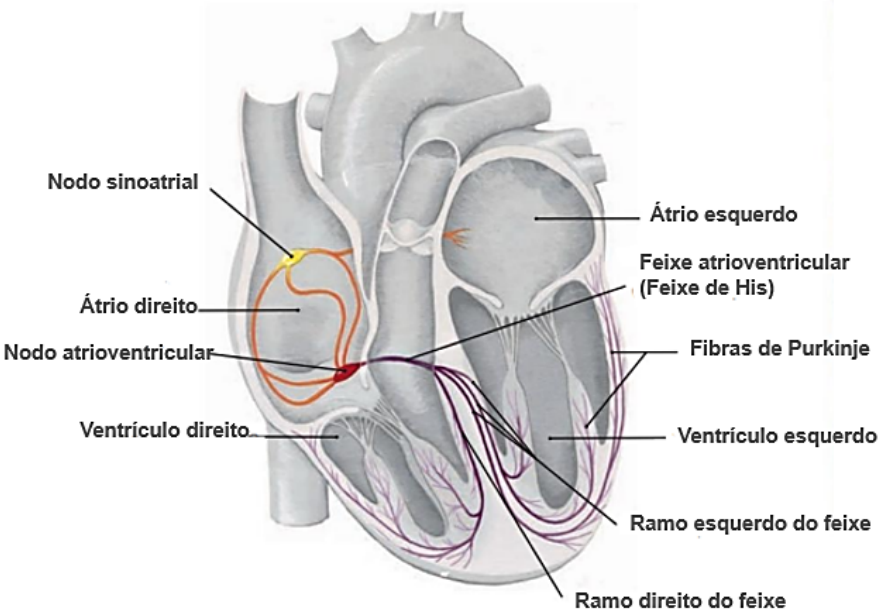
A sequência dos batimentos cardíacos forma o ciclo cardíaco. Este ciclo possui três etapas (Figura 2). A seguir é apresentado um detalhamento sobre cada uma destas etapas(TORTORA; DERRICKSON, 2014):

1. **Período de relaxamento:** Ocorre quando temos a repolarização dos ventrículos, iniciando o relaxamento dos ventrículos e provocando a diminuição da pressão em seu interior até que a mesma fique abaixo da pressão arterial. Este processo promove a abertura das válvulas AV e os ventrículos começam a encher, dando início à segunda fase.
2. **Sístole atrial:** Consiste na contração dos átrios promovida pelo potencial das atividades do nodo sinoatrial. Este fenômeno ocorre com as válvulas AV ainda abertas e as válvulas semilunares fechadas, iniciando a terceira e última fase.
3. **Sístole ventricular:** Esta fase começa com a despolarização ventricular que leva à contração dos ventrículos, os quais conduzem o sangue contra as válvulas AV que se fecham, promovendo a elevação da pressão nos ventrículos. Quando a pressão do ventrículo esquerdo supera a da aórtica e a do tronco pulmonar, ocorre a abertura das válvulas semilunares e temos a ejeção de sangue no coração. Além disto, os ventrículos começam a relaxar até a pressão neles diminuir e as válvulas semilunares se fecharem, dando início a

um novo ciclo.

Além disto, os impulsos elétricos gerados durante os batimentos do coração seguem uma rota específica por todo o miocárdio (Figura 2). Tais impulsos ocorrem por meio das células do miocárdio que são capazes de gerar seu próprio potencial eletrônico, o qual rapidamente se espalha do miocárdio para as demais células musculares por meio dos seus discos intercalados, promovendo assim a contração dos átrios seguida da contração dos ventrículos e gerando um ciclo cardíaco (SCANLON; SANDERS, 2018).

Figura 3 – Sistema de condução elétrica do coração.



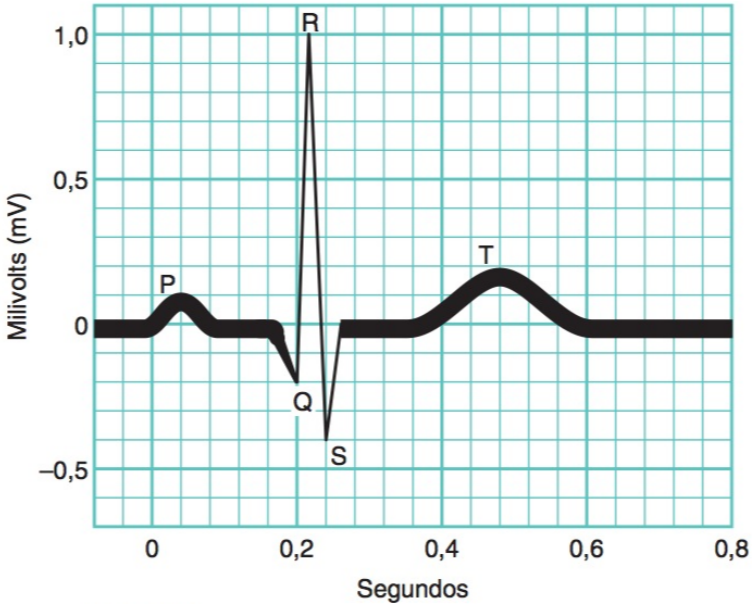
Fonte: Adaptado de Scanlon e Sanders (2018)

A estimulação elétrica cardíaca inicial ocorre por meio de um grupo de células capazes de gerar um estímulo elétrico (sinal). Estas células são conhecidas como o marca-passo natural do coração. O estímulo gerado por estas células é direcionado primeiramente para o átrio direito e depois para o átrio esquerdo, tendo início no nodo sinusal ou nodo sinoatrial (Figura 3), localizado no átrio direito próximo à veia cava superior e se espalhando para o nodo atrioventricular e o feixe de His; depois disto, o estímulo se espalha simultaneamente pelos ramos esquerdo e direito do feixe do miocárdio ventricular por meio das células condutoras das fibras de Purkinje, localizadas no miocárdio ventricular. Logo, a propagação dos estímulos elétricos pelos átrios e pelos ventrículos resulta no bombeamento do sangue para o pulmão e para a circulação geral (GOLDBERGER, 2006).

3.3 Eletrocardiograma (ECG)

O eletrocardiograma (ECG) corresponde à representação gráfica (Figura 4) da diferença de potencial durante as atividades de batimento do coração. Estes potenciais são refletidos no sinal ECG (OGANISYAN *et al.*, 2020).

Figura 4 – Eletrocardiograma (ECG) normal de um único batimento cardíaco.



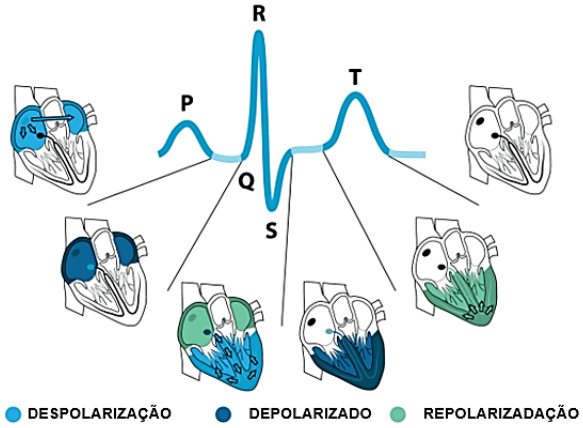
Fonte: Adaptado de Tortora e Derrickson (2014)

O eletrocardiograma típico se baseia em três ondas (Figura 4): a Onda P, o Complexo QRS e a Onda T (que correspondente a um evento específico (SCANLON; SANDERS, 2018):

- A Onda P corresponde à despolarização dos átrios;
- O Complexo QRS corresponde à despolarização dos ventrículos;
- A Onda T corresponde à repolarização dos ventrículos. O sinal referente à repolarização dos átrios coincide com o Complexo QRS.

Assim, o sinal do ECG em condições normais corresponde a repetições regulares das cinco deflexões das ondas P, Q, R, S e T (Figura 5), onde cada onda representa um único evento ocorrido durante o batimento cardíaco. Ou seja, cada onda está relacionada somente a um evento, podendo este ser o de despolarização ou de repolarização do coração. Deste modo, poderá ser associada e rasteada até o evento que lhe deu origem (PINTO *et al.*, 2018).

Figura 5 – Sequências de eventos de despolarização e repolarização e sua relação com as ondas P, Q, R, S e T.



Fonte: Adaptado de Pinto *et al.* (2018)

Os sinais destas ondas são detectados por eletrodos colocados em locais específicos do corpo que, em sistemas informatizados, serão digitalizados, armazenados e analisados seguindo determinados critérios para que o diagnóstico forneça a interpretação do eletrocardiograma, seja manualmente ou por meio do auxílio de computadores (MIRVIS; GOLDBERGER, 2001).

3.3.1 Sistema de derivações

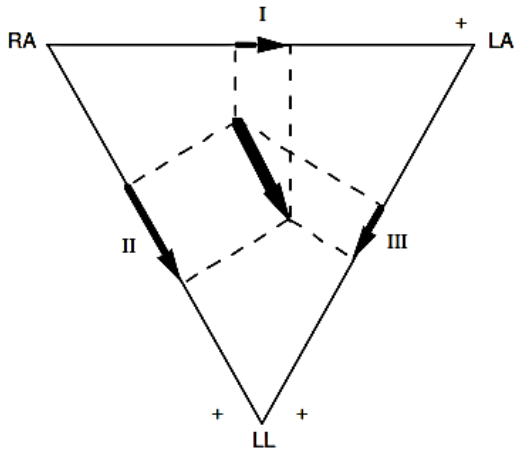
O primeiro sistema de derivações foi proposto pelo próprio Eintrhoven em 1912 quando desenvolveu o Triângulo de Eintrhoven (Figura 6), o qual representa a magnitude e a direção da diferença de potência criada pelo coração em vários locais do corpo humano, obtendo a relação das derivações I, II e III (ARORA; MISHRA, 2021).

As derivações I, II e III (Figura 7) são consideradas as derivações de membros bipolares ou derivações de membro padrão em que cada uma dessas derivações registra a diferença de potencial criado entre dois membros, conforme descrito abaixo (WASIMUDDIN *et al.*, 2020):

- Derivação padrão I: Entre o braço esquerdo e o braço direito;
- Derivação padrão II: Entre a perna esquerda e o braço direito;
- Derivação padrão III: Entre a perna esquerda e o braço esquerdo.

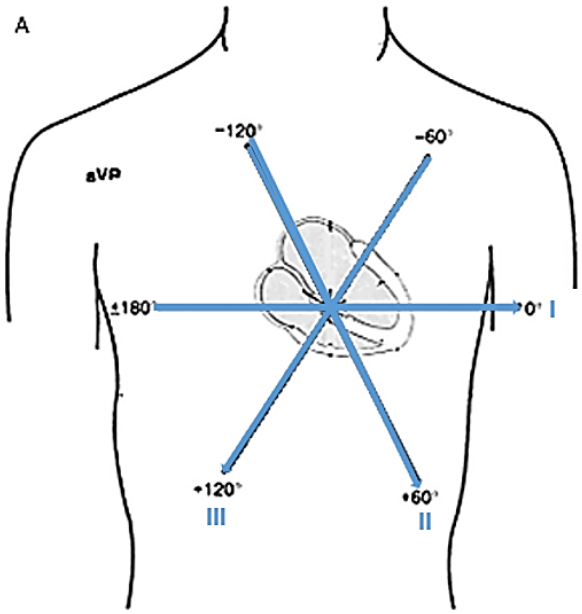
O eletrocardiograma clínico padrão possui, além das três derivações padrões, mais nove derivações: Três precordiais representadas por V1 a V6 (Figura 8) e três derivações de membros aumentadas representadas por aVR, aVL e aVF (Figura 9), as quais são considera-

Figura 6 – Triângulo de Eintrhoven. As abreviaturas RA, LA e LL (Abreviaturas do termo em inglês) representam respectivamente os membros: Braços direito, braço esquerdo e perna esquerda.



Fonte: (TOMPKINS, 2000, p. 28)

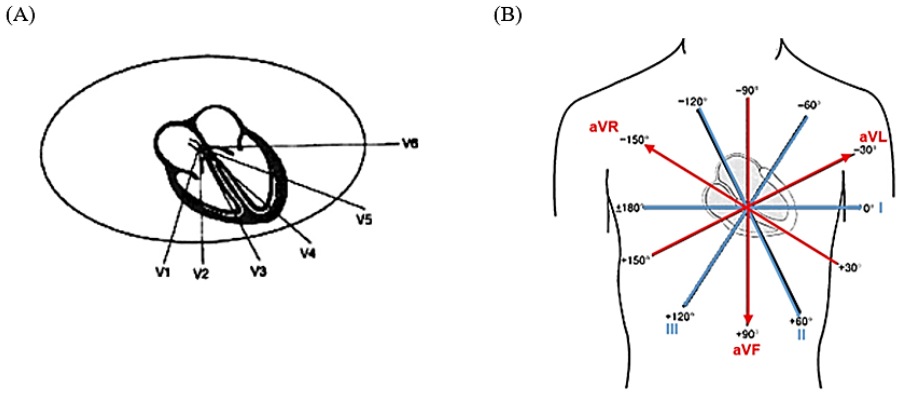
Figura 7 – Derivações bipolares em 60° de intervalos.



Fonte: Adaptado de AlGhatrif e Lindsay (2012)

das unipolares em virtude do potencial destas derivações corresponder à média dos potenciais detectados em dois ou mais eletrodos que formam o eletrodo composto. Fora isto, as derivações aumentadas geram amplitudes 50% maiores que as geradas pelas derivações precordiais (MIRVIS; GOLDBERGER, 2001).

Figura 8 – (A) Derivações precordiais (B) Derivações aumentadas em 60° de intervalos.



Fonte: Adaptado de AlGhatrif e Lindsay (2012)

As derivações aumentadas e precordiais devem ser posicionadas seguindo locais padrões para a disposição dos eletrodos conforme lista abaixo (MIRVIS; GOLDBERGER, 2001):

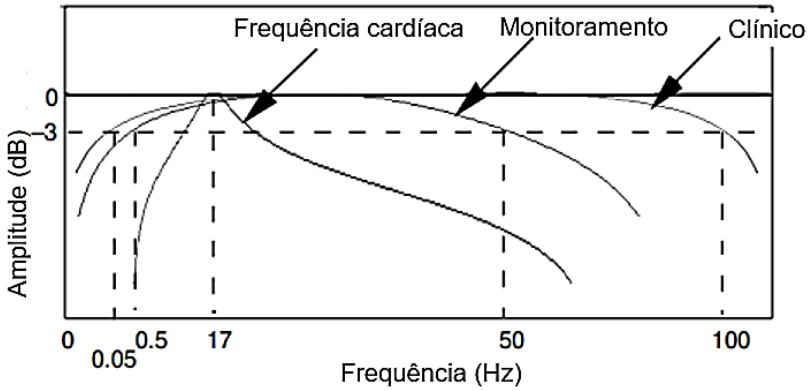
- Derivação aVR - Eletrodo colocado no braço direito e em uma entrada negativa no braço esquerdo mais perna esquerda;
- Derivação aVL – Eletrodo colocado no braço esquerdo e em uma entrada negativa no braço direito mais perna esquerda;
- Derivação aVF – Eletrodo colocado na perna esquerda e em uma entrada negativa no braço esquerdo mais braço direito;
- Derivação V1 - Eletrodo colocado no quarto espaço intercostal direito, rebordo esternal;
- Derivação V2 - Eletrodo colocado no quarto espaço intercostal esquerdo, rebordo esternal;
- Derivação V3 - Eletrodo colocado a meio caminho entre V2 e V4;
- Derivação V4 - Eletrodo colocado no quinto espaço intercostal, linha médio-clavicular;
- Derivação V5 - Eletrodo colocado no mesmo plano de V4 (mesma altura), linha axilar anterior;
- Derivação V6 - Eletrodo colocado no mesmo plano de V4 e V5, linha axilar média.

3.3.2 Características Elétricas do Sinal Cardíaco

Os sinais elétricos das atividades cardíacas são capturados por meio de eletrodos dispostos em locais específicos pela eletrocardiografia (TORTORA; DERRICKSON, 2014). As faixas de frequência capturadas variam conforme sua aplicação como, por exemplo, ECG padrão de 12 derivações que possuem bandas com variação de 0,05-100 Hz e o ECG dinâmico que possui variação de 0,5 – 50 Hz. As bandas utilizadas do sinal de ECG para diferentes

aplicações representadas na Figura 9 são compostas por três frequências: Frequência cardíaca, monitoramento e clínico (TOMPKINS, 2000).

Figura 9 – Banda utilizada do sinal de ECG para diferentes aplicações.



Fonte: Adaptado de TOMPKINS (2000)

3.4 Monitoramento Prolongado do ECG

O monitoramento do ECG é essencial para a detectação precoce e prevenções de doenças cardíacas, garantindo que haja uma maior eficiência na preceituação de medicamentos ao paciente (CHEE; SEOW, 2007). Além disto, o monitoramento é fundamental, pois algumas alterações cardíacas ocorrem principalmente durante o período de sono ou perante mudanças no estado mental, emocional ou físico do paciente (KADISH *et al.*, 2001).

Os sistemas de monitoramento de ECG sofreram diversas modificações e são atualmente aplicados em diferentes ambientes como em hospitais, residências, ambulatórios além do contexto remoto, utilizando uma ampla gama de tecnologias (SERHANI *et al.*, 2020). Assim, os sistemas de monitoramentos de ECG caminham para sistemas sem fios combinados a tecnologias informáticas avançadas e a ferramentas de inteligência artificial (IA), melhorando o serviço de saúde e facilitando o monitoramento da funcionalidade cardíaca (COSOLI *et al.*, 2021).

Existem duas maneiras de realizar os registros de ECG: Por meio dos registros contínuos conhecidos como Monitoramento de Holter ou eletrocardiografia dinâmica (os quais são realizados normalmente em um período de 24 horas ou 48 horas) e por meio dos registros intermitentes ou conhecidos como monitores de eventos ou gravadores de loop, realizados por um longo período para fornecer registros breves e sendo em alguns casos incorporados a um

loop de memória que captura eventos sintomáticos. Esta prática é utilizada normalmente em pacientes que apresentam sintomas infrequentes (KADISH *et al.*, 2001).

3.4.1 Eletrocardiografia Dinâmica

A eletrocardiografia dinâmica ou Monitoramento de Holter se baseiam no registro de uma ou duas derivações das 12 do eletrocardiograma clínico padrão, para seu monitoramento, de modo a analisar ocorrências de distúrbios (TOMPKINS, 2000). Tais registros são realizados normalmente em um período de 24 ou 48 horas (KADISH *et al.*, 2001).

Assim, a eletrocardiografia dinâmica vem sendo cada vez mais utilizada nas últimas duas décadas, principalmente no diagnóstico de arritmias cardíacas. Isto se deve às limitações do ECG convencional em detectar este tipo de doença, uma vez que a mesma pode ocorrer em qualquer horário do dia. Diante disto, a eletrocardiografia dinâmica tem sido utilizada no diagnóstico de distúrbios cardíacos, frequência ou condução em paciente com queixas cardiovasculares como palpitações, dor torácica ou sensação de falta de ar ou até mesmo em distúrbios transitórios como tonturas e desmaios (FIEV *et al.*, 2019).

O primeiro dispositivo portátil de gravação de ECG ficou conhecido como Monitor Holter, por ter recebido o nome do seu criador, Dr. Norman Holter. O dispositivo foi desenvolvido em 1957 e se utilizava de fitas magnéticas para armazenar o ECG, sendo usado por vários anos, permitindo assim o registro do ECG a um baixo custo. Contudo, o uso de fitas magnéticas acabava causando distorções de fase e de frequência do sinal do ECG armazenado em alguns dispositivos devido às limitações de modulação e pela inteligência mecânica empregada no dispositivo (CRAWFORD *et al.*, 1999).

O Monitor de Holter convencional é frequentemente desconfortável, pois além de ser um dispositivo pesado (com vários eletrodos conectados ao tórax, dificultando o seu deslocamento pelo paciente e fazendo com que normalmente o monitoramento exija uma disponibilidade maior do paciente em ficar no hospital), o aparelho ainda apresenta um alto custo, não sendo sofisticado se levada em consideração a tecnologia moderna (MAHDY *et al.*, 2018).

Em virtude dos avanços tecnológicos, os equipamentos de ECG utilizam agora gravações de estado sólido. Isto permite que as gravações passem direto do sinal ECG em formato digital, evitando as distorções anteriormente registrada pelas fitas magnéticas usadas pelo Monitor de Holter. Diante disto, as tecnologias com formato digital permitem uma análise mais rápida do ECG, porém necessitam de uma grande memória para a gravação dos dados

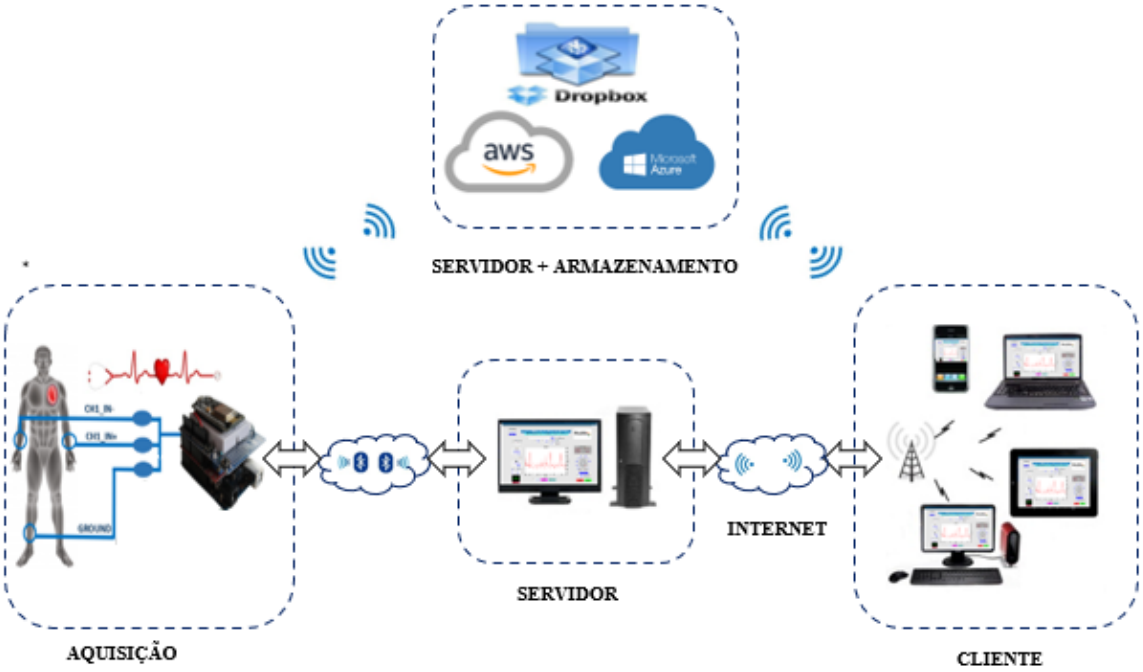
(CRAWFORD *et al.*, 1999).

No entanto, o surgimento da internet, do **Wi-Fi**, da transmissão de banda larga e do **smartphone** móvel aliado a esta nova capacidade digital levou à criação de tecnologias com custo-efetivo mais rápidos para o Monitor de Holter. Um exemplo é a criação de aplicativos Holter para **smartphone** que podem se tornar um instrumento móvel de diagnóstico fisiológico, prognóstico, terapêutico e de vigilância com arquivamento (KENNEDY, 2013).

4 METODOLOGIA

Neste trabalho, é proposto um sistema IoT para aquisição, monitoramento, processamento, armazenamento em nuvem e visualização de sinais de ECG. A arquitetura proposta dispõe de algoritmos para a detecção não supervisionada dos principais ruídos que corrompem sinais de ECG, além de uma aplicação WEB para mediar a interação entre o usuário e o sistema de aquisição. A Figura 10 apresenta um esquemático detalhado do sistema proposto.

Figura 10 – Arquitetura do sistema proposto.



Fonte: Próprio Autor.

A arquitetura do sistema apresentado na Figura 10 é formada por: Um biossensor ECG-EMG Olimex, um Arduíno UNO R3, um módulo *WI-FI* ESP8266, um módulo *blueto-oth* HC-05, uma unidade central de processamento (CPU) e uma aplicação *WEB* que pode ser hospedada em um servidor de uma máquina local ou em um servidor de uma máquina virtual de um serviço de computação em nuvem. O biossensor é utilizado para detectar o sinal eletrocardiográfico e em seguida condicioná-lo em um sinal elétrico analógico. Já o Arduíno UNO R3 é responsável por gerar uma representação digital do sinal analógico proveniente do biossensor e enviar para uma aplicação *WEB* hospedada em uma máquina local ou em um serviço de computação em nuvem. A comunicação entre o Arduíno UNO e a aplicação *WEB* pode ocorrer de três formas: Via comunicação serial, *bluetooth* ou *Wi-Fi*. A aplicação *WEB* contempla algoritmos responsáveis por processar, monitorar e armazenar os sinais de ECG coletados. Além

disto, a aplicação *WEB* dispõe de algoritmos para detecção não supervisionada dos principais ruídos que corrompem sinais de ECG.

Neste capítulo, serão apresentados conceitos gerais sobre os dispositivos que compõem o *hardware* do sistema (seção 4.1). Na seção 4.2, serão apresentados os procedimentos executados para o desenvolvimento da aplicação *WEB*. Por fim, a seção 4.3 apresentará um detalhamento sobre o desenvolvimento dos algoritmos para processamento e detecção não supervisionada dos principais ruídos que corrompem sinais de ECG.

4.1 Hardware do dispositivo móvel de aquisição

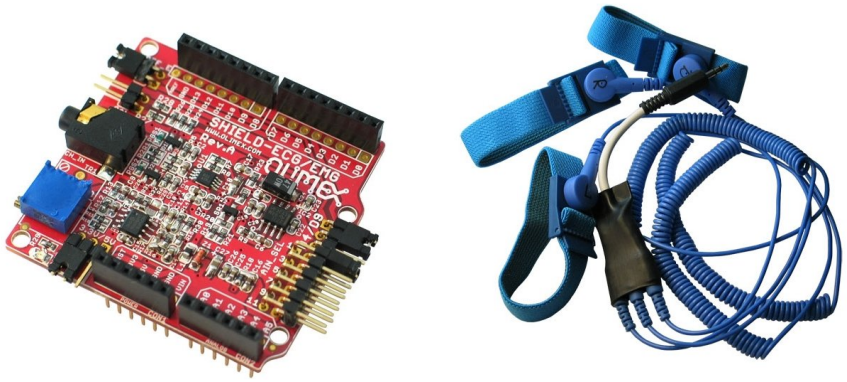
O dispositivo utilizado no processo de aquisição de sinais de ECG é formado por uma placa microcontroladora Arduíno UNO R3, um módulo WI-FI ESP8266, um módulo *bluetooth* HC-05 e um biossensor ECG-EMG Olimex. Para a escolha dos componentes que compõem o *hardware* do dispositivo móvel de aquisição, foram levados em consideração os seguintes critérios: Apresentar um baixo consumo de energia, baixo custo, possuir *hardware* livre e proporcionar uma prototipagem rápida.

Dentre os diversos microcontroladores com *hardware* livre e baixo consumo de energia, o Arduíno UNO R3 foi escolhido devido sua construção mecânica e elétrica adequadas para a interligação com o biossensor OLIMEX ECG-EMG por meio de encaixe. Para estabelecer uma troca de dados entre o dispositivo móvel de aquisição e a aplicação *WEB*, são necessários módulos adicionais para realizar esta comunicação, levando em consideração um baixo consumo de energia e um baixo custo. Para o sistema proposto, foi utilizado um módulo para se conectar à internet com o objetivo de realizar coletas *wireless* de forma *online* e um módulo *bluetooth* para coletas *wireless* de forma *offline*. Os módulos HC-05 e ESP8266, além de atenderem a todos os critérios estabelecidos, são capazes de ser configurados e programados de forma prática, utilizando a IDE (do inglês: I*ntegrated Development Environment*) do Arduíno. Nas subseções seguintes, serão apresentados conceitos gerais sobre os componentes utilizados no *hardware* do dispositivo móvel de aquisição. Além disto, será apresentado o passo a passo da montagem do *hardware*.

4.1.1 Biossensor ECG-EMG Olimex

Como os sinais brutos de ECG são pré-amplificados dentro dos eletrodos que ficam dispostos sobre a pele do paciente, estes sinais, portanto, precisam ser condicionados em um nível adequado para digitalização e processamento(OLIMEX, 2014). Para realizar esse condicionamento, foi utilizado a *shield* ECG-EMG desenvolvida pela empresa Olimex LTDA. A Figura 11 apresenta uma foto do biossensor ECG-EMG Olimex juntamente com os eletrodos utilizados no processo de aquisição.

Figura 11 – Biossensor ECG-EMG Olimex.



Fonte: (AMAZON, 2015).

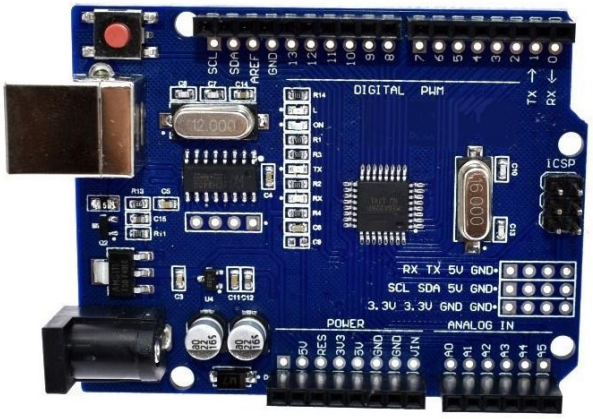
O dispositivo apresentado na Figura 11 é responsável por converter a diferença de potencial gerada pela atividade elétrica cardíaca e pela atividade muscular em um sinal elétrico analógico anexado às suas entradas $CH1_IN+/CH1_IN-$ e em um único fluxo de dados como saída(OLIMEX, 2020). O sinal de saída é analógico e deve ser discretizado com o objetivo de aplicar técnicas de processamento digital. Normalmente, isto é feito via conversor analógico digital (ADC) dedicado, embutido em placas Arduíno de 3.3 e 5V. No presente trabalho, foi utilizado o conversor ADC de 10 bits de um Arduíno UNO R3 para discretizar o sinal de ECG bruto coletado pelo biossensor ECG-EMG Olimex.

4.1.2 Arduíno UNO R3

Arduíno UNO R3 é uma placa de prototipagem eletrônica baseada no microcontrolador ATmega328P de 8 bits, com um *clock* máximo de 16 MHz, memória SRAM de 2KB e uma memória *flash* de 32KB(ATMEL, 2015). Esta placa possui 14 pinos de entrada/saída digitais (dos quais seis podem ser usados como saídas PWM), uma conexão USB, um conector

de alimentação, um botão de *reset*, e um conector ICSP (do inglês: In *Circuit Serial Programmer*) para possibilitar a programação do microcontrolador sem removê-lo do circuito (ARDUINO.CC, 2021). A Figura 12 apresenta uma foto de um Arduíno UNO R3.

Figura 12 – Arduino UNO R3.



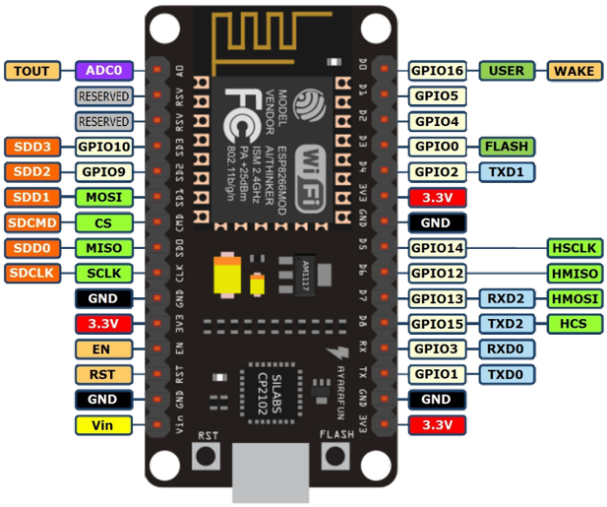
Fonte: (JOOM, 2022).

A placa apresentada na Figura 12 contempla seis entradas analógicas com conversor digital-analógico de 10 bits, no qual é possível estabelecer uma resolução de 4,8mV em uma referência de 5V. No presente trabalho, foi utilizada uma destas entradas analógicas para obter a representação discreta (no domínio do tempo) do sinal analógico detectado pelo biossensor, considerando uma taxa de amostragem de 256 Hz.

4.1.3 Módulo WIFI ESP8266 NodeMcu

O ESP8266 consiste em um microcontrolador com *Wi-Fi* integrado e desenvolvido pelo fabricante chinês Espressif Systems. Ele é formado por um microprocessador RISC L106 de 32 bits baseado no Tensilica, com um *clock* máximo de 160 MHz, memória RAM de 20KB e uma memória *flash* SPI externa para armazenar programas do usuário de até 32 MB e recurso *Wi-Fi* 802.11 b/g/n. (ESPRESSIF-SYSTEMS, 2022). A Figura 13 ilustra um microcontrolador ESP8266.

Figura 13 – Módulo Wi-Fi ESP8266.



Fonte: (FATEC, 2022).

O dispositivo apresentado na Figura 13 apresenta recursos de memória aproximadamente 100 vezes maiores do que outros microcontroladores de baixo custo como, por exemplo, o microprocessador ATmega328 usado em Placa Arduíno UNO. Assim, o ESP8266 pode ser usado para desenvolver um *hardware* de aquisição de dados IoT com mais facilidade do que placas baseadas em Arduíno. No presente trabalho, foi utilizado o ESP8266 para intermediar a troca de dados entre o Arduíno UNO R3 e a aplicação *WEB*. A troca de dados entre o ESP8266 e a aplicação da *WEB* ocorre via *Wi-Fi*, utilizando o protocolo de comunicação MQTT (do inglês: *Message Queue Telemetry Transport*). A programação do ESP8266 foi escrita em linguagem de programação C e C++ executada a partir do ambiente de desenvolvimento integrado (IDE) do Arduíno.

4.1.4 Comunicação MQTT

As aplicações em que há necessidade de comunicação em tempo real exigem o uso de protocolos leves que permitam altas taxas de transferência em baixa latência. Assim, o protocolo MQTT, desenvolvido por Andy Stanford-Clark e Arlen Nipper em 1999, se aplica às características exigidas neste contexto. Este protocolo se encontra na camada de aplicação da pilha TCP/IP e é caracterizado por permitir conexões assíncronas (HIVEMQ, 2015; MORAES, 2020).

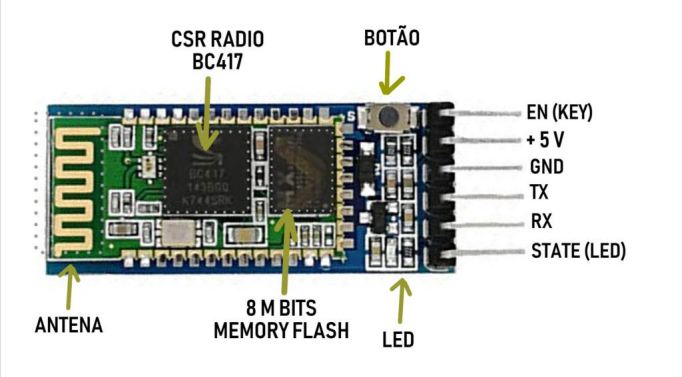
No sistema proposto, a comunicação *WI-FI* utiliza o protocolo de comunicação MQTT para envio de dados do dispositivo móvel para a aplicação *WEB*. Com isto, a troca de dados entre clientes se baseia no modelo de mensagem *publish/subscribe*. Este modelo possui

três componentes: Publicador(es), serviço de eventos e assinante(s). Os assinantes informam ao barramento de eventos que desejam receber uma determinada notificação e este roteia para os consumidores interessados, as notificações geradas pelos publicadores. Neste sistema de integração, a topologia básica de rede é estruturada em estrela, sendo o *broker* o elemento que gerencia o fluxo de dados da rede. Este é encarregado pelo roteamento e encaminhamento da mensagem entre o cliente emissor e o de destino. As mensagens são classificadas em tópicos e entregues aos clientes que tenham interesse. No sistema proposto, foi utilizado o *broker* *Mosquitto.org* para gerenciar o fluxo de dados da rede. Este é encarregado pelo roteamento e encaminhamento da mensagem entre o dispositivo móvel (cliente emissor) e a aplicação *WEB* (cliente receptor). As mensagens são classificadas em tópicos e entregues aos clientes inscritos nos tópicos.

4.1.5 Módulo bluetooth HC-05

O módulo *bluetooth* HC-05 é um módulo SPP (do inglês: *Serial Port Protocol*) projetado para transferir dados serialmente do controlador ou PC por meio de uma configuração de conexão serial sem fio. Ele detecta a tensão de nível de 3,3V para transmissão ou recepção do microcontrolador e possui dois modos de operação como modo de dados e modo de comando(BALAKRISHNA; RAJESH, 2022). O *bluetooth* HC-05 se trata de uma tecnologia de rádio frequência que opera na frequência de nível global sem restrições ISM (do inglês: *Industrial, Scientific, and Medical*). Com o auxílio do pino de chave, podemos configurar este módulo utilizando o conjunto de instruções de comando AT para definir nome, senha e selecionar a taxa de transmissão(SAIRAM *et al.*, 2002). A Figura 14 apresenta a foto de um módulo *bluetooth* HC-05.

Figura 14 – Módulo bluetooth HC-05.



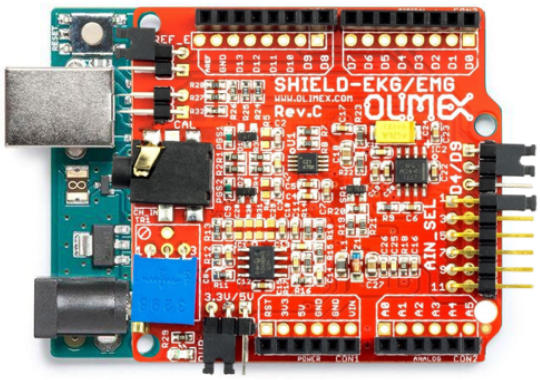
Fonte: (MóDULO-ELETRôNICA, 2018).

O dispositivo apresentado na Figura 14 foi utilizado para enviar os sinais de ECG bruto coletados para a aplicação *WEB*. Na configuração da senha, na taxa de transmissão, e no nome do módulo *bluetooth* HC-05 foi utilizado o *software Config HC*. Com o auxilio da caixa de ferramentas *MathWorks Bluetooth Toolbox*, foi possível estabelecer uma comunicação *wireless* entre o módulo *bluetooth* supracitado e a aplicação *WEB*, com uma taxa de transferência de dados de 57600 bits por segundo (bps). Para que a comunicação *bluetooth* ocorra, é necessário que a CPU que hospeda a plicação *WEB* tenha a tecnologia de conexão *wireless bluetooth* embutida. Caso a CPU não tenha a tecnologia *bluetooth* embutida, é possível habilitar essa funcionalidade utilizando um adaptador e receptor *bluetooth USB*.

4.1.6 Montagem do hardware

A *shield* ECG-EMG Olimex possui uma padronização elétrica e mecânica que possibilita sua interligação com um Arduíno UNO R3 por meio de encaixe. Logo, o primeiro passo da montagem do *hardware* consiste na conexão entre a *shield* ECG-EMG Olimex e o Arduíno UNO R3, conforme apresentado na Figura 15.

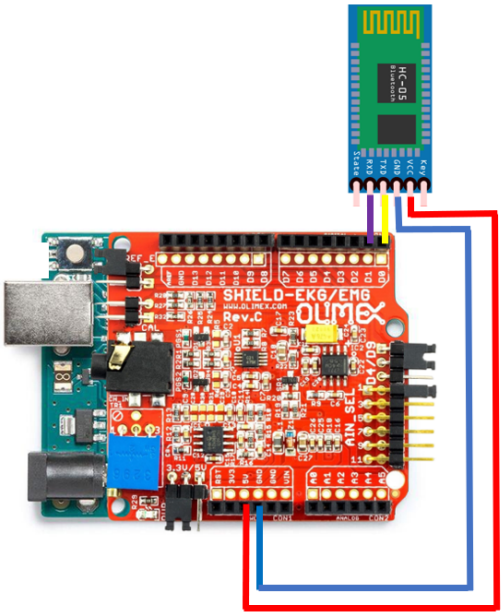
Figura 15 – Conexão entre a *shield* ECG-EMG Olimex e o Arduíno UNO R3.



Fonte: Próprio Autor.

O segundo passo da montagem do *hardware* consiste na integração do módulo *bluetooth* HC-05 com o conjunto apresentado na Figura 15. Para esta integração, foi considerado o esquemático de ligação apresentado na Figura 16.

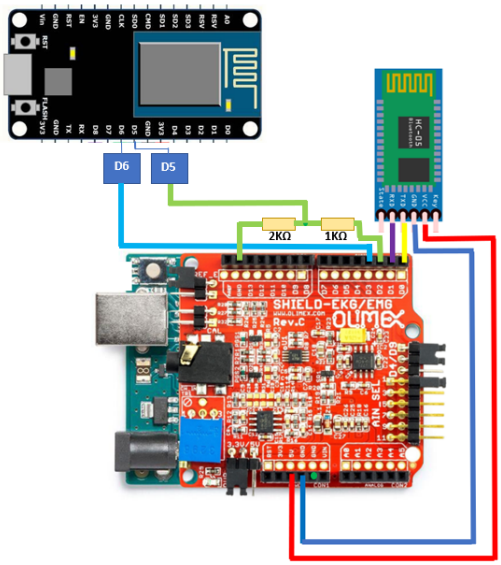
Figura 16 – Conexão entre a *shield* ECG-EMG Olimex, Arduino UNO R3, e o módulo *bluetooth* HC-05.



Fonte: Próprio Autor.

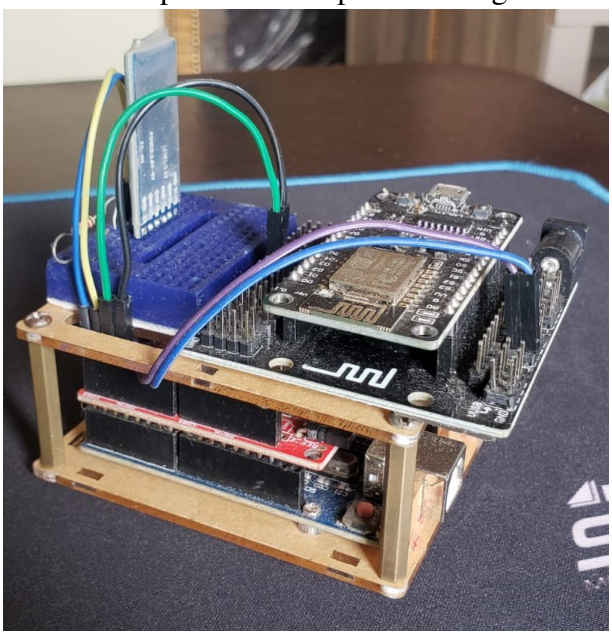
O terceiro e último passo da montagem do *hardware* consiste na ligação elétrica entre o conjunto apresentado na Figura 16 com o módulo *Wi-Fi* ESP8266. Nesta ligação elétrica foram acrescentados dois resistores: Um de 1 $K\Omega$ e outro de 2 $K\Omega$, conforme apresentado na Figura 17. Estes dois resistores foram instalados como divisor de tensão, substituindo assim o conversor de nível bidirecional. Já a Figura 18 apresenta uma foto do *hardware* do dispositivo móvel montado.

Figura 17 – Conexão entre a *shield* ECG-EMG Olimex, Arduíno UNO R3, módulo *bluetooth* HC-05 e módulo *Wi-Fi* ESP8266.



Fonte: Próprio Autor.

Figura 18 – *Hardware* do dispositivo IoT após a montagem.



Fonte: Próprio Autor.

Como pode ser observado na Figura 18, foi adicionado uma case para proteção do biossensor, além de uma placa de expansão para conectar o módulo ESP8266 e possibilitar a alimentação dos dispositivos do *hardware*.

4.2 Desenvolvimento da Aplicação *WEB*

A arquitetura do sistema proposto contempla uma aplicação *WEB* desenvolvida com o auxílio do ambiente de desenvolvimento de aplicativos executáveis e aplicativos *WEB* do MATLAB ®. Este ambiente de desenvolvimento é denominado de MATLAB *App Designer*. Nele, é possível desenvolver aplicativos de alta qualidade apenas movendo os componentes visuais para definir o *design* da sua interface gráfica com o usuário (GUI), utilizando o editor integrado para programar rapidamente seu comportamento(MATHWORKS, 2020a). O que motivou a escolha da ferramenta MATLAB *App Designer* para desenvolver a aplicação *WEB* foi a disponibilidade de amplas ferramentas para implementação de técnicas de processamento digital de sinais, comunicação serial, *bluetooth* e MQTT. Além disto, a vasta experiência do autor com o uso do MATLAB foi um fator decisivo. Nesta aplicação *WEB*, foram implementados algoritmos e protocolos de comunicação com o objetivo de agregar as seguintes funcionalidades ao sistema: Coletar, armazenar em nuvem, processar e monitorar os sinais de ECG e sua FFT (do inglês: *Fast Fourier Transform*). A troca de informações entre a plicação *WEB* e o *hardware* apresentado na seção 4.1, ocorre via protocolo de comunicação serial ou via tecnologia *wireless*

(*bluetooth* ou Wi-Fi).

Os procedimentos realizados para criar, executar e acessar remotamente a aplicação *WEB* foram divididos em três etapas, conforme detalhadas abaixo:

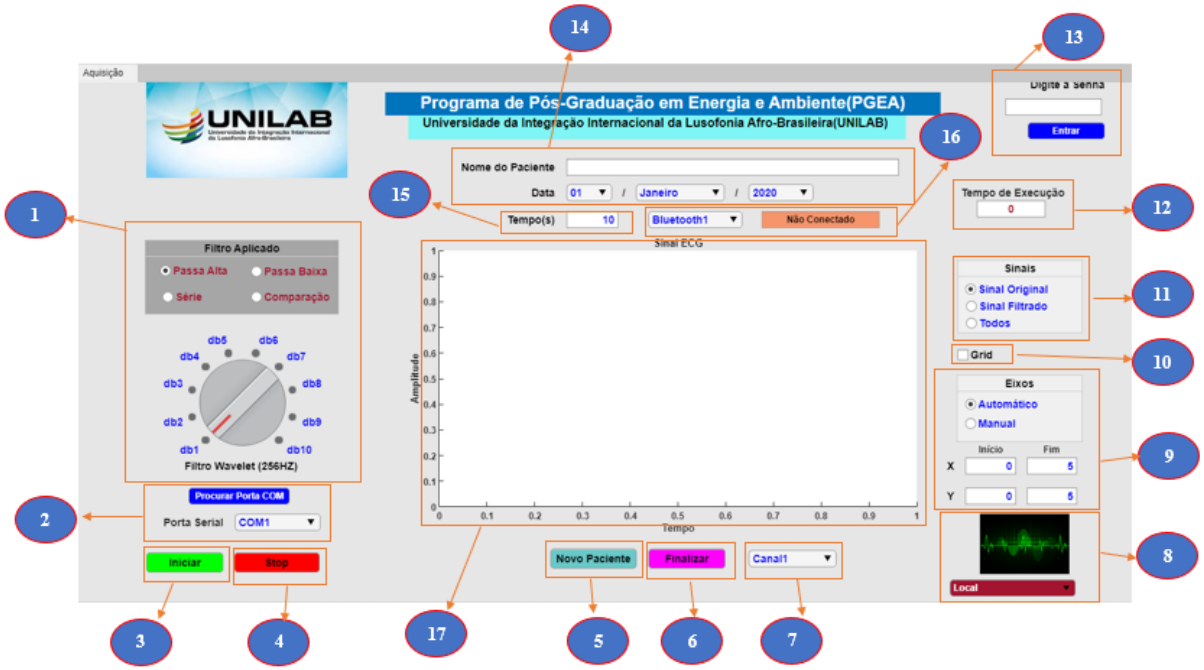
4.2.1 Etapa 1: Instalação do software MATLAB ®

O *software* MATLAB ® é uma plataforma de programação projetada com o objetivo de possibilitar engenheiros e cientistas analisarem e projetarem sistemas que utilizam a matemática computacional. Com o auxílio deste *software* foi possível desenvolver a interface da aplicação *WEB*, além de algoritmos para fins de processamento de sinais, comunicação serial, comunicação *wireless*, envio de dados para plataformas de armazenamento em nuvem e comunicação com o *hardware* apresentado na seção 4.1. No presente trabalho, foi utilizada a versão 2021b deste *software*. Os requisitos mínimos necessários para utilização da versão 2021b consistem em: 60 GB de capacidade de disco, 1 (um) GB de memória RAM, alocação de 1 (um) núcleo de processador, além da utilização de um sistema operacional *Windows* ou *Linux*.

4.2.2 Etapa 2: Escrita da programação e desenvolvimento da interface gráfica da aplicação WEB

Com o auxílio do MATLAB *App Designer*, foram desenvolvidas duas interfaces gráficas para aplicação *WEB* do sistema proposto. A primeira interface foi desenvolvida com o objetivo de auxiliar o usuário no processo de aquisição e monitoramento dos sinais de ECG. Para isto, foram inseridos blocos com o objetivo de selecionar o tempo de aquisição, nome do paciente, visualizar o tempo de aquisição, selecionar a porta de comunicação serial, escolher o modo de comunicação e aplicar um filtro passa baixa para verificar se os sinais que estão sendo coletados não são ruídos de alta frequência. Esta primeira interface foi desenvolvida para ser usada por pacientes no processo de coleta e armazenamento dos registros cardiológicos e por profissionais da área de saúde para monitorar remotamente os pacientes. Logo, o modelo da tela foi projetado para possuir blocos com as funcionalidades obrigatórias para configurar uma coleta via serial, *WI-FI* e *bluetooth*. Além disto, para um armazenamento dos registros em um banco de dados colaborativo foi necessário limitar o tempo de aquisição e identificar o paciente ao qual pertence o registro. Isto foi o que motivou a inserção dos demais blocos na tela da primeira interface. A Figura 19 mostra a interface gráfica da tela inicial da aplicação *WEB* desenvolvida com o auxílio do MATLAB ®*App Designer*.

Figura 19 – Interface gráfica da tela inicial da aplicação *WEB*.



Fonte: Próprio Autor.

Cada bloco inserido na interface gráfica apresentada na Figura 19 recebeu uma programação com o objetivo de proporcionar diferentes funcionalidades na aplicação *WEB*. A programação implementada em cada bloco foi escrita em linguagem de programação C e C++ com o auxílio do ambiente de programação *MATLAB Coder*. A seguir é apresentado um detalhamento sobre a funcionalidade proporcionada pela programação implementada em cada conjunto de blocos enumerados na Figura 19:

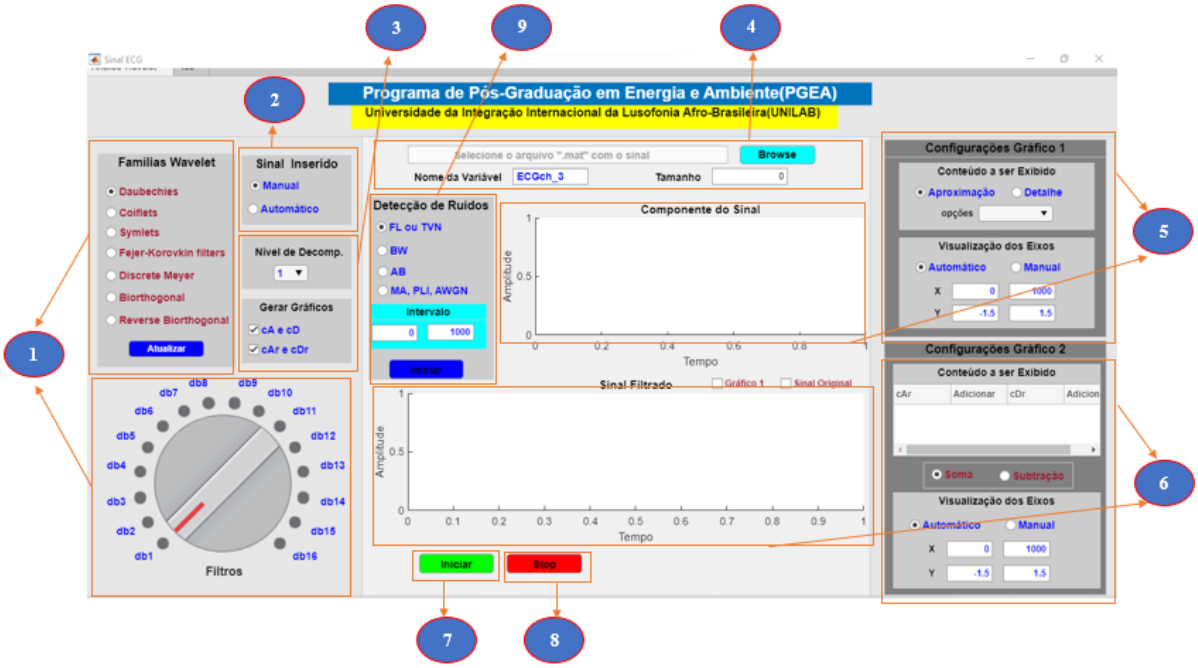
1. Conjuntos de blocos que permitem ao usuário implementar em tempo real um filtro passa alta, filtro passa baixa ou comparar os resultados de um filtro passa alta com um filtro passa baixa. Tudo isto utilizando um filtro *Daubechies wavelet*.
2. Conjuntos de blocos que permitem o usuário selecionar de forma manual ou automatizada a porta de comunicação serial em que o dispositivo móvel está conectado;
3. Botão para iniciar o processo de aquisição de sinais de ECG após a conexão com o *hardware* do dispositivo móvel de aquisição;
4. Botão para interromper o processo de aquisição de sinais de ECG, caso seja necessário terminar a aquisição antes do tempo definido;
5. Botão para limpar os dados inseridos do paciente anterior, caso seja necessário inserir os dados de um novo paciente;
6. Botão para para encerrar o aplicativo;
7. *Drop Down* Utilizado para possibilitar ao usuário selecionar se quer visualizar em tempo

real o sinal coletado ou a FFT (do inglês: *Fast Fourier Transform*) do sinal coletado;

8. Conjunto de blocos utilizados para possibilitar que o usuário escolha o local onde deseja salvar o arquivo contendo os dados do paciente, sinal de ECG bruto e sinal filtrado. O cliente pode fazer o *download* dos dados coletados ou enviá-los para uma plataforma de armazenamento em nuvem;
9. Conjunto de bloco que possibilita selecionar qual o intervalo do sinal que o usuário deseja plotar;
10. *Check box*: Utilizado para mostrar ou ocultar as linhas de grade de eixos;
11. Conjunto de blocos utilizado para selecionar qual o sinal que o usuário irá acompanhar em tempo real: Sinal original ou sinal filtrado;
12. *Edit Field*: Utilizado para mostrar a contagem do tempo no decorrer de um processo de aquisição;
13. Conjunto de blocos destinado à inserção de senha para liberar o acesso à aplicação *WEB*. As funcionalidades do aplicativo *WEB* só serão liberadas após a inserção da senha correta;
14. Conjunto de blocos destinado à inserção de dados do paciente tais como nome completo e dada da coleta do sinal. No final da coleta, o nome do paciente e data são inseridos automaticamente no arquivo que contem o sinal coletado;
15. *Edit Field*: Utilizado para mostrar tempo de coleta do sinal de ECG;
16. Conjunto de blocos utilizado para possibilitar ao usuário selecionar o tipo de comunicação (serial, *bluetooth Wi-Fi*) entre o *hardware* dispositivo móvel e a aplicação *WEB*;
17. Ambiente de plotagem dos sinais coletados e dos sinais obtidos a partir da aplicação de uma técnica de processamento;

O designer da segunda interface foi criado com objetivo de aplicar técnicas de processamento digital nos sinais durante o processo de coleta ou em qualquer sinal salvo em formato ".*m*". Para isso, foi pensado em uma tela que permita que especialistas da área de processamento digital de sinais de ECG consigam analisar os sinais no domínio do tempo ou no domínio da frequência sem a necessidade de programação. Com isso, auxiliar especialistas no processo de detecção de padrões de doenças cardiovasculares a partir de informações sobre a morfologia e o comportamento do sinal analisado. Na segunda interface, foram aplicadas diversas ferramentas processamento digital, detecção de ruídos, análise e visualização dos sinais coletados. A Figura 20 mostra o resultado da interface gráfica da segunda tela desenvolvida para a aplicação *WEB*.

Figura 20 – Interface gráfica da segunda tela da aplicação *WEB*.



Fonte: Próprio Autor.

Cada bloco inserido na interface gráfica apresentada na Figura 20 recebeu uma programação com o objetivo de proporcionar diferentes funcionalidades na aplicação *WEB*. A programação implementada em cada bloco foi escrita em linguagem de programação C e C++ com o auxílio do ambiente de programação *MATLAB Coder*. A seguir é apresentado um detalhamento sobre a funcionalidade proporcionada pela programação implementada em cada conjunto de blocos enumerado na Figura 20:

1. Conjunto de blocos que possibilita ao usuário aplicar uma decomposição *wavelet* baseada em funções que formam a família *Wavelet*: *Daubechies, Biorthogonal, Coiflets, Symlets, Discrete Meyer, e Reverse Biorthogonal*;
2. Conjunto de blocos que possibilita ao usuário escolher se o sinal vai ser inserido automaticamente (a partir do processo de aquisição) ou se o sinal vai ser inserido manualmente;
3. Conjunto de blocos que possibilita ao usuário determinar qual o nível da decomposição *wavelet* que será aplicado;
4. Conjunto de blocos que possibilita ao usuário carregar o sinal que será inserido manualmente;
5. Conjunto de blocos que possibilita ao usuário plotar as componentes de detalhe e de aproximação do sinal decomposto;
6. Conjunto de blocos que possibilita ao usuário reconstruir e plotar um sinal a partir da adição dos componentes de frequências desejados;

7. Botão para interromper a execução do processamento aplicado pelos blocos da segunda tela da aplicação *WEB*;
8. Botão para iniciar o processamento aplicado pelos blocos da segunda tela da aplicação *WEB*;
9. Conjunto de blocos que possibilita ao usuário aplicar os algoritmos de detecção não supervisionada dos principais ruídos que corrompem sinais de ECG;

4.2.3 Etapa 3: Instalação do MATLAB Web App Server e hospedagem da aplicação WEB

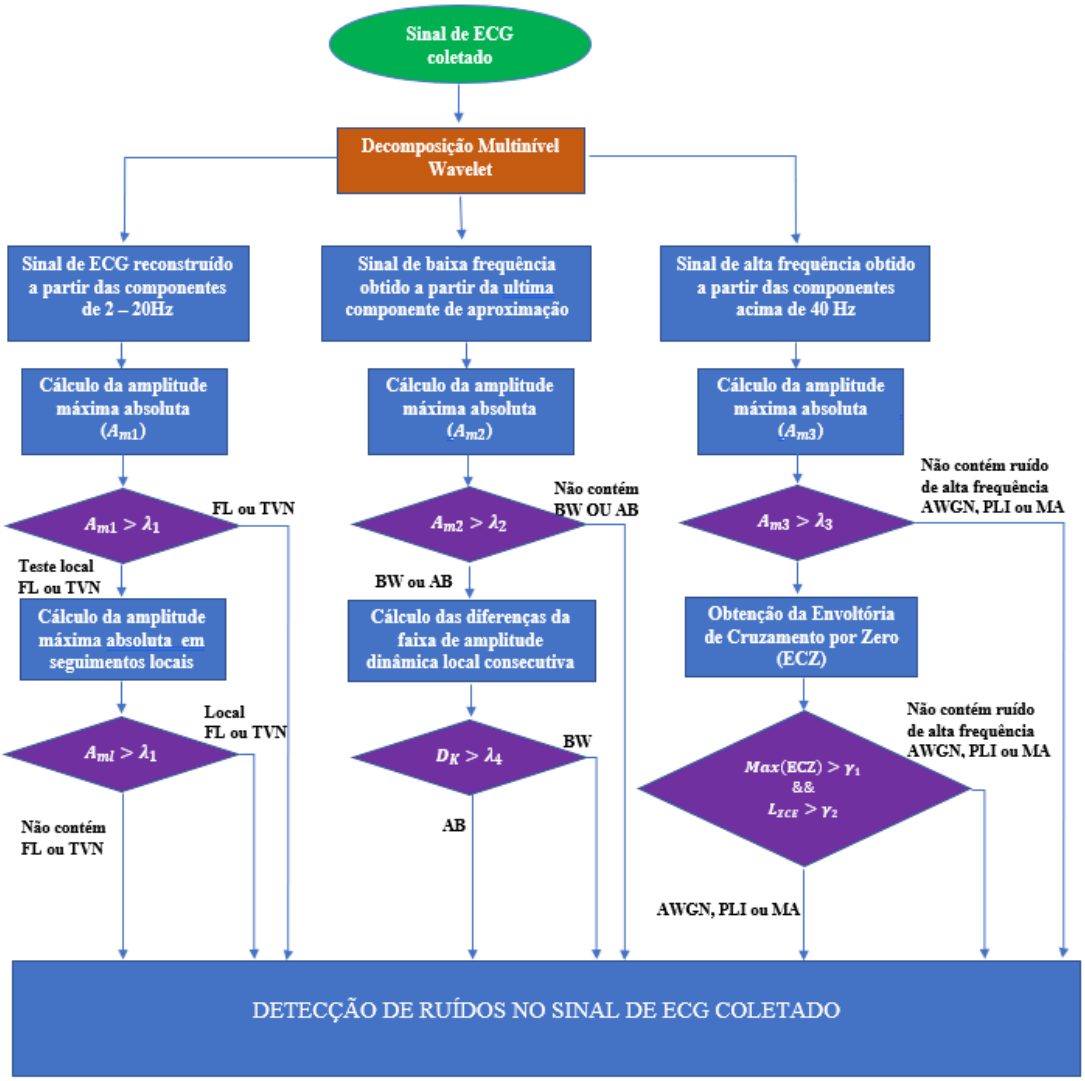
O *MATLAB Web App Server* ™ consiste em um *software* desenvolvido com o objetivo de hospedar aplicativos do MATLAB e aplicativos interativos da *WEB*. Para isto, é necessário que os aplicativos sejam desenvolvidos no *MATLAB Compiler* ™(MATHWORKS, 2020b). *O MATLAB Web App Server*™ pode ser executado em uma máquina local ou em uma máquina virtual de um serviço de computação em nuvem. Uma vez hospedada, a aplicação *WEB* pode ser controlada de forma remota a partir de um navegador *WEB*. Esta aplicação pode ser acessada por meio de dispositivos conectados à rede local ou a uma rede externa. Para acessar à aplicação *WEB* por meio de uma rede local é necessário que o *MATLAB Web App Server* ™ esteja sendo executado em uma máquina conectada à mesma rede do dispositivo que deseja acessar a aplicação. Já para acesso remoto por meio de dispositivos conectados a uma rede externa é necessário que a máquina que hospeda o App esteja conectada a um roteador e que este roteador esteja configurado para direcionar o acesso de uma porta externa para o servidor local onde o App está hospedado.

4.3 Desenvolvimento dos algoritmos para detecção não supervisionada dos principais ruídos que corrompem sinais de ECG

O sistema proposto dispõe de algoritmos de processamento de sinais para detecção e identificação de ruídos nos sinais de ECG coletados. Os algoritmos desenvolvidos e implementados no sistema utilizam a metodologia de detecção e identificação não supervisionada de ruídos apresentada em Satija *et al.* (2018). Dentre os principais tipos de ruídos que corrompem sinais de ECG, é possível detectar e identificar de forma não supervisionada os seguintes ruídos: FL (do inglês: *Flat Line*), TVN (do inglês: *Time-Varying Noise or Pause*), BW (do inglês: *Baseline Wander*), AB (do inglês: *Abrupt Change*), MA (do inglês: *Muscle Artifacts*), PLI (do inglês:

Power Line Interference) e AWGN (do inglês: *Additive White Gaussian Noise*). Os algoritmos implementados detectam de forma não supervisionada ruídos de curta e de longa duração sem a necessidade de detecção do complexo QRS, além de não exigirem fases de aprendizagem antes de sua utilização. A Figura 21 apresenta o fluxograma do método utilizado para detecção de ruídos nos sinais de ECG. O método de detecção de ruídos foi implementado em três etapas: Decomposição *wavelet* multinível, reconstrução simultânea do sinal e do ruído de ECG e detecção de ruídos com base na extração de características das componentes de detalhe e de aproximação do sinal decomposto. As seções seguintes apresentam um breve detalhamento sobre o passo a passo do desenvolvimento do algoritmo de detecção dos principais ruídos que corrompem sinais de ECG.

Figura 21 – Fluxograma do método de detecção de ruídos.



Fonte: Próprio Autor.

4.3.1 Análise wavelet

A análise *wavelet* é uma ferramenta matemática bem adequada para sinais não estacionários e transitórios. Sua aplicação permite decompor um sinal em um conjunto de formas de onda ortogonais localizadas tanto no domínio do tempo quanto no domínio da frequência. A localização temporal de componentes espectrais pode ser obtida por análise *wavelet* multiresolução, a qual fornece a representação tempo-frequência do sinal(SATIJA *et al.*, 2018; BOUAZIZ *et al.*, 2014). Matematicamente, a transformada contínua *wavelet* consiste na operação de convolução do sinal a ser analisado $x(t)$ e a família de funções *wavelet*, $\psi(t)$:

$$T_{m,n} = \frac{1}{\sqrt{n}} \int_{-\infty}^{+\infty} x(t)\, \psi^* \left(\frac{t-n}{m} \right) dt \tag{4.1}$$

Onde $\psi^*(t)$ é o conjugado complexo da *wavelet* mãe $\psi(t)$, que é deslocada por um tempo n e dilatada ou contraída por um fator m antes de calcular sua correlação com o sinal $x(t)$.

A transformada *wavelet* é uma transformação linear que decompõe um sinal em componentes que aparecem em diferentes escalas (ou resolução). A Transformada Wavelet Discreta (TWD) proporciona uma análise multinível por meio da decomposição do sinal em várias sub-bandas de frequência sucessivas(BANERJEE *et al.*, 2012). A TWD utiliza dois conjuntos de funções $\phi(t)$ e $\psi(t)$ que estão associadas aos filtros passa-baixa e passa-alta respectivamente.

$$\phi(t) = \sum_n h[n]\phi(2t-n) \tag{4.2}$$

$$\psi(t) = \sum_n g[n]\phi(2t-n) \tag{4.3}$$

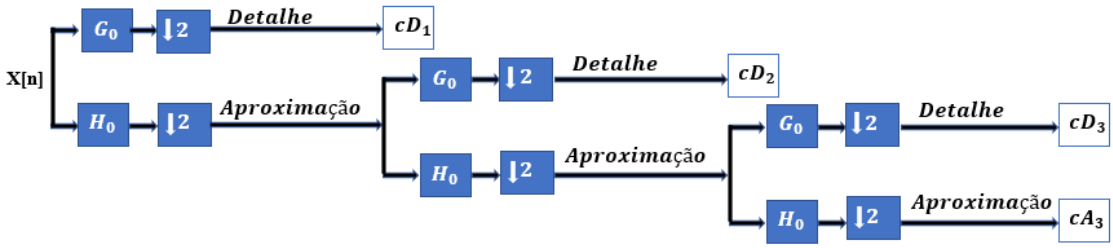
O sinal de domínio de tempo original $x(t)$ amostrado a 1000 amostras/s forma um sinal de tempo discreto $x[n]$, o qual é primeiro passado por meio de um filtro passa-alta de meia banda $g[n]$ e um filtro passa-baixa $h[n]$. A filtragem reduz a resolução pela metade, mas deixa a escala inalterada. O sinal é então subamostrado por dois. Este procedimento pode ser expresso matematicamente como (POLIKAR, 2004; BANERJEE *et al.*, 2012):

$$cA[k] = Y_{LOW}[k] = \sum_n x[n] \cdot h[2k-n] \tag{4.4}$$

$$cD[k] = Y_{HIGH}[k] = \sum_{n} x[n] \cdot g[2k-n] \tag{4.5}$$

O filtro passa-baixa gera a componente de aproximações (cA), enquanto o passa-alta gera a componente de detalhes (cD). Usando o algoritmo de Mallat, a TWD pode ser implementada utilizando o banco de filtros. A decomposição *wavelet* multinível do sinal em sua forma discreta $x[n]$ usando bancos de filtros é ilustrada na Figura 22.

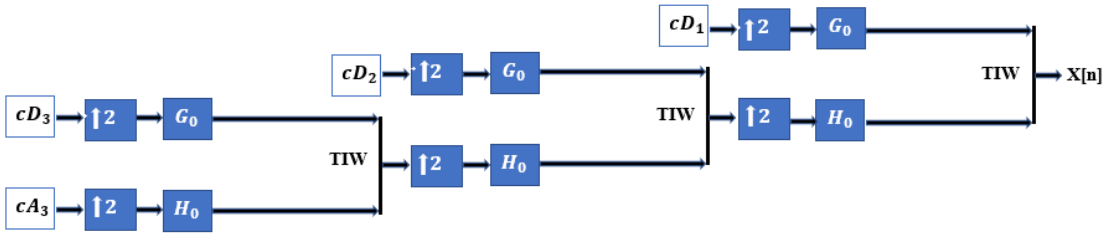
Figura 22 – Decomposição *wavelet* multinível utilizando um banco de filtro em cascata.



Fonte: Próprio Autor.

O sinal original pode ser reconstruído a partir das componentes de detalhes e aproximação. Para isto, é necessário aplicar a transformada inversa de *wavelet* (TIW). O esquemático do processo de reconstrução do sinal original está apresentado na Figura 23:

Figura 23 – Processo de reconstrução do sinal original a partir das componentes de detalhe e aproximação.



Fonte: Próprio Autor.

O processo de reconstrução apresentado na Figura 23 pode ser representado da seguinte forma:

$$x[n] \sum_{i=1}^{L} cD_i[n] + A_L[n] \tag{4.6}$$

Onde:

L = Número de decomposições;

$cD_i[n]$ = Componente de detalhe do i_{simo} nível de decomposição;

$A_L[n]$ = Componente de aproximação do último nível de decomposição.

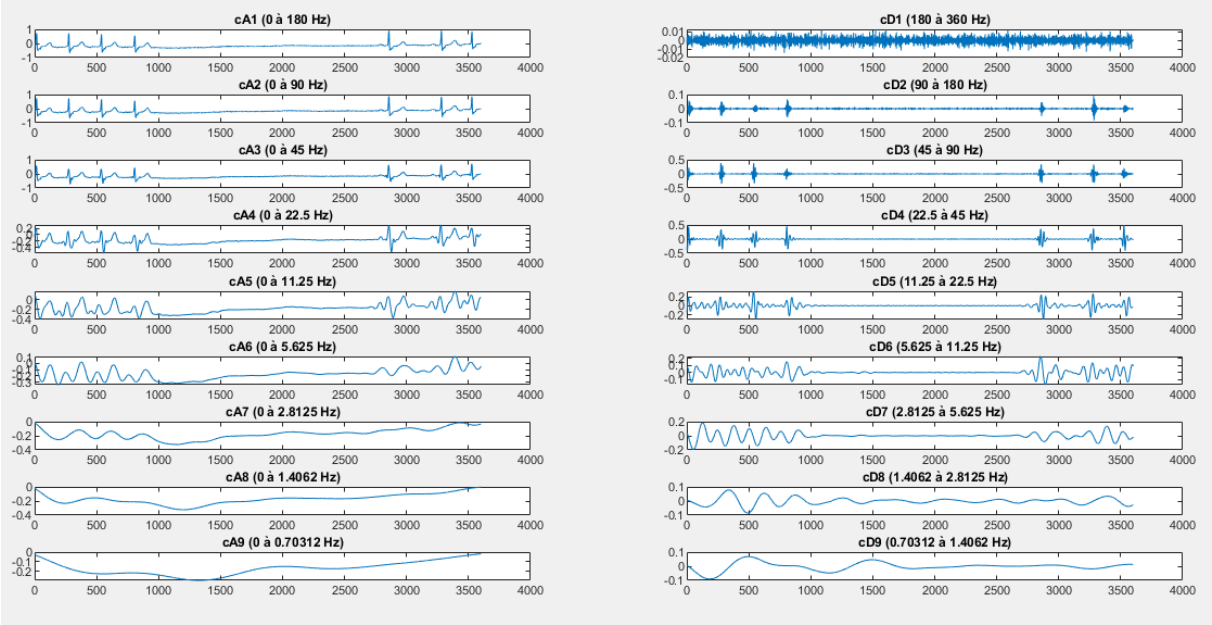
As duas primeiras etapas do método de detecção de ruídos (Decomposição *Wavelet* Multinível e reconstrução simultânea do sinal e ruído de ECG) são obtidas a partir da análise *Wavelet*. Com o auxílio das bibliotecas de análise *Wavelet* do *software MATLAB coder* foram desenvolvidos algoritmos que implementam a decomposição *Wavelet* multinível e reconstrução do sinal original. Para teste e validação dos algoritmos desenvolvidos, foram utilizados sinais do banco de dados do *MIT-BIH Arrhythmia Database*.

4.3.2 Detecção de ruídos FL ou TVN

O ruído de *Flat Line* (FL) consiste em uma componente CC nos sinais fisiológicos capturados. Na literatura, o ruído FL é considerado como uma linha CC de amplitude zero. Quando os dispositivos microcontrolados saturam ou apresentam problemas na comunicação, os registros de ECG coletados ficam cheios de pausa. Estas pausas são classificadas como ruídos *Time-Varying Noise or Pause* (TVN). Elas caracterizam a ausência de informação no sinal e podem ser de curta ou longa duração que variam com o tempo(SATIJA *et al.*, 2018).

O Sinal do registro 232 do banco de dados *MIT-BIH Arrhythmia Database* possui ruídos FL e TVN, por isto logo ele foi utilizado para teste e validação dos algoritmos. Como se trata de um sinal amostrado a uma frequência de 360 Hz, foi aplicada uma decomposição *wavelet* de nove níveis. A Figura 24 apresenta as componentes de detalhes e de aproximação após a aplicação da decomposição *wavelet* multinível no registro contaminado com FL e TVN (registro 232).

Figura 24 – Componentes de aproximação e de detalhe de uma decomposição multinível aplicada ao registro 232.



Fonte: Próprio Autor.

A maior parte das informações das ondas locais de ECG se situa entre 2 e 20 Hz. Assim, a aproximação do sinal de ECG é reconstruída usando as sub-bandas cD5, cD6 e cD7.

$$\widehat{x}(n) = cD5 + cD6 + cD7 \tag{4.7}$$

Onde $\widehat{x}(n)$ é o sinal de ECG reconstruído e cD5, cD6, cD7 são as componentes de detalhe contidas na faixa de frequência 2 a 20 Hz. O próximo passo consiste no cálculo da amplitude máxima absoluta (A_{m1}). Caso a amplitude máxima seja menor que o limiar pré-definido experimentalmente $\lambda_1 = 0.1$, pode-se concluir que o sinal é formado apenas por FL ou TVN. O valor de λ_1 foi obtido com base na amplitude mínima da onda local do ECG (ou seja, onda P). Se a amplitude máxima for maior que λ_1,deve-se realizar uma análise para verificar se não há trechos locais do sinal de ECG com a presença de FL ou TVN.

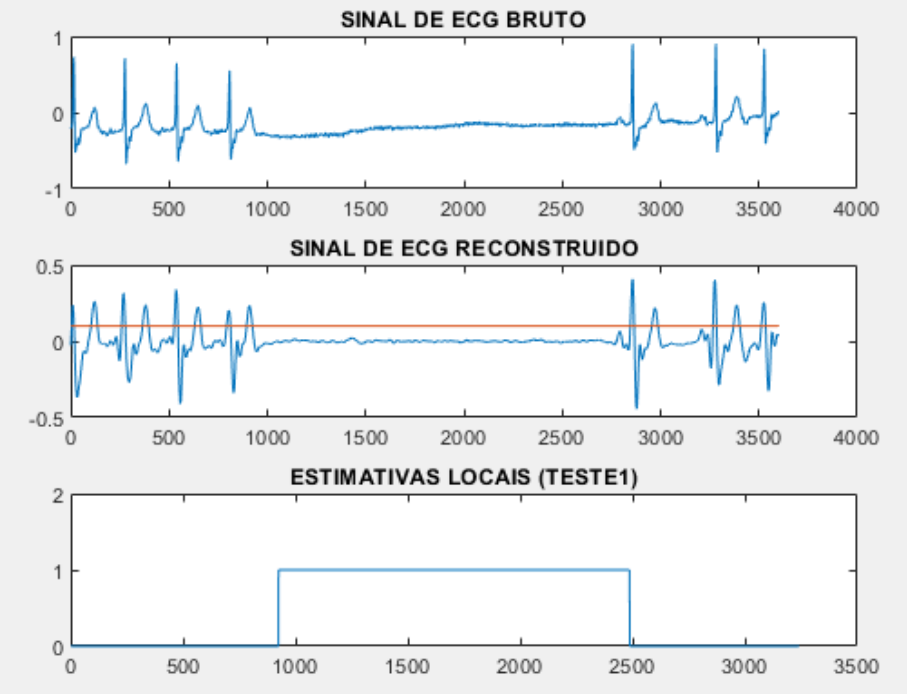
Para detecção local de FL ou TVL foi implementado uma janela de tamanho de 1 (um) segundo, e, em seguida foi utilizada esta janela para percorrer o sinal de ECG de 1 (um) em uma amostra. Logo após, foi calculada a amplitude absoluta local (A_{ml}). Se a amplitude máxima A_{ml} for menor do que o limiar λ_1, caracteriza que o trecho de 11 segundos testado é formado por FL ou TVN. O vetor $TESTE_1$ apresentado em (4.8) assume valor 1 (um) quando for detectado

ruídos FL ou TVN e valor zero caso contrário.

$$TESTE_1 = \begin{cases} 1 & se \;\; A_{ml} < \lambda_1 \\ 0 & CASO \;\; CONTRARIO \end{cases} \tag{4.8}$$

A Figura 25 apresenta os resultados da aplicação dos algoritmos de detecção de FL e TVN no registro 232 do banco *MIT-BIH Arrhythmia Database*. Nela, é possível observar o sinal original e o sinal de ECG reconstruído após a decomposição multinível e as estimativas locais de detecção de FL ou TVN.

Figura 25 – Resultados da aplicação dos algoritmos no registro 232 do banco *MIT-BIH Arrhythmia Database*.



Fonte: Próprio Autor.

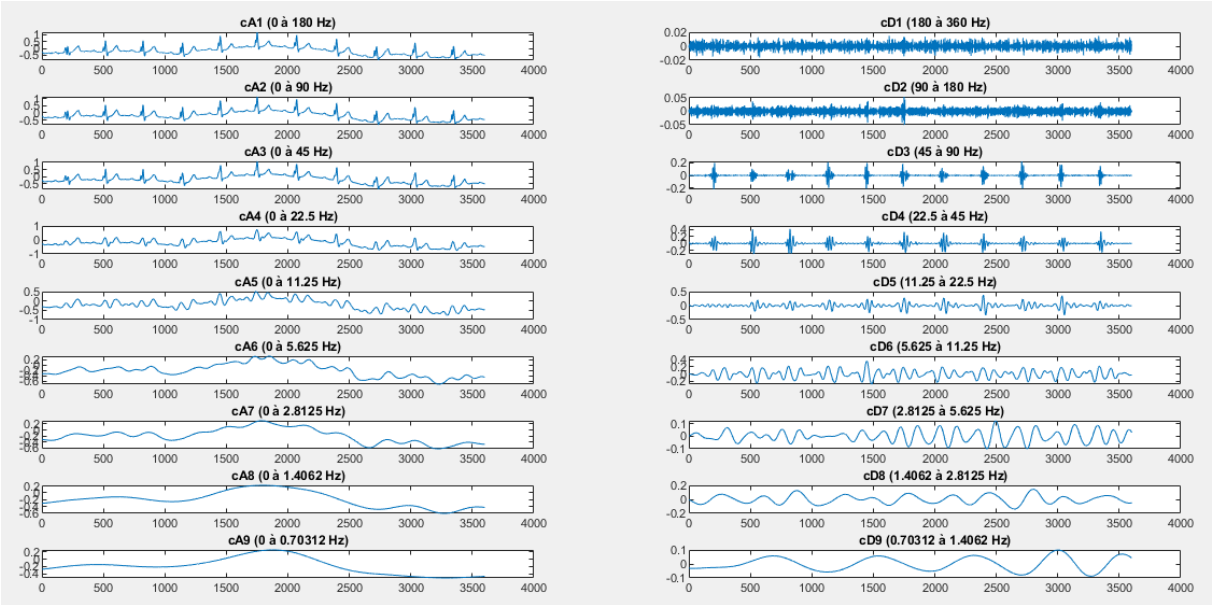
4.3.3 Detecção de ruídos BW ou AB

Os ruídos *Baseline Wander* (BW) e *Abrupt Change* (AB) consistem em ruídos de baixa frequência que afetam os sinais de ECG. A frequência máxima destes tipos de ruídos é menor do que 1 (um) Hz. Logo, as características deles podem ser detectadas a partir das componentes de um sinal decomposto que apresenta sub-bandas de frequência menores que 1 (um) Hz (SATIJA *et al.*, 2018).

Sinal de ECG do registro 111 do banco de dados *MIT-BIH Arrhythmia Database* possui ruídos BW localizados em alguns trechos. Sendo assim, este registro foi utilizado para

teste e validação do algoritmo de detecção de ruído BW. Como se trata de um sinal amostrado a uma frequência de 360 Hz foi aplicada uma decomposição *wavelet* de nove níveis. A Figura 26 apresenta as componentes de detalhes e de aproximação após a aplicação da decomposição *wavelet* multinível no registro contaminado com BW (registro 111).

Figura 26 – Resultados da aplicação dos algoritmos no registro 111 do banco *MIT-BIH Arrhythmia Database*.



Fonte: Próprio Autor.

Após a aplicação da decomposição multinível apresentada na Figura 26, consideramos $x_b(n)$ como um sinal reconstruído e formado pelas componentes de frequência menores do que 1 (um) Hz. Desta forma, foi obtida a equação:

$$\widehat{x}_b(n) = cA9 \tag{4.9}$$

onde $cA9$ consiste na componente de aproximação do último nível da decomposição. O próximo passo consiste no cálculo da amplitude máxima absoluta de $x_b(n)$. Caso a amplitude máxima seja maior do que o limiar pré-definido experimentalmente $\lambda_2 = 0.1$, pode-se concluir que o sinal está contaminado com BW. O valor de λ_2 foi obtido com base na amplitude mínima da onda P. O vetor $TESTE_2$ apresentado em (4.10) assume valor 1 (um) quando for detectado ruído BW e valor zero em caso contrário.

$$TESTE_2 = \begin{cases} 1 & se \quad A_{m2} > \lambda_2 \\ 0 & CASO \quad CONTRARIO \end{cases} \tag{4.10}$$

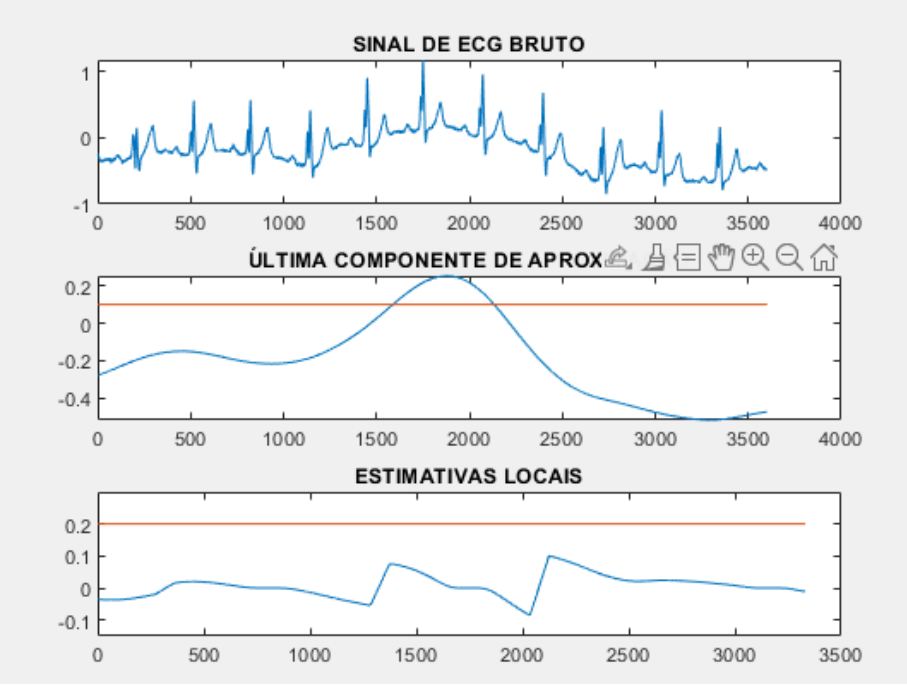
Onde A_{m2} é a amplitude máxima absoluta do sinal $\widehat{x}_b(n)$. Para detectarmos a presença de BW por meio de uma estimação local, devemos segmentar o sinal $x_b(n)$ em blocos sobrepostos de 500ms com um fator de sobreposição de 50%. Em seguida, é preciso calcular as diferenças da faixa de amplitude dinâmica local consecutiva, conforme representado em (4.11).

$$d(k) = max\{|\, u_{k+1} \,|\} - max\{|\, u_k \,|\} \tag{4.11}$$

Onde $d(k)$ é a diferença da faixa de amplitude dinâmica local consecutiva. Já o u_{k+1} e u_k são segmentos consecutivos de 500ms do sinal $x_b(n)$

A Figura 27 apresenta os resultados da aplicação dos algoritmos de detecção de BW no registro 111 do banco *MIT-BIH Arrhythmia Database*. Nela, é possível observar o sinal original, o sinal $\widehat{x}_b(n)$ reconstruído após a decomposição multinível e as estimativas locais de detecção de BW. A partir das estimavas locais, é possível determinar a gravidade de BW.

Figura 27 – Resultados da aplicação do algoritmo de detecção de ruído BW no registro 111 do banco *MIT-BIH Arrhythmia Database*.



Fonte: Próprio Autor.

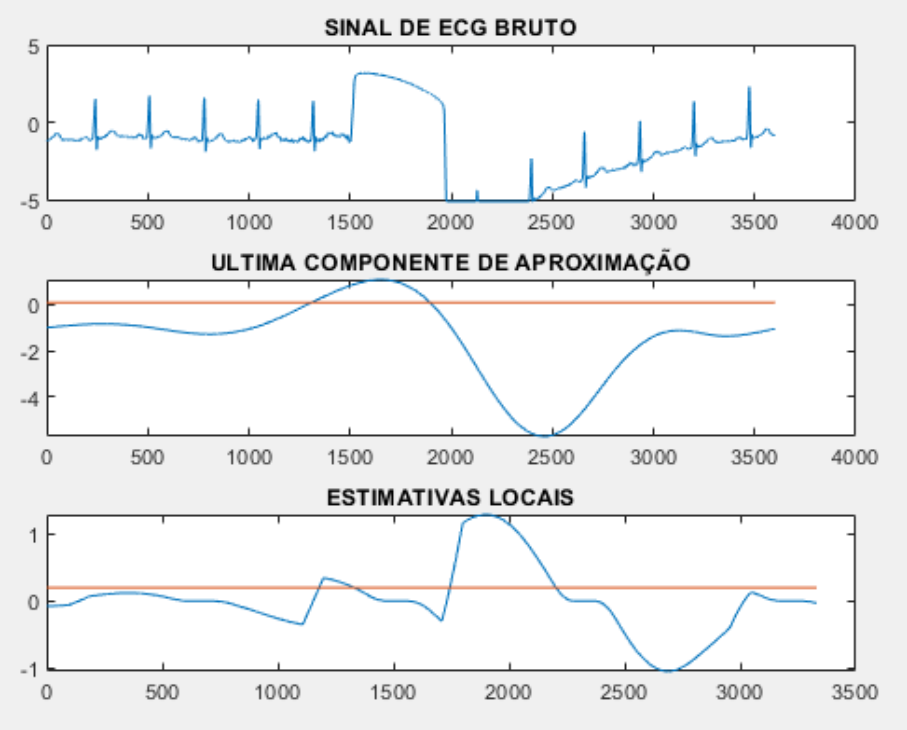
Como pode ser observado na Figura 27, as estimativas locais de BW não ultrapassam o valor de λ_2. Tanto BW como AB causam uma alteração de amplitude no sinal de ECG. Os mesmos procedimentos para detecção do ruído BW também se aplicam na detecção de ruídos AB.

A amplitude máxima absoluta de $d(k)$ nos permite classificar se o ruído detectado é AB ou BW. Experimentalmente foi obtido um limiar $\lambda_4 = 0.2$ para distinguir os ruídos BW e AB, conforme apresentado em (4.12):

$$\Gamma = \begin{cases} AB & se \quad max\{|\, d(k)\}\,| > \lambda_4 \\ BW & CASO \;\; CONTRARIO \end{cases} \tag{4.12}$$

O sinal de ECG do registro 116 do banco de dados *MIT-BIH Arrhythmia Database* possui ruídos AB localizados em alguns trechos. Logo esse registro foi utilizado para teste e validação do algoritmo de detecção de ruído AB. A Figura 28 apresenta os resultados da aplicação dos algoritmos de detecção de AB no registro 116 do banco *MIT-BIH Arrhythmia Database*. Nela, é possível observar o sinal original, o sinal $\widehat{x}_b(n)$ reconstruído após a decomposição multinível e as estimativas locais de detecção de AB.

Figura 28 – Resultados da aplicação do algoritmo de detecção de ruído AB no registro 116 do banco *MIT-BIH Arrhythmia Database*.



Fonte: Próprio Autor.

A partir da comparação entre as Figuras 27 e 28, podemos observar que as estimativas locais de AB ultrapassam o dobro de λ_2, enquanto as estimativas locais de BW não ultrapassam sequer o valor deste limiar.

4.3.4 Detecção de ruídos de alta frequência (MA, PLI ou AWGN)

Muscle Artifacts (MA), *Power Line Interference* (PLI) e *Additive White Gaussian Noise* (AWGN) são ruídos de alta frequência que corrompem sinais de ECG, comprometendo com isto a precisão de um diagnóstico médico. A maior parte da energia dos ruídos de alta frequência está acima de 40 Hz. Logo podemos decompor um sinal e analisar as componentes acima de 40 Hz para detectar ruídos de alta frequência(SATIJA *et al.*, 2018). Considerando $X_h(k)$ como sendo o sinal formado por todas as componentes acima de 40 Hz, temos:

$$X_h(n) = \sum_{i=1}^{l} cD_i(n) \tag{4.13}$$

Onde $X_h(n)$ é o sinal formado pelas componentes acima de 40 Hz, cD_i é a i-ésima decomposição TWD e l é o número de níveis contendo o sinal de frequência acima de 40 Hz. Para um sinal amostrado a uma frequência de 360 Hz, temos l=3. Para detectar a presença de ruído de alta frequência, calculamos o número local de cruzamentos por zero (CZ). De modo análogo às estimativas locais apresentadas nas seções anteriores, temos que segmentar o sinal que será analisado. Para esta análise, foi utilizada uma janela de tamanho de 50ms para percorrer o sinal $X_h(n)$ com um deslocamento de 1 (uma) amostra. Para cada segmento de sinal formado pelas amostras contidas na janela de tamanho 50ms, foi calculado o valor médio de amplitude. Caso o valor médio de amplitude de um segmento seja superior ao limiar $\lambda_3 = 0,002mV$, temos que calcular o número de cruzamento por zero (CZ) deste respectivo segmento. Caso contrário, será contabilizado como sendo o número de cruzamentos por zero. O valor de λ_3 foi estimado experimentalmente com base na amplitude mínima da onda P. O vetor z apresentado em (4.14) armazena o valor de CZ para cada seguimento do sinal $X_h(n)$

$$z(k) = \begin{cases} CZ \quad se \quad \frac{1}{M}\sum_{k=1}^{M} X_h(n) \; > \; \lambda_3 \\ 0 \quad CASO \;\; CONTRARIO \end{cases} \tag{4.14}$$

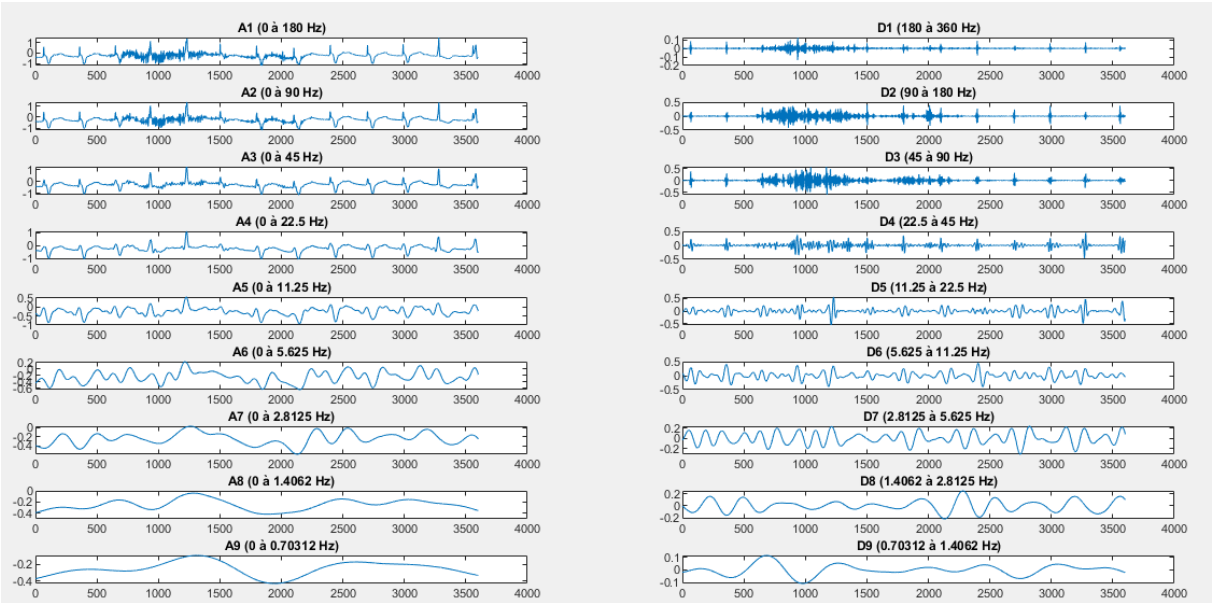
Onde M é o número de amostra em cada seguimento de 50ms. Após a obtenção do vetor z, aplicamos uma filtragem para obter a envoltória de cruzamento por zero (ECZ). O vetor $TESTE_3$ apresentado em (4.15) assume valor 1 (um) quando for detectado ruídos MA, PLI ou AWGN e valor zero caso ocorra o contrário (SATIJA *et al.*, 2018).

$$TESTE_3 = \begin{cases} 1 & se \ \ (max\{ECZ\} > \gamma_1) \ e \ (W_{ECZ} > \gamma_2) \\ 0 & CASO \ \ CONTRARIO \end{cases} \tag{4.15}$$

Onde W_{ECZ} consiste na largura do lóbulo de ECZ que é comparada com o limiar γ_1. Aqui consideramos o valor de γ_1 como sendo 300 ms, baseado no intervalo RR do período refratário mínimo. Já o valor do limiar γ_2 foi obtido empiricamente e assume valor igual a 4.

O sinal do registro 104 do banco de dados *MIT-BIH Arrhythmia Database* possui ruídos de alta frequência. Logo ele foi utilizado para teste e validação dos algoritmos. Como se trata de um sinal amostrado a uma frequência de 360 Hz, foi aplicado uma decomposição *wavelet* de nove níveis. A Figura 29 apresenta as componentes de detalhes e de aproximação após a aplicação da decomposição *wavelet* multinível no registro contaminado com ruídos de alta frequência (registro 104).

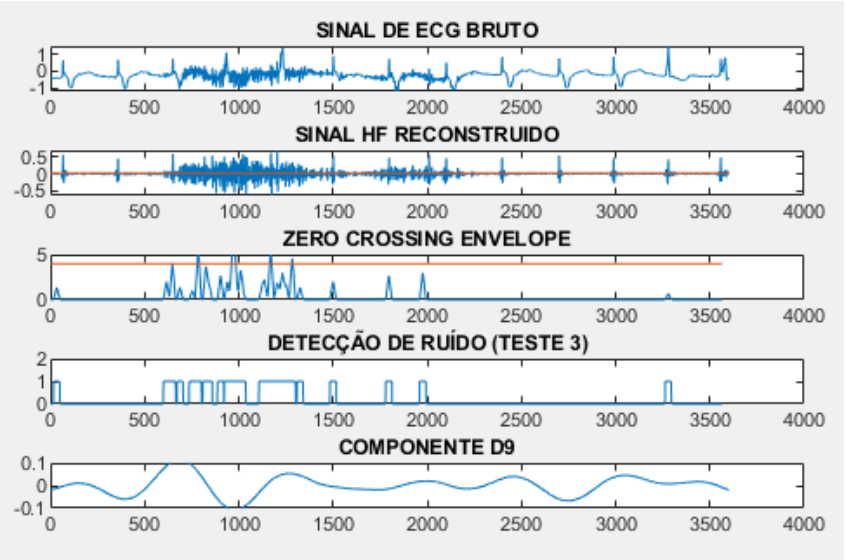
Figura 29 – Resultados da aplicação dos algoritmos no registro 104 do banco *MIT-BIH Arrhythmia Database*.



Fonte: Próprio Autor.

Já a Figura 30 apresenta os resultados da aplicação dos algoritmos de detecção de ruídos de alta frequência no registro 104 do banco *MIT-BIH Arrhythmia Database*. Nela, é possível observar o sinal original, o sinal que contem as frequências acima de 40 Hz reconstruído após a decomposição multinível e as estimativas locais de detecção de ruídos de alta frequência.

Figura 30 – Resultados da aplicação do algoritmo de detecção de ruídos de alta frequência no registro 104 do banco *MIT-BIH Arrhythmia Database*.



Fonte: Próprio Autor.

5 RESULTADOS E DISCUSSÕES

Nesta seção, serão discutidos os principais resultados inerentes ao sistema proposto. Na subseção 5.1, são apresentados os parâmetros e orçamentos do sistema; em seguida são apresentados os parâmetros de energia (subseção 5.2). Na subseção 5.3, serão mostrados os parâmetros de coleta dos registros cardiográficos e utilização de recursos a partir de parcerias com projetos de pesquisa da Universidade Federal do Ceará (UFC). Já na subseção 5.4, apresentaremos os testes de operação do sistema, enquanto na subseção 5.5 será apresentada a análise de desempenho da aplicação *WEB*. Por fim, na subseção 5.6 são apresentadas comparações entre o sistema proposto e trabalhos correlatos.

5.1 Parâmetros e orçamento do sistema

O orçamento do sistema proposto é mostrado na Tabela 1, considerando o custo em reais (BRL) e em dólares americanos (USD). Como resultado do uso de um serviço de computação em nuvem para estudantes, não consideramos o custo do serviço de máquina virtual e rede em nuvem para hospedar as aplicações *WEB*. Além disto, para utilizar a funcionalidade de coleta de sinais via *WI-FI*, fez-se necessária a contratação de um serviço de computação e armazenamento em nuvem.

Tabela 1 – Orçamento do sistema

Componente	Custo (BRL)	Custo (USD)
ESP8266+ Placa de Expansão	R$ 43,82	U$ 8,10
Shild ECG-EMG Olimex com eletrodos	R$ 162,95	U$ 30,12
Arduíno UNO R3	R$ 44,85	U$ 8,29
Módulo HC-05	R$ 12,88	U$ 2,38
Power bank 10000mAH	R$ 48,90	U$ 9,04
Fonte chaveada 9v	R$ 13,47	U$ 2,49
Custos adicionais	R$ 30	U$ 5,54
Custo Total	R$ 356,87	U$ 65,96

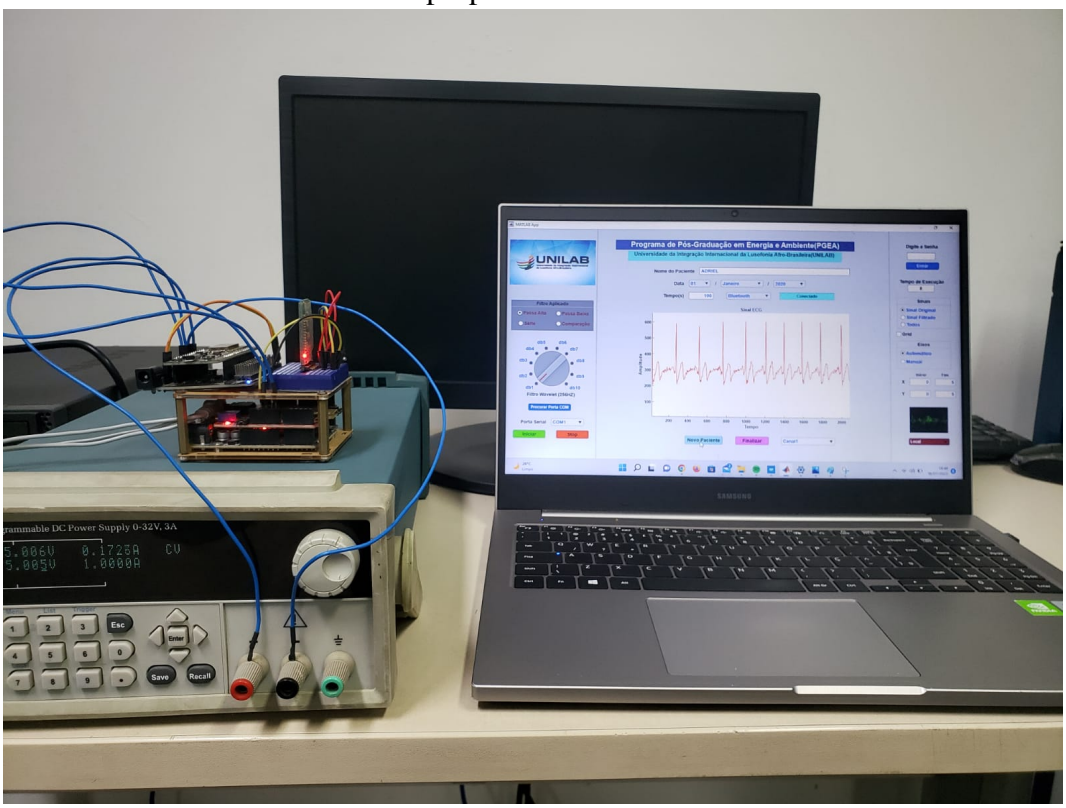
Fonte: Próprio autor (2022).

Os valores apresentados na Tabela 1 consideraram valores de mercado para os componentes em julho de 2022. Foi considerado 1 USD= 5,41 BRL e um custo referente às características de *hardware* para uma coleta via comunicação: Serial, *bluetooth* e *Wi-Fi*.

5.2 Parâmetros de Energia

Para medir o consumo de energia, configuramos nosso dispositivo para coletar amostras de dados, considerando uma frequência de amostragem de 256 Hz. Medimos um sinal de 7 segundos e atualizamos a medição a cada 10 segundos (o que significa que o *hardware* do sistema realizou 6 medições por minuto). Nos testes de consumo, o sistema foi alimentado com uma fonte de tensão variável e, com o auxílio de um multímetro de bancada, foi possível verificarmos o consumo médio de corrente. A Figura 31 apresenta uma foto do experimento realizado para verificar o consumo de energia do dispositivo.

Figura 31 – Testes de bancada para verificar o consumo de energia durante o funcionamento do sistema proposto.



Fonte: Próprio autor (2022).

Considerando o consumo de corrente, o *hardware* do sistema apresentou uma entrada média de corrente de 180 miliamperes (mA). Em relação à média corrente e tensão, o consumo de energia estimado do dispositivo móvel foi de 0,90 Watt-hora (Wh), menor do que o consumo de um *Raspberry Pi* (1,4 Wh). Considerando um consumo médio de energia de 0,90 Wh, um *power bank* de 10000 mAh leva aproximadamente 55 horas para descarregar.

5.3 Parâmetros de coleta dos registro cardiográficos e utilização de recursos

Em parceria com o projeto de pesquisa "Caracterização do Perfil da Variabilidade da Frequência Cardíaca em Indivíduos com Covid-19 e Associação com Variáveis Inflamatórias e Prognóstico Clínico", o sistema proposto foi utilizado no Hospital Universitário Walter Cantídio da Universidade Federal do Ceará (HUWC-UFC). Foram coletados registros eletrocardiográficos de 50 pacientes hospitalizados em fase ativa da doença Covid-19. Cada registro coletado possui uma duração de 5 minutos e juntos formam um banco de dados para estudos colaborativos com o objetivo de identificar a existência de um padrão de alteração na variabilidade da frequência cardíaca. Para utilização do sistema em pacientes do hospital universitário foi concedido pelo Comitê de Ética um parecer consubstanciado. Com Certificado de Apresentação de Apreciação Ética (CAAE) Nº 47229221.9.0000.5045

Em parceria com o projeto de pesquisa "Sensoriamento e Monitoramento Remoto da Vitalidade Fetal", foram utilizados recursos de computação em nuvem da AWS para hospedar a aplicação *WEB* desenvolvida em uma máquina virtual e então realizar testes experimentais de comunicação entre o dispositivo móvel e a aplicação *WEB* hospedada em uma máquina virtual.

5.4 Testes de operação do sistema

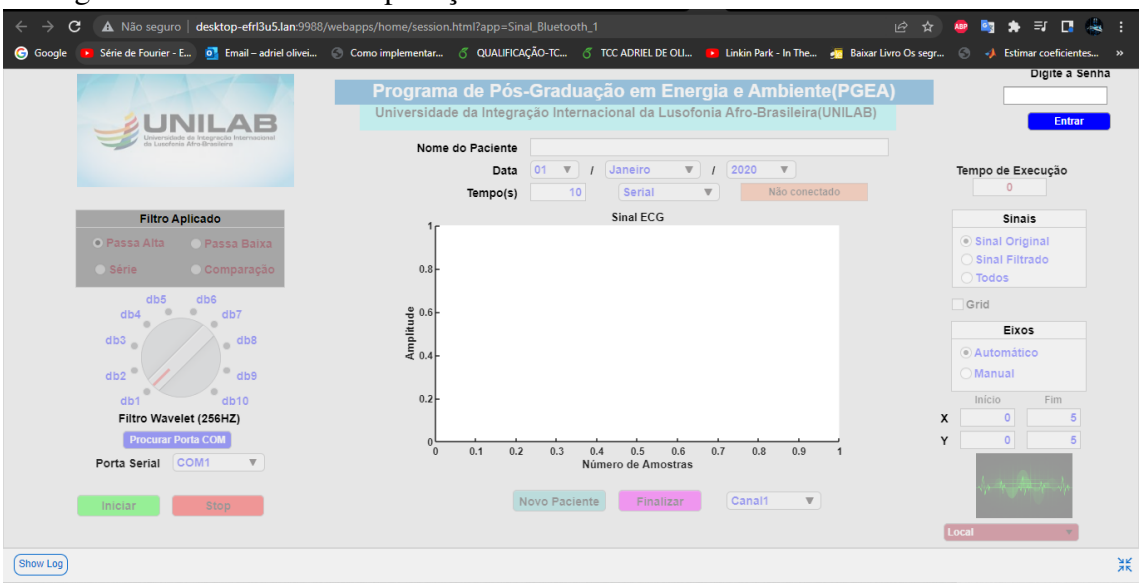
Esta seção mostra a operação de todos os elementos do sistema IoT para aquisição, monitoramento, processamento, armazenamento em nuvem e visualização de sinais de ECG como um todo. Os ensaios experimentais foram realizados com os eletrodos do *hardware* do sistema IoT conectados ao braço direito, braço esquerdo e perna direita do paciente (Figura 32).

Figura 32 – Eletrodos do *hardware* do sistema conectados ao paciente.

Fonte: Próprio autor (2022).

Com o auxílio do *hardware* do sistema IoT proposto, o sinal que caracteriza a atividade elétrica do coração do paciente foi detectado, condicionado e representado em uma forma digital (discreta no tempo). Em seguida, este sinal foi enviado para uma aplicação *WEB* de três formas: Serial, *bluetooth* e *Wi-Fi*. Para limitar o acesso por usuários não autorizados, a aplicação *WEB* só pode ser utilizada após a inserção de uma senha. A Figura 33 mostra uma foto da interface da aplicação *WEB* com as funcionalidades desativadas. Estas funcionalidades são liberadas somente após a inserção da senha de usuário.

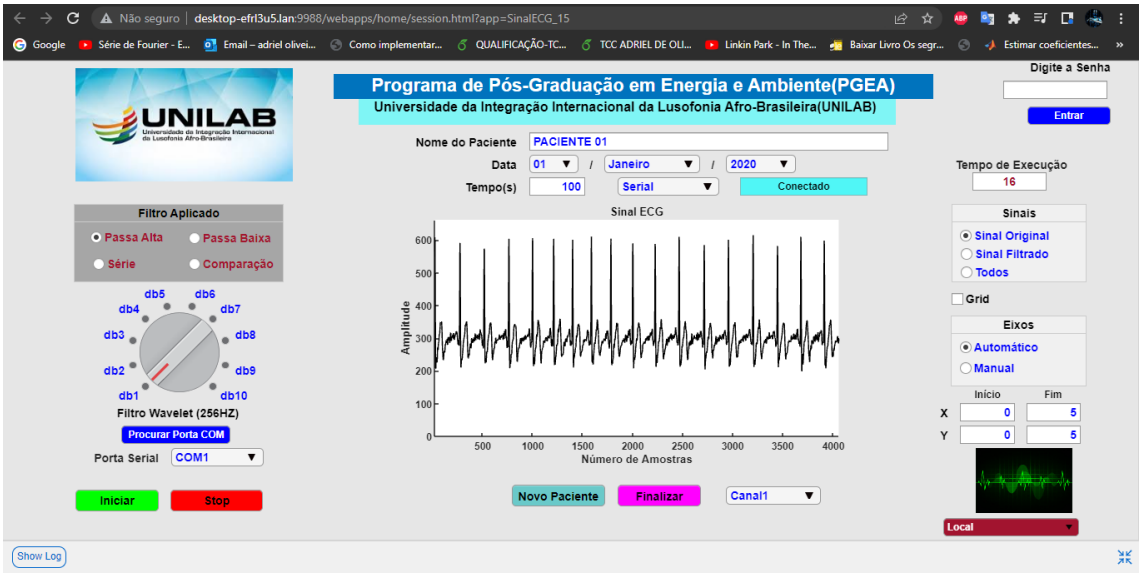
Figura 33 – Interface da aplicação *WEB* com as funcionalidades desativadas.



Fonte: Próprio autor (2022).

Com o auxílio da aplicação *WEB* foi possível inserir os seguintes dados: Nome do paciente, data de aquisição e tempo de aquisição. Para estabelecer uma comunicação serial entre o *hardware* e *software*, é necessário que a aplicação *WEB* esteja hospedada em uma CPU local. Esta CPU deve estar conectada fisicamente com o *hardware* do sistema por meio de um cabo USB. A Figura 34 apresenta uma foto da interface da aplicação *WEB* durante o processo de aquisição de um sinal de ECG via comunicação serial-UART (do inglês: *Universal asynchronous receiver/transmitter*).

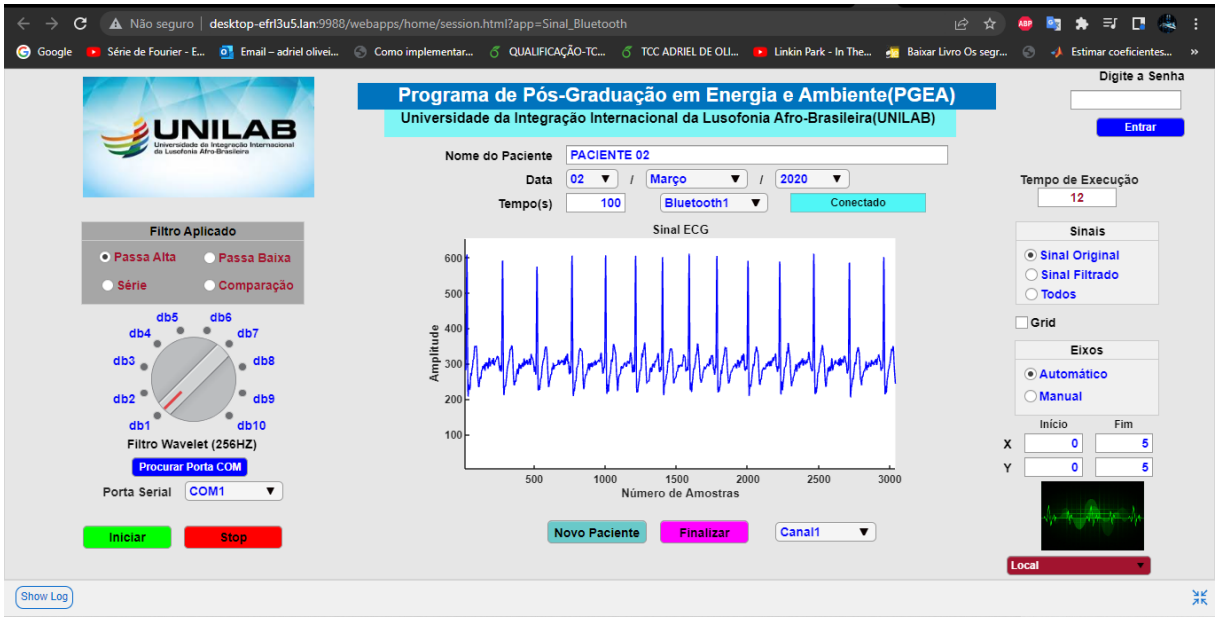
Figura 34 – Interface da aplicação *WEB* durante o procedimento de coleta de um sinal de ECG via comunicação serial.



Fonte: Próprio autor (2022).

Para estabelecer uma comunicação *bluetooth* entre o *hardware* do sistema e a aplicação *WEB* é necessário que a aplicação *WEB* esteja hospedada em uma CPU local com conexão *bluetooth*. Esta CPU deve estar localizada a uma distância máxima de 30 metros do *hardware* do sistema. A Figura 35 apresenta uma foto da interface da aplicação *WEB* durante o processo de aquisição de um sinal de ECG via comunicação *wireless bluetooth*.

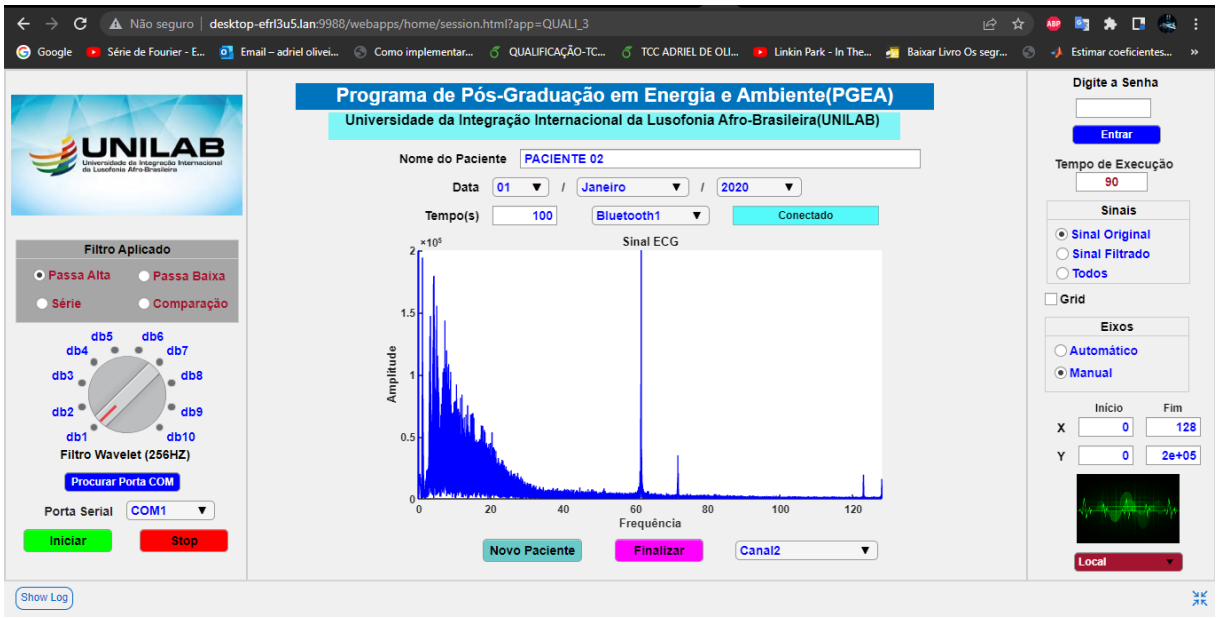
Figura 35 – Interface da aplicação *WEB* durante o procedimento de coleta de um sinal de ECG via comunicação *wireless bluetooth*.



Fonte: Próprio autor (2022).

Ao selecionar a opção "Canal2"na interface da aplicação *WEB*, é ativado o algoritmo que calcula a DFT (do inglês: *Discrete Fourier Transform*) do sinal coletado. A Figura 36 apresenta uma foto da interface *WEB* com a representação no domínio da frequência do sinal coletado após o cálculo e plotagem da DFT em tempo real.

Figura 36 – Cálculo e plotagem da representação no domínio da frequência do sinal coletado.



Fonte: Próprio autor (2022).

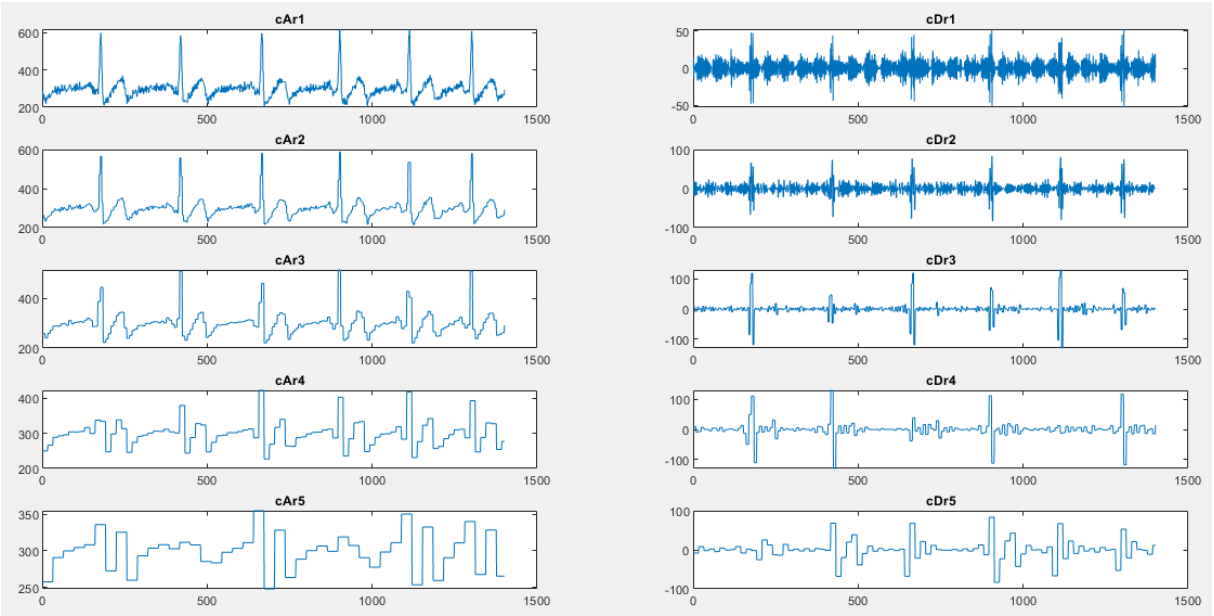
Com o auxílio da segunda interface da aplicação WEB, foi possível aplicar outras técnicas de processamento digital de sinais, tais como: Decomposição wavelet, reconstrução de um sinal contendo apenas componentes de frequência desejadas, aplicar filtros com base na família de wavelets: *Daubechies, Biorthogonal, Coiflets, Symlets, Discrete Meyer, e Reverse Biorthogonal*. A Figura 37 apresenta uma foto da segunda interface da aplicação WEB após a aplicação de uma decomposição wavelet de 5 níveis em um sinal de ECG.

Figura 37 – Interface da aplicação WEB após a aplicação de uma decomposição *wavelet* de cinco níveis em um sinal de ECG.

Fonte: Próprio autor (2022).

No primeiro gráfico da Figura 37, é possível visualizar o sinal formado pelas componentes de frequência 0 a 90 Hz, conforme selecionado pelo usuário. Já no segundo gráfico é possível visualizar a comparação do sinal original com o sinal do gráfico 1. A Figura 38 mostra as componentes de aproximação e de detalhe do sinal de ECG geradas pela aplicação *WEB* após a aplicação de uma decomposição multinível.

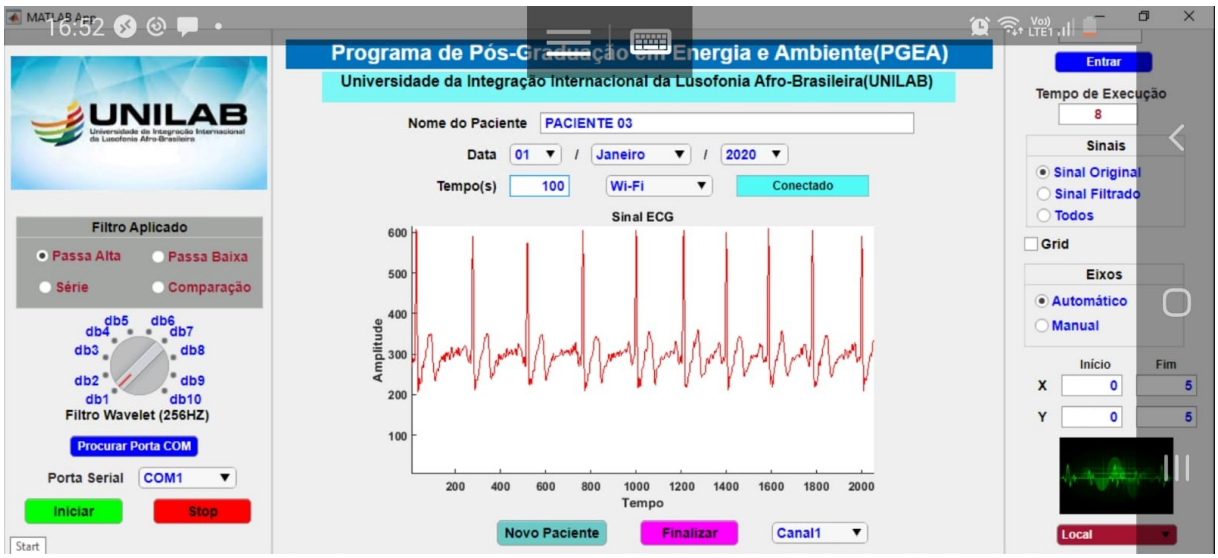
Figura 38 – Componentes de aproximação e de detalhe de um sinal de ECG após a aplicação de uma decomposição *wavelet* em cinco níveis.



Fonte: Próprio autor (2022).

Para estabelecer uma comunicação *Wi-Fi* entre o *hardware* do sistema e a aplicação *WEB*, é opcional a utilização de uma CPU física. Para a exclusão da CPU física, é necessário hospedar a aplicação *WEB* em uma plataforma de computação em nuvem (AWS ou Azure). Nessa comunicação, foi utilizado o *broker Mosquitto.org* para receber e encaminhar as mensagens criptografadas por meio da camada subjacente TLS/SSL entre os clientes *WEB* e o dispositivo. A Figura 39 apresenta uma foto da interface da aplicação *WEB* sendo acessada por um *smarthphone* durante o processo de aquisição de um sinal de ECG via comunicação *wireless Wi-Fi*. Já a Figura 40 mostra uma foto da segunda interface da aplicação *WEB*, sendo acessada a partir de um *smartphone* após uma decomposição *wavelet* multinível.

Figura 39 – Interface da aplicação *WEB* sendo acessada por um *smartphone* durante o procedimento de coleta de um sinal de ECG via comunicação *wireless Wi-Fi*.



Fonte: Próprio autor (2022).

Figura 40 – Interface da aplicação *WEB* sendo acessada a partir de um *smartphone* após a aplicação de uma decomposição *wavelet* de cinco níveis em um sinal de ECG.

Fonte: Próprio autor (2022).

Nas aplicações demonstradas nas Figuras 39 e 40, o servidor foi instalado em uma máquina virtual do serviço de computação em nuvem AZURE. Com o auxílio de um *smarthphone*, a aplicação *WEB* foi acessada por meio de uma conexão RDP (do inglês: *Remote Desktop Protocol*). Outra forma de acessar remotamente uma aplicação *WEB* hospedada em um serviço de computação em nuvem consiste no uso de um endereço URL em um navegador de um *smartphone*, *notebook* ou *tablet*. Este URL é formado pelo IP público da máquina virtual, porta

de acesso externo e link da aplicação *WEB*. O recurso *Auto-Reflow* proporcionado pelo *software* MATLAB faz com que a aplicação *WEB* se adapte ao tamanho de tela de qualquer *smartphone*, *notebook* ou *tablet*.

5.5 Análise de desempenho da aplicação *WEB*

O desempenho de códigos desenvolvidos no MATLAB é medido usando um conjunto de *benchmarks* que abrange operações unitárias e aplicativos completos que representam fluxos de trabalho reais do usuário. Com isto, é possível verificar quais partes do programa levam mais tempo para serem executadas e identificar os gargalos que afetam o desempenho dos códigos implementados. Algumas técnicas foram aplicadas para aumento do desempenho dos códigos implementados tais como: Utilização de sub-rotinas locais ao invés de sub-rotinas aninhadas, pré-alocação de *arrays*, vetorização, uso de funções ao invés de *scripts*, uso de programação modular, uso de *short-circuit operators*, evitar utilização de variáveis globais, criar novas variáveis quando houver mudança no tipo de dados, além de evitar funções (*eval, evalc, evaline, clear all*) do MATLAB que afetam o desempenho dos algoritmos.

Na análise de desempenho dos códigos implementados na aplicação *WEB*, foi utilizado MATLAB *profile*. Com esta ferramenta, foi possível identificar quais funções estão consumindo mais tempo e determinar quais linhas de código não são executadas. Desta forma, melhorias de desempenho foram realizadas nos códigos implementados. A análise de desempenho dos códigos da aplicação *WEB* foi executada em paralelo com a utilização do sistema proposto. Isto foi possível a partir da combinação dos recursos MATLAB *profile* com o *Parallel Computing Toolbox*. Como resultado da análise de desempenho, o MATLAB *profile* gerou um relatório contendo o nome de todas as funções utilizadas nos códigos, número de vezes que cada função foi executada no decorrer da utilização da aplicação *WEB*, bem como o tempo gasto em cada uma delas. Ao clicar em cada função do relatório, é possível visualizar o tempo gasto em cada linha de código e quantas vezes cada linha foi executada.

Para a análise de desempenho da aplicação *WEB*, foram considerados 30 ensaios. Em cada ensaio, a aplicação *WEB* foi submetida à introdução de uma decomposição multinível *wavelet*, cálculo da DFT, plotagem do sinal coletado, plotagem da comparação entre o sinal coletado com um sinal reconstruído a partir das componentes de 0 a 90 Hz do sinal original e a plotagem das componentes de detalhe e de aproximação de uma decomposição de nove níveis em um *buffer* de sete segundos. As ferramentas MATLAB *profile* com o *Parallel Computing*

Toolbox foram executadas em paralelo com a utilização da aplicação *WEB*. A Figura 41 apresenta uma foto do relatório de desempenho da aplicação *WEB* gerado durante um ensaio.

Figura 41 – Relatório de desempenho gerado com o auxílio das ferramentas MATLAB *profile* e *Parallel Computing Toolbox*.

Profile Summary

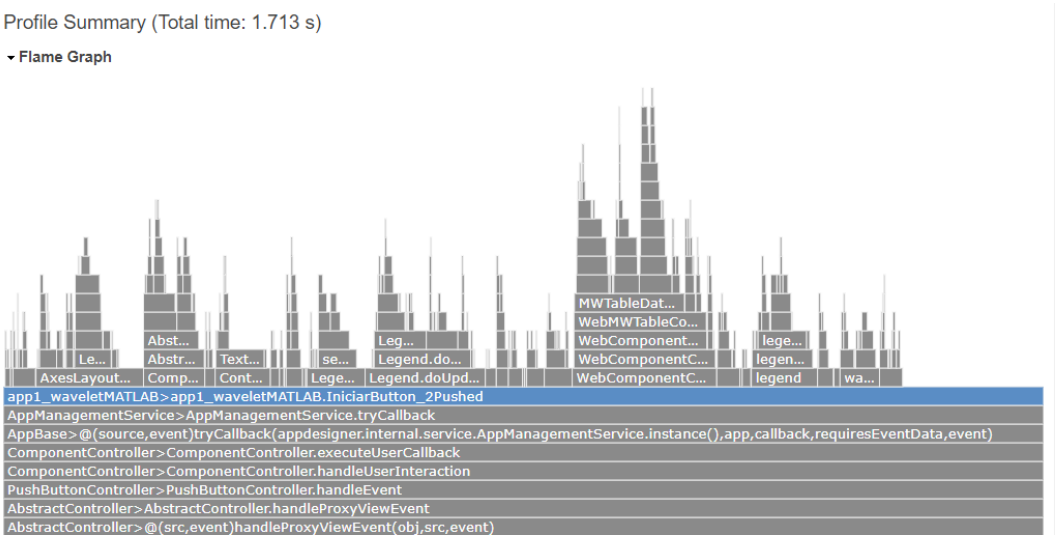
Generated 22-Aug-2022 13:28:58 using performance time.

Function Name	Calls	**Total Time**	Self Time*	Total Time Plot (dark band = self time)
...ucer.convertStructToEventData(event))	1	1.698 s	0.000 s	
...andlePeerEventFromClient(varargin{:})	1	1.698 s	0.000 s	
...deProxyView.handlePeerEventFromClient	1	1.698 s	0.000 s	
...ProxyView>AbstractProxyView.notify	1	1.698 s	0.000 s	
...t)handleProxyViewEvent(obj,src,event)	1	1.698 s	0.000 s	
...stractController.handleProxyViewEvent	1	1.698 s	0.000 s	
...r>PushButtonController.handleEvent	1	1.698 s	0.000 s	
...onentController.handleUserInteraction	1	1.698 s	0.000 s	
...mponentController.executeUserCallback	1	1.698 s	0.000 s	
...app.callback,requiresEventData,event)	1	1.698 s	0.000 s	
...e>AppManagementService.tryCallback	1	1.698 s	0.000 s	
...1_waveletMATLAB.IniciarButton_2Pushed	1	1.698 s	0.240 s	
...gt;WebComponentController.setProperty	1	0.237 s	0.003 s	
....triggerUpdateOnDependentViewProperty	1	0.233 s	0.003 s	
...(varargin)obj.updateData(varargin{:})	1	0.219 s	0.001 s	
...er>WebMWTableController.updateData	1	0.218 s	0.009 s	
Legend.doMethod	14	0.206 s	0.007 s	

Fonte: Próprio autor (2022).

Conforme apresentado na Figura 41, o relatório de desempenho fornece o nome das funções utilizadas, número de chamadas e tempo gasto em cada uma destas funções durante a utilização da aplicação *WEB*. Ao clicar no nome de cada função, é possível verificar o número de chamadas e tempo gasto em cada linha de código. Junto com os relatórios, em cada ensaio foi gerada uma representação gráfica de dados hierárquicos (Figura 42) para visualizar rastreamentos de pilha de *software* com perfil para que os caminhos de código mais frequentes sejam identificados com rapidez e precisão.

Figura 42 – Gráfico de chama gerado a partir dos dados de desempenho da aplicação *WEB*.



Fonte: Próprio autor (2022).

O gráfico apresentado na Figura 42 consiste em uma representação compacta dos dados hierárquicos para verificar o rastreamento de pilhas. Nele, as funções são representadas por *frames* retangulares de mesma altura. O tempo de CPU total usado pela função nomeada no *frame* determina a largura do *frame*. Assim, quanto mais largo for um *frame*, mais frequentemente ele esteve presente nas pilhas. O eixo x mostra a população do perfil da pilha (classificada em ordem alfabética) e o eixo y mostra a profundidade da pilha ao contar a partir de zero na parte inferior.

Após a realização dos 30 ensaios, foi verificado que a aplicação *WEB* apresentou um tempo médio de 1,783 s com um desvio padrão de 0,0737 s. A Tabela 2 mostra um resumo dos resultados obtidos nos ensaios realizados.

Tabela 2 – Resultados obtidos na análise de desempenho da aplicação *WEB*

Ensaios experimentais	Tempo total (s)	Ensaios experimentais	Tempo total (s)
Ensaio 01	1,9190	Ensaio 16	1,7210
Ensaio 02	1,6160	Ensaio 17	1,7650
Ensaio 03	1,7560	Ensaio 18	1,7180
Ensaio 04	1,8760	Ensaio 19	1,7820
Ensaio 05	1,7120	Ensaio 20	1,7060
Ensaio 06	1,7650	Ensaio 21	1,7630
Ensaio 07	1,8330	Ensaio 22	1,8190
Ensaio 08	1,8760	Ensaio 23	1,7770
Ensaio 09	1,7400	Ensaio 24	1,7140
Ensaio 10	1,7200	Ensaio 25	1,7820
Ensaio 11	1,8180	Ensaio 26	1,8730
Ensaio 12	1,8400	Ensaio 27	1,9370
Ensaio 13	1,8820	Ensaio 28	1,8240
Ensaio 14	1,7320	Ensaio 29	1,7240
Ensaio 15	1,7130	Ensaio 30	1,7870
Média (s)		1,7830	
Desvio padrão (s)		0,0737	

Fonte: Próprio autor (2022).

Os resultados apresentados na Tabela 2 foram obtidos com a aplicação *WEB* hospedada em uma instância do MATLAB *Web App Server* executada em uma CPU com as seguintes especificações: Processador Intel Core I7 de 10ª Geração, 8GB RAM, 256GB SSD e sistema operacional *Windows* 10.

5.6 Comparação com trabalhos relacionados

Uma comparação mais detalhada do sistema proposto com artigos selecionados é apresentada nas Tabelas 3 e 4. Com exceção do trabalho proposto por Christ *et al.* (2020), todas as soluções relacionadas não proporcionam um monitoramento contínuo do sinal de ECG sem a dependência de uma CPU. Apenas os trabalhos: Christ *et al.* (2020) e Pereira *et al.* (2022) foram projetados para mobilidade, portanto os demais trabalhos requerem uma fonte de alimentação CC. Alguns trabalhos relacionados não apresentaram parâmetros de coleta de sinal e custo agregado ao protótipo.

Tabela 3 – Comparação com trabalhos relacionados

	(PEREIRA *et al.*, 2022)	(AL-BUSAIDI; KHRIJI, 2013)	(GUPTA *et al.*, 2010)
Interface gráfica	✗	✗	✓
Comunicação serial	✗	✓	✓
Comunicação Wi-Fi	✓	✗	✗
Comunicação Bluetooth	✗	✗	✗
Detecção de ruídos	✗	✗	✗
Algoritmos de PDS	✓	✓	✓
Mobilidade	✓	✗	✗
Armazenamento em nuvem	✗	✗	✗
Custo (USD)	U$ 57,90	NA	U$ 65,00

Fonte: Próprio autor (2022).

Tabela 4 – Comparação com trabalhos relacionados

	(BANSAL *et al.*, 2009)	(CHRIST *et al.*, 2020)	Sistema proposto
Interface gráfica	✓	✓	✓
Comunicaçãoserial	✓	✓	✓
Comunicação Wi-Fi	✗	✓	✓
Comunicação Bluetooth	✗	✗	✓
Detecção de ruídos	✗	✗	✓
Algoritmos de PDS	✓	✓	✓
Mobilidade	✗	✓	✓
Armazenamento em nuvem	✗	✓	✓
Custo (USD)	N/A	NA	U$ 65,96

Fonte: Próprio autor (2022).

Com exceção dos trabalhos Gupta *et al.* (2010) e Bansal *et al.* (2009), todas as soluções relacionadas não são aptas à utilização de diversas técnicas de processamentos simultaneamente devido as limitações de memória. Apenas o trabalho proposto por Pereira *et al.* (2022) envia os registros cardiológicos para uma plataforma de armazenamento em nuvem. Nenhuma

das arquiteturas dos trabalhos relacionados proporciona a funcionalidade de alternar entre coleta no modo *offline* e modo *online*. Apenas no sistema proposto é possível realizar a aplicação de técnicas de processamento digital de sinais em tempo real nos dados cardiográficos coletados, possibilitando uma análise no domínio do tempo e no domínio da frequência com o objetivo de detectar padrões que possibilitem diagnosticar doenças cardiovasculares.

6 CONCLUSÕES E TRABALHOS FUTUROS

Neste trabalho, foi demonstrado com êxito o desenvolvimento e implementação da arquitetura de sistema IoT móvel para aquisição, processamento, armazenamento em nuvem e monitoramento remoto de sinais de ECG para fins de pesquisa em ambientes colaborativos. No processo de aquisição, os sinais que caracterizam a atividade elétrica do coração foram detectados e condicionados por uma *shield* ECG-EMG Olimex. Com o auxílio de um conversor ADC de 10 *bits* de um Arduíno UNO R3, foi obtido uma representação digital do sinal analógico proveniente do biossensor. As etapas de processamento, monitoramento e armazenamento do sinal coletado foram realizadas a partir de uma aplicação *WEB* desenvolvida no ambiente de programação do MATLAB *app designer*. Os teste de operação de todos os elementos do sistema proposto foram realizados em pacientes do Hospital Universitário Walter Cantídio da Universidade Federal do Ceará (UFC). Para utilização do sistema proposto em pacientes, foi concedido pelo Comitê de Ética um parecer consubstanciado com Certificado de Apresentação de Apreciação Ética (CAAE) Nº 47229221.9.0000.5045. Foram coletados registros eletrocardiográficos de 50 pacientes hospitalizados em fase ativa da doença Covid-19. Cada registro coletado possui uma duração de 5 (cinco) minutos e uma frequência de amostragem de 256 Hz. Juntos, estes registros formam um banco de dados para estudos colaborativos com o objetivo de identificar a existência de um padrão de alteração na variabilidade da frequência cardíaca. Os testes de desempenho da aplicação *WEB* desenvolvida foram realizados a partir das ferramentas MATLAB *profile* e *Parallel Computing Toolbox*, por meio das quais foi possível identificar quais funções estão consumindo mais tempo e determinar quais linhas de código não são executadas. Com isto, melhorias de desempenho foram implementadas nos códigos. Os resultados de desempenho da aplicação *WEB* foram satisfatórios, apresentando um tempo médio de 1,7830 s com um desvio padrão de 0,0737 s para calcular a DFT, aplicar uma decomposição *wavelet* multinível em um *Buffer* de 7 segundos, gerar um novo sinal a partir das componentes de frequências desejadas e plotar os resultados. Em termos de consumo de energia, o *hardware* do sistema apresentou um consumo médio de 180 mA durante o funcionamento. Em relação à média de tensão e corrente, o consumo de energia estimado do dispositivo móvel é de 0,90 Watt-hora, com um *power bank* de 10000 mAh conectado ao *hardware* do sistema proposto, é possível obter 55 horas de coleta. A aplicação *WEB* desenvolvida foi hospedada tanto em uma CPU local como em máquinas virtuais dos serviços de computação em nuvem AWS e AZURE. Para acessar a aplicação *WEB*, foram utilizados navegadores de *smartphone*, *tablet* e *notebook*. O monitoramento remoto e armazenamento em

nuvem dos registros cardiográficos foram realizados com êxito. Dentre diferentes arquiteturas de IoT para coletar registros cardiográficos, o sistema proposto se destacou por proporcionar diversas funcionalidades tais como comunicação *bluetooth*, serial, *Wi-Fi*, processamento em tempo real dos registro coletados e comunicação direta com serviços de armazenamento em nuvem. Além disso, o hardware do sistema proposto foi prototipado utilizando componentes de baixo custo e baixo consumo de energia, tornando-o uma tecnologia móvel acessível capaz de alimentar automaticamente base de dados colaborativas.

6.1 Trabalhos Futuros

Pretende-se para atividades futuras:

- **Implementar algoritmos de segmentação:** Acrescentar na aplicação *WEB* a funcionalidade de segmentação completa do sinal ECG com a derivação das principais séries temporais resultantes como série de intervalos R-R (variabilidade da frequência cardíaca), série de durações do complexo QRS, séries de durações das ondas P e T, séries PR e QT;
- **Implementar algoritmos de *Machine Learning*:** Adicionar na aplicação *WEB* algoritmos capazes de classificar os principais tipos de arritmias cardíacas.
- **Migrar para uma plataforma *Open Source*:** Replicar a programação desenvolvida no MATLAB *App Designer* em uma plataforma *Open Source* com o objetivo de tornar o sistema aberto para pesquisas colaborativas. Com isto, novos pesquisadores podem utilizar o sistema proposto para aplicar seus algoritmos de detecção de doenças cardíacas.

REFERÊNCIAS

AHAMED, M. A.; HASAN, M. K.; ALAM, M. S. Design and implementation of low cost ecg monitoring system for the patient using smartphone. **2015 International Conference on Electrical & Electronic Engineering (ICEEE)**, p. 261–264, 2015.

AL-BUSAIDI, A. M.; KHRIJI, L. Digitally filtered ecg signal using low-cost microcontroller. In: **2013 International Conference on Control, Decision and Information Technologies (CoDIT)**. [S.l.: s.n.], 2013. p. 258–263.

ALGHATRIF, M.; LINDSAY, J. A brief review: history to understand fundamentals of electrocardiography. **Journal of Community Hospital Internal Medicine Perspectives**, Taylor Francis, v. 2, n. 1, p. 14383, 2012. PMID: 23882360. Disponível em: <https://doi.org/10.3402/jchimp.v2i1.14383>.

AMAZON. **Shield Ekg Emg Kit Including Leads Shield Ekg Emg Pa.** 2015. Disponível em: <https://www.amazon.in/Shield-Ekg-Emg-Including-Leads/dp/B00R3QXDX0>. Acesso em: 08 de Abril de 2020.

ARDUINO.CC. **Arduino Uno Rev3**. 2021. Disponível em: <https://store.arduino.cc/products/arduino-uno-rev3>. Acesso em: 25 de junho de 2021.

ARORA, N.; MISHRA, B. Origins of ecg and evolution of automated dsp techniques: A review. **IEEE Access**, v. 9, p. 140853–140880, 2021.

ATMEL. **ATmega328P Datasheet.** 2015. Disponível em: <http://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-7810-Automotive-Microcontrollers-ATmega328P_Datasheet.pdf>. Acesso em: 13 de fevereiro de 2022.

BALAKRISHNA, K.; RAJESH, N. Design of remote monitored solar powered grasscutter robot with obstacle avoidance using iot. **Global Transitions Proceedings**, v. 3, n. 1, p. 109–113, 2022. ISSN 2666-285X. International Conference on Intelligent Engineering Approach(ICIEA-2022). Disponível em: <https://www.sciencedirect.com/science/article/pii/S2666285X22000590>.

BANERJEE, S.; GUPTA, R.; MITRA, M. Delineation of ecg characteristic features using multiresolution wavelet analysis method. **Measurement**, v. 45, n. 3, p. 474–487, 2012. ISSN 0263-2241. Disponível em: <https://www.sciencedirect.com/science/article/pii/S026322411100371X>.

BANSAL, D.; KHAN, M.; SALHAN, A. K. A computer based wireless system for online acquisition, monitoring and digital processing of ecg waveforms. **Computers in Biology and Medicine**, v. 39, n. 4, p. 361–367, 2009. ISSN 0010-4825. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0010482509000195>.

BOUAZIZ, F.; BOUTANA, D.; BENIDIR, M. Multiresolution wavelet-based qrs complex detection algorithm suited to several abnormal morphologies. **IET Signal Processing**, v. 8, n. 7, p. 774–782, 2014. Disponível em: <https://ietresearch.onlinelibrary.wiley.com/doi/abs/10.1049/iet-spr.2013.0391>.

CAJAVILCA, C.; VARON, J. Willem einthoven: The development of the human electrocardiogram. **Resuscitation**, v. 76, n. 3, p. 325–328, 2008. ISSN 0300-9572. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0300957207005813>.

CHEE, J.; SEOW, S.-C. The electrocardiogram. In: ____. **Advances in Cardiac Signal Processing**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007. p. 1–53. ISBN 978-3-540-36675-1. Disponível em: <https://doi.org/10.1007/978-3-540-36675-1_1>.

CHRIST, M.; NARAYANAN, A.; SELVARAJ, J. Low cost internet of things based remote monitoring system and ecg analysis. In: . [S.l.: s.n.], 2020. v. 17, p. 1863–1866.

COSOLI, G.; SPINSANTE, S.; SCARDULLA, F.; D'ACQUISTO, L.; SCALISE, L. Wireless ecg and cardiac monitoring systems: State of the art, available commercial devices and useful electronic components. **Measurement**, v. 177, p. 109243, 2021. ISSN 0263-2241. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0263224121002542>.

CRAWFORD, M. H.; BERNSTEIN, S. J.; DEEDWANIA, P. C.; DIMARCO, J. P.; FERRICK, K. J.; GARSON, A.; GREEN, L. A.; GREENE, H.; SILKA, M. J.; STONE, P. H.; TRACY, C. M.; GIBBONS, R. J.; ALPERT, J. S.; EAGLE, K. A.; GARDNER, T. J.; GARSON, A.; GREGORATOS, G.; RUSSELL, R. O.; RYAN, T. J.; SMITH, S. C. Acc/aha guidelines for ambulatory electrocardiography. **Journal of the American College of Cardiology**, v. 34, n. 3, p. 912–948, 1999. ISSN 0735-1097. Disponível em: <https://www.sciencedirect.com/science/article/pii/S073510979900354X>.

ESPRESSIF-SYSTEMS. **ESP8266EX Datasheet.** 2022. Disponível em: <https://www.espressif.com/en/support/documents/technical-documents>. Acesso em: 13 de fevereiro de 2022.

FATEC. **NodeMCU.** 2022. Disponível em: <http://www.fatecjd.edu.br/estacao/nodemcu.html>. Acesso em: 12 de Janeiro de 2022.

FIEV, D.; VINAROV, A.; TSARICHENKO, D.; KOPYLOV, P.; DEMIDKO, Y.; SYRKIN, A.; RAPOPORT, L.; ALYAEV, Y.; GLYBOCHKO, P. Holter monitoring (24-hour ecg) parameter dynamics in patients with ischemic heart disease and lower urinary tract symptoms due to benign prostatic hyperplasia. **Advances in Therapy**, v. 36, 05 2019.

GOLDBERGER, A. L. Chapter 1 - introductory principles. In: GOLDBERGER, A. L. (Ed.). **Clinical Electrocardiography: A Simplified Approach (Seventh Edition)**. Seventh edition. Philadelphia: Mosby, 2006. p. 3–6. ISBN 978-0-323-04038-9. Disponível em: <https://www.sciencedirect.com/science/article/pii/B0323040381500020>.

GUPTA, R.; BERA, J.; MITRA, M. Development of an embedded system and matlab-based gui for online acquisition and analysis of ecg signal. **Measurement**, v. 43, n. 9, p. 1119–1126, 2010. ISSN 0263-2241. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0263224110001144>.

HARSKAMP, R. E. Electrocardiographic screening in primary care for cardiovascular disease risk and atrial fibrillation. **Prim. Health Care Res. Dev.**, Cambridge University Press (CUP), v. 20, n. e101, p. e101, jun. 2019.

HIVEMQ. **MQTT Security Fundamentals**. 2015. Disponível em: <https://www.hivemq.com/blog/introducing-the-mqtt-security-fundamentals>. Acesso em: 15 de Agosto de 2022.

JANVEJA, M.; TRIVEDI, G. An area and power efficient vlsi architecture for ecg feature extraction for wearable iot healthcare applications. **Integration**, v. 82, p. 96–103, 2022. ISSN 0167-9260. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0167926021001115>.

JOOM. **Uno R3 Development Board ATmega328P CH340 CH340G para Arduino UNO R3.** 2022. Disponível em: <https://www.joom.com/pt-br/products/5dd7a50a36b54d01015bd0ed>. Acesso em: 04 de março de 2022.

KADISH, A. H.; BUXTON, A. E.; KENNEDY, H. L.; KNIGHT, B. P.; MASON, J. W.; SCHUGER, C. D.; TRACY, C. M.; WINTERS, W. L.; BOONE, A. W.; ELNICKI, M.; HIRSHFELD, J. W.; LORELL, B. H.; RODGERS, G. P.; TRACY, C. M.; WEITZ, H. H. Acc/aha clinical competence statement on electrocardiography and ambulatory electrocardiography. **Journal of the American College of Cardiology**, v. 38, n. 7, p. 2091–2100, 2001. ISSN 0735-1097. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0735109701016801>.

KENNEDY, H. L. The evolution of ambulatory ecg monitoring. **Progress in Cardiovascular Diseases**, v. 56, n. 2, p. 127–132, 2013. ISSN 0033-0620. Ambulatory ECG Monitoring: Clinical Practice and Research Applications. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0033062013001503>.

Li, J.; Deng, G.; Wei, W.; Wang, H.; Ming, Z. Design of a real-time ecg filter for portable mobile medical systems. In: IEEE ACCESS. [S.l.], 2017. v. 5, p. 696–704.

MAHDY, L. N.; EZZAT, K. A.; TAN, Q. Smart ecg holter monitoring system using smartphone. In: **2018 IEEE International Conference on Internet of Things and Intelligence System (IOTAIS)**. [S.l.: s.n.], 2018. p. 80–84.

MATHWORKS. **MATLAB App Designer**. 2020. Disponível em: <https://www.mathworks.com/products/matlab/app-designer.html>. Acesso em: 22 Janeiro 2020.

MATHWORKS. **MATLAB Web App Server**. 2020. Disponível em: <https://www.mathworks.com/products/matlab-web-app-server.html>. Acesso em: 07 Julho 2020.

MIRVIS, D. M.; GOLDBERGER, A. L. Electrocardiography. **Heart disease**, WB Saunders, Philadelphia, v. 1, p. 82–128, 2001.

MORAES, D. F. C. de. **Sistema de automação predial utilizando protocolo MQTT e plataforma web**. 74 f. Monografia (Graduação) — Instituto Federal do Ceará, Fortaleza, 2020.

MóDULO-ELETRôNICA. **Umidade-temperatura-presença no smartphone com módulo bluetooth HC05.** 2018. Disponível em: <https://blog.moduloeletronica.com.br/umidade-temperatura-presenca-no-smartphone-com-modulo-bluetooth-hc05.> Acesso em: 25 de junho de 2021.

OGANISYAN, B. A.; OGANESYAN, T. N.; MAKARYAN, A. O. Time-frequency analysis of electric cardiograms. **Journal of Contemporary Physics (Armenian Academy of Sciences)**, v. 55, n. 4, p. 371–375, 2020. ISSN 1934-9378. Disponível em: <https://doi.org/10.3103/S1068337220040155>.

OLIMEX. **SHIELD-EKG-EMG user's manual**. 2014. Disponível em: <https://www.olimex.com/Products/Duino/Shields/SHIELD-EKG-EMG/resources/SHIELD-EKG-EMG.pdf>. Acesso em: 15 Julho 2020.

OLIMEX. **SHIELD-EKG-EMG user's manual**. 2020. Disponível em: <https://www.olimex.com/Products/Duino/Shields/SHIELD-EKG-EMG/\open-source-hardware>. Acesso em: 25 Julho 2020.

OMS. **The top 10 causes of death**. 2020. Disponível em: <https://www.who.int/news-room/fact-sheets/detail/the-top-10-causes-of-death>. Acesso em: 15 Dezembro de 2020.

PEREIRA, M. B.; MADEIRO, J. P. V.; FREITAS, A. O.; GOMES, D. G. An iot-cloud enabled real time and energy efficient ecg reader architecture. In: BASTOS-FILHO, T. F.; CALDEIRA, E. M. de O.; FRIZERA-NETO, A. (Ed.). **XXVII Brazilian Congress on Biomedical Engineering**. Cham: Springer International Publishing, 2022. p. 989–995. ISBN 978-3-030-70601-2.

PINTO, J. R.; CARDOSO, J. S.; LOURENçO, A. Evolution, current challenges, and future possibilities in ecg biometrics. **IEEE Access**, v. 6, p. 34746–34776, 2018.

POLIKAR, R. **Multiresolution Analysis: The Discrete Wavelet Transform**. 2004. Disponível em: <http://www.cs.ucf.edu/courses/cap5015/WTpart4.pdf>. Acesso em: 11 Maio 2022.

RIERA, A. R. P.; UCHIDA, A. H.; FILHO, C. F.; MENEGHINI, A.; FERREIRA, C.; SCHAPACKNIK, E.; DUBNER, S.; MOFFA, P. Significance of vectorcardiogram in the cardiological diagnosis of the 21st century. **Clinical Cardiology: An International Indexed and Peer-Reviewed Journal for Advances in the Treatment of Cardiovascular Disease**, Wiley Online Library, v. 30, n. 7, p. 319–323, 2007.

SAIRAM, K.; GUNASEKARAN, N.; REDD, S. Bluetooth in wireless communication. **IEEE Communications Magazine**, v. 40, n. 6, p. 90–96, 2002.

SATIJA, U.; RAMKUMAR, B.; MANIKANDAN, M. S. An automated ecg signal quality assessment method for unsupervised diagnostic systems. In: BIOCYBERNETICS AND BIOMEDICAL ENGINEERING. [S.l.], 2018. v. 38, p. 54–70.

SCANLON, V. C.; SANDERS, T. **Essentials of anatomy and physiology**. [S.l.]: FA Davis, 2018.

SERHANI, M. A.; KASSABI, H. T. E.; ISMAIL, H.; NAVAZ, A. N. Ecg monitoring systems: Review, architecture, processes, and key challenges. **Sensors**, v. 20, n. 6, 2020. ISSN 1424-8220. Disponível em: <https://www.mdpi.com/1424-8220/20/6/1796>.

TOMPKINS, W. J. Biomedical digital signal processing. 2000.

TORTORA, G. J.; DERRICKSON, B. H. Introduction to the human body. In: **Introduction to the Human Body: The Essentials of Anatomy and Physiology**. [S.l.]: Wiley, 2014.

WASIMUDDIN, M.; ELLEITHY, K.; ABUZNEID, A.-S.; FAEZIPOUR, M.; ABUZAGHLEH, O. Stages-based ecg signal analysis from traditional signal processing to machine learning approaches: A survey. **IEEE Access**, v. 8, p. 177782–177803, 2020.

ZACARIAS, H. M. J.; MARQUES, J. A. L.; CORTEZ, P. C.; MADEIRO, J. P. V.; CAVALCANTE, C. C. Detrended fluctuation analysis como ferramenta para avaliação do comportamento da frequência cardíaca fetal em exames cardiotocográficos. In: XXIV CONGRESSO BRASILEIRO DE ENGENHARIA BIOMéDICA. Uberlândia, 2014. p. 2912–2915.

ZHONG, L.; TAN, R. S.; GHISTA, D. N. Anatomy and physiology of the heart. In: **Computational and Mathematical Methods in Cardiovascular Physiology**. [S.l.]: World Scientific, 2019. p. 3–37.

ANEXO A – CÓDIGOS UTILIZADOS NO MICROCONTROLADOR ARDUÍNO UNO R3

```
/*******************************************************/
#include<compat/deprecated.h>
#include <FlexiTimer2.h>
//http://www.arduino.cc/playground/Main/FlexiTimer2

// All definitions
#define NUMCHANNELS 6
#define HEADERLEN 4
#define PACKETLEN (NUMCHANNELS * 2 + HEADERLEN + 1)
#define SAMPFREQ 256
#define TIMER2VAL (1024/(SAMPFREQ))
#define LED1  13
#define CAL_SIG 9

// Global constants and variables
volatile unsigned char TXBuf[PACKETLEN];
volatile unsigned char TXIndex;
volatile unsigned char CurrentCh;
volatile unsigned char counter = 0;
volatile unsigned int ADC_Value = 0;

//~~~~~~~~~~
// Funções
//~~~~~~~~~~


void Toggle_LED1(void){

 if((digitalRead(LED1))==HIGH){ digitalWrite(LED1,LOW); }
 else{ digitalWrite(LED1,HIGH); }

}

void toggle_GAL_SIG(void){

 if(digitalRead(CAL_SIG) == HIGH){ digitalWrite(CAL_SIG, LOW); }
 else{ digitalWrite(CAL_SIG, HIGH); }

}

void setup() {

 noInterrupts();  // Disable all interrupts before initialization

 // LED1
 pinMode(LED1, OUTPUT);  //Setup LED1 direction
 digitalWrite(LED1,LOW); //Setup LED1 state
 pinMode(CAL_SIG, OUTPUT);
```

```
 //Write packet header and footer
 TXBuf[0] = 0xa5;
 TXBuf[1] = 0x5a;
 TXBuf[2] = 2;
 TXBuf[3] = 0;
 TXBuf[4] = 0x02;
 TXBuf[5] = 0x00;
 TXBuf[6] = 0x02;
 TXBuf[7] = 0x00;
 TXBuf[8] = 0x02;
 TXBuf[9] = 0x00;
 TXBuf[10] = 0x02;
 TXBuf[11] = 0x00;
 TXBuf[12] = 0x02;
 TXBuf[13] = 0x00;
 TXBuf[14] = 0x02;
 TXBuf[15] = 0x00;
 TXBuf[2 * NUMCHANNELS + HEADERLEN] =  0x01;

FlexiTimer2::set(TIMER2VAL,Timer2_Overflow_ISR);
FlexiTimer2::start();

Serial.begin(57600);

interrupts();
}


voidTimer2_Overflow_ISR()
{

  Toggle_LED1();


 for(CurrentCh=0;CurrentCh<6;CurrentCh++){
    ADC_Value = analogRead(CurrentCh);
    TXBuf[((2*CurrentCh) + HEADERLEN)] = ((unsigned char)((ADC_Value & 0xFF00) >>
8));
    TXBuf[((2*CurrentCh) + HEADERLEN + 1)] = ((unsigned char)(ADC_Value &
0x00FF));
  }


 for(TXIndex=0;TXIndex<17;TXIndex++){
    Serial.write(TXBuf[TXIndex]);
  }

  TXBuf[3]++;
```

```
  counter++;
  if(counter == 12){
    counter = 0;
    toggle_GAL_SIG();
}


void loop() {

 __asm__ __volatile__ ("sleep");

}
```

ANEXO B – CÓDIGOS UTILIZADOS NO MÓDULO WI-FI ESP8266

```
#include <ESP8266WiFi.h>    // Biblioteca que gerencia o wifi
#include <PubSubClient.h>   // Biblioteca que gerencia o cliente mqtt
#include <SoftwareSerial.h> // Biblioteca que cria a porta serial nos pinos
digitais para se comunicar com o Arduino

//Parâmetros de configuração da tua aplicação
#define BUFFER_SIZE  (50)   // Tamanho do teu buffer de dados recebido pelo
Arduino
byte ret = 0;
 long int timeelapsed ;
 long int currentime;
//int dados[BUFFER_SIZE];    // variável que obtém o buffer do Arduino
char dados[BUFFER_SIZE];
int idx = 0;                // incrementa o buffer de dados
String GERAL="";
// Configuração do Wifi e MQTT
const char* ssid = "Adriel";
const char* password = "asdfghjk";
const char* mqtt_server = "test.mosquitto.org";
// Topico que o esp8266 vai publicar os dados do arduino. O matlab deverá estar
// inscrito neste tópico para receber as mensagens do esp8266.
const char* topicPublish = "teste";

// Inicializa cliente wifi
WiFiClient espClient;
// Inicializa cliente mqtt
PubSubClient client(espClient);
//Inicializa Serial que se conecta com o Arduino.
// Rx: GPIO 14 (D5)
// Tx: GPIO 12 (D6)
SoftwareSerial serialArduino(14, 12);


/*
 *  Inicializa conexão do esp8266 com a rede wifi
 */
void setup_wifi() {

  delay(10);

  Serial.println();
  Serial.print("Connecting to ");
  Serial.println(ssid);

  WiFi.mode(WIFI_STA);
  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
```

```
    Serial.print(".");
  }

 randomSeed(micros());

 Serial.println("");
 Serial.println("WiFi connected");
 Serial.println("IP address: ");
 Serial.println(WiFi.localIP());
}

/*
 * Função de retorno de chamada do mqtt. Quando chega uma mensagem
 * de um tópico que você se inscreveu o mqtt executa essa função
 * onde será processada a mensagem recebida.
 * Detalhe: esta função somente será chamada se você estiver inscrito
 * em algum tópico, caso contrário ela nunca será chamada. Para a sua
 * aplicação não será necessário, uma vez que você só precisa publicar
 * os dados para o mqtt.
 */
void callback(char* topic, byte* payload, unsigned int length) {
 Serial.print("Mensagem recebida [");
 Serial.print(topic);
 Serial.print("] ");
 // Imprime mensagem recebida
 for (int i = 0; i < length; i++) {
  Serial.print((char)payload[i]);
 }
 Serial.println();
}

/*
 * Essa função reconecta com o broker, caso tenha se desconectado
 * deste ou do wifi.
 */
void reconnect() {
 // Verifica se o cliente foi desconectado e entra no loop até que ele se
conecte.
 while (!client.connected()) {
  Serial.print("Attempting MQTT connection...");
  // Inicia conexão com o broker mqtt e verifica se deu certo.
  // cliente_esp8266 é o ID do cliente mqtt.
  if (client.connect("cliente_esp8266")) {
    Serial.println("Conectado ao Broker MQTT!");
    // Aqui você já pode se inscrever em algum tópico que eu deseje receber
mensagem, ex:
    // client.subscribe("nome do topico");
  } else {
    Serial.print("Conexão com Broker MQTT falhou, rc=");
```

```
      Serial.print(client.state());
      Serial.println("Tentando se reconectar em 5s");
      delay(5000);
     }
   }
}

void setup() {
serialArduino.setTimeout(50);
  pinMode(BUILTIN_LED, OUTPUT);
  // Inicializa Serial padrão do Esp8266 para logs
  Serial.begin(57600);
  // Inicializa Serial para comunicação com Arduino
  serialArduino.begin(57600);
  // Conecta-se na rede Wifi
  setup_wifi();
  // Define os parâmetros de configuração MQTT
  client.setServer(mqtt_server, 1883);
  client.setCallback(callback);
}

void loop() {

  // Verifica constantemente se o cliente mqtt foi
  // desconectado e reconecta caso necessário.
  if (!client.connected()) {
    reconnect();
  }

  // Mantém a conexão mqtt aberta
  client.loop();
 // Recebe os dados do Arduino e publica no mqtt
 currentime = millis();
 while(serialArduino.available() > 0){
   GERAL=serialArduino.readString();
    Serial.println(GERAL);
//      if( idx < BUFFER_SIZE ){
//        // Ler um byte recebido e incrementa o buffer até completá-lo
//        dados[idx++] = serialArduino.read();
//        Serial.println(serialArduino.read());}
//      }else{
//        // Significa que o buffer está cheio, agora é só publicar no mqtt.
//        // Imprime na Serial padrão do esp8266, você pode comentar isso.
//        Serial.print("dados: ");
//        Serial.println(dados);
//        // Publica no mqtt.
         client.publish(topicPublish, (char*) GERAL.c_str());
//        client.publish(topicPublish, GERAL);
//        client.publish(topicPublish, (char*) payload.c_str())
```

ANEXO C – CÓDIGOS UTILIZADOS NA SEGUNDA INTERFACE DA APLICAÇÃO WEB

```
classdef app1_waveletMATLAB < matlab.apps.AppBase

    % Properties that correspond to app components
    properties (Access = public)
        SinalECGUIFigure               matlab.ui.Figure
        TabGroup                       matlab.ui.container.TabGroup
        AnliseWaveletTab               matlab.ui.container.Tab
        DetecodeRudosButtonGroup       matlab.ui.container.ButtonGroup
        IniciarButton_3                matlab.ui.control.Button
        IntervaloButtonGroup           matlab.ui.container.ButtonGroup
        XEditField_10                  matlab.ui.control.NumericEditField
        XEditField_12                  matlab.ui.control.NumericEditField
        MAPLIAWGNButton                matlab.ui.control.RadioButton
        ABButton                       matlab.ui.control.RadioButton
        BWButton                       matlab.ui.control.RadioButton
        FLouTVNButton                  matlab.ui.control.RadioButton
        Grfico1CheckBox                matlab.ui.control.CheckBox
        TamanhoEditField               matlab.ui.control.NumericEditField
        TamanhoEditFieldLabel          matlab.ui.control.Label
        SinalOriginalCheckBox          matlab.ui.control.CheckBox
        NomedaVarivelEditField         matlab.ui.control.EditField
        NomedaVarivelEditFieldLabel    matlab.ui.control.Label
        StopButton_2                   matlab.ui.control.StateButton
        IniciarButton_2                matlab.ui.control.Button
        SelecioneoarquivomatcomosinalLabel  matlab.ui.control.Label
        DiretriodoarquivoEditField     matlab.ui.control.EditField
        BrowseButton                   matlab.ui.control.Button
        Panel_3                        matlab.ui.container.Panel
        text19_2                       matlab.ui.control.Label
        text18_2                       matlab.ui.control.Label
        Panel_2                        matlab.ui.container.Panel
        ConfiguraesGrfico2Panel        matlab.ui.container.Panel
        ButtonGroup                    matlab.ui.container.ButtonGroup
        SubtraoButton                  matlab.ui.control.RadioButton
        SomaButton                     matlab.ui.control.RadioButton
        ContedoaserExibidoPanel        matlab.ui.container.Panel
        UITable                        matlab.ui.control.Table
        VisualizaodosEixosButtonGroup_2  matlab.ui.container.ButtonGroup
        YEditField_3                   matlab.ui.control.NumericEditField
        YEditField_3Label              matlab.ui.control.Label
        XEditField_9                   matlab.ui.control.NumericEditField
        XEditField_9Label              matlab.ui.control.Label
        XEditField_8                   matlab.ui.control.NumericEditField
        XEditField_7                   matlab.ui.control.NumericEditField
        ManualButton_4                 matlab.ui.control.RadioButton
        AutomticoButton_4              matlab.ui.control.RadioButton
        ConfiguraesGrfico1Panel        matlab.ui.container.Panel
        VisualizaodosEixosButtonGroup  matlab.ui.container.ButtonGroup
        YEditField_2                   matlab.ui.control.NumericEditField
        YEditField_2Label              matlab.ui.control.Label
        XEditField_6                   matlab.ui.control.NumericEditField
        XEditField_6Label              matlab.ui.control.Label
        XEditField_5                   matlab.ui.control.NumericEditField
        XEditField_4                   matlab.ui.control.NumericEditField
        ManualButton_3                 matlab.ui.control.RadioButton
        AutomticoButton_3              matlab.ui.control.RadioButton
        ContedoaserExibidoButtonGroup  matlab.ui.container.ButtonGroup
        opesDropDown                   matlab.ui.control.DropDown
        opesDropDownLabel              matlab.ui.control.Label
        DetalheButton                  matlab.ui.control.RadioButton
        AproximaoButton                matlab.ui.control.RadioButton
        Panel                          matlab.ui.container.Panel
        SinalInseridoButtonGroup       matlab.ui.container.ButtonGroup
        AutomticoButton_2              matlab.ui.control.RadioButton
        ManualButton_2                 matlab.ui.control.RadioButton
        FamliasWaveletButtonGroup      matlab.ui.container.ButtonGroup
        AtualizarButton                matlab.ui.control.StateButton
        ReverseBiorthogonalButton      matlab.ui.control.RadioButton
        BiorthogonalButton             matlab.ui.control.RadioButton
        DiscreteMeyerButton            matlab.ui.control.RadioButton
        FejerKorovkinfiltersButton     matlab.ui.control.RadioButton
        SymletsButton                  matlab.ui.control.RadioButton
        CoifletsButton                 matlab.ui.control.RadioButton
        DaubechiesButton               matlab.ui.control.RadioButton
        GerarGrficosPanel              matlab.ui.container.Panel
        cArecDrCheckBox                matlab.ui.control.CheckBox
        cAecDCheckBox                  matlab.ui.control.CheckBox
        Panel_4                        matlab.ui.container.Panel
        NveldeDecompDropDown           matlab.ui.control.DropDown
        NveldeDecompDropDownLabel      matlab.ui.control.Label
        FiltrosKnob                    matlab.ui.control.DiscreteKnob
        FiltrosKnobLabel               matlab.ui.control.Label
        UIAxes2_2                      matlab.ui.control.UIAxes
        UIAxes2                        matlab.ui.control.UIAxes
        Tab                            matlab.ui.container.Tab
    end


    properties (Access = private)
        aux1=0;
    end


    % Callbacks that handle component events
    methods (Access = private)

        % Callback function
        function IniciarButtonPushed(app, event)

        FILTRO=app.FiltroWavelet256HZKnob.Value;
            PORTA=app.PortaSerialDropDown.Value;
            NOME=app.NomedoPacienteEditField.Value;
```

```
            DIA=app.DataDropDown.Value;
            MES=app.DropDown.Value;
            ANO=app.DropDown_2.Value;
            TEMPO=app.TemposEditField.Value;
            DATA= strcat(DIA,'/',MES,'/',ANO);
            global counter1;
            global counter;
             counter1=0;
             counter=0;
             A3=0;
            switch FILTRO
                case 'OFF'
                    Inicio_Espera_Buffer = [];
Fim_Espera_Buffer = [];
Inicio_ReadSave_Buffer = [];
Fim_ReadSave_Buffer = [];
Inicio_Filtra_Buffer = [];
Fim_Filtra_Buffer = [];

tic;
tstart = tic;
Inicio_Espera_Buffer = [Inicio_Espera_Buffer tstart];
serial_list = instrfind;
delete(serial_list);
s = serial(PORTA);   %% PORTA COM A QUAL O ARDUINO SE COMUNICA COM O COMPUTADOR  %%
s.InputBufferSize = 2*16384;
s.BaudRate = 57600;   %% TAXA DE TRANSMISSÃO ARDUINO - PORTA USB - COMPUTADOR
fopen(s);
parou = 0;
buffer = 0;
time = 0;
sfreq = 256;

while s.BytesAvailable < s.InputBufferSize
    time = time+1;
end
tending = toc(tstart)
Fim_Espera_Buffer = [Fim_Espera_Buffer tending];
tic;
tstart2 = tic;
Inicio_ReadSave_Buffer = [Inicio_ReadSave_Buffer tstart2];
out = fread(s,s.BytesAvailable,'char');

find_sync = 0;
jj=1;
while find_sync == 0;
    sync0 = dec2hex(out(jj));
    sync1 = dec2hex(out(jj+1));
    if sum(sync0 == 'A5') == 2 && sum(sync1 == '5A') == 2
        find_sync = 1;
    else jj = jj+1;
    end
end

% index_hb_ch1 = jj+14:17:length(out);
% index_lb_ch1 = jj+15:17:length(out);

index_hb_ch2 = jj+6:17:length(out);
index_lb_ch2 = jj+7:17:length(out);

hb = out(index_hb_ch2);
lb = out(index_lb_ch2);
Lmin = min(length(hb),length(lb));

ECGch_2 = bitshift(hb(1:Lmin),8) + lb(1:Lmin);
%ECGch_2 = ECGch_2-mean(ECGch_2);
%save('ECG_SAC_12082019','ECGch_2');   %% ARQUIVO QUE CONTERÁ O ECG EM AQUISIÇÃO %%
tending2 = toc(tstart2)
Fim_ReadSave_Buffer = [Fim_ReadSave_Buffer tending2];

tstart3 = tic;
Inicio_Filtra_Buffer = [Inicio_Filtra_Buffer tstart3];
%sinalf1 = FiltroECGMuscular_WaveletFs256_TesteArduino(ECGch_2',sfreq);
%sinalf1 = FiltroECGMuscular_WaveletFs256_TesteArduino(ECGch_2',sfreq,'db10');   %% APLICAÇÃO DE FILTRO PARA ELIMINAÇÃO DE RUÍDO DE ALTA FREQU
tending3 = toc(tstart3)
Fim_Filtra_Buffer = [Fim_Filtra_Buffer tending3];

tic
tstart = tic;
Inicio_Espera_Buffer = [Inicio_Espera_Buffer tstart];
Antigo_Buffer = 0;
Novo_Buffer = 0;
t = 0 ;
k = 1;
x = 0 ;
y = 0 ;
startSpot = 0;
interv = length(ECGch_2)/sfreq; % considering 1000 samples
step = 1/sfreq ; % lowering step has a number of cycles and then acquire more data
%figure;
Esperando_Buffer = 0;

TEMPO=TEMPO*10000000;
A1=tic;
while ( t < length(ECGch_2)/sfreq )
A2=tic;
A3=A2-A1;
TEMPO1=(A2-A1)/(10^7);
app.TempodeExecuoEditField.Value=double(TEMPO1);
 cC=counter1;
    if (cC==1)
        break;
    end
```

```
    if (A3>=TEMPO)
        break;
    end

    b = ECGch_2(k);
    %c = sinalf1(k);
    x = [ x, b ];
    y = [ y, b ];
    tt = (0:1:length(x)-1)/sfreq;
    plot(app.UIAxes,tt,x,'b') ;
    %plot(x) ;
      if ((t/step)-1000 < 0)
          startSpot = 0;
      else
          startSpot = (t/step)-500;
      end
      app.UIAxes.XLim=[startSpot/sfreq (t/step+50)/sfreq];
      app.UIAxes.YLim=[min(1.2*min(x),0) 1.2*max(x)];
      %axis([ startSpot/sfreq, (t/step+50)/sfreq, min(1.2*min(x),0) , 1.2*max(x) ]);
      %axis([ startSpot, t/step+50, 1.2*min(ECGch_2) , 1.2*max(ECGch_2) ]);

      switch app.GridCheckBox.Value
          case 1
             grid(app.UIAxes,'on');
          case 0
              grid(app.UIAxes,'off');
      end

      %global value1;
      %adr11=value1;
       %if adr11==1
% app.UIAxes.XGrid='on';
% app.UIAxes.YGrid='on';
       %end


      t = t + step;
      k = k + 1;
      drawnow;
      if s.BytesAvailable < s.InputBufferSize
          time = time + 1;
      else

          tending = toc(tstart)
          Fim_Espera_Buffer = [Fim_Espera_Buffer tending];
          tic;
          tstart2 = tic;
          Inicio_ReadSave_Buffer = [Inicio_ReadSave_Buffer tstart2];
          out = fread(s,s.BytesAvailable,'char');

          find_sync = 0;
          jj=1;
          while find_sync == 0;
            sync0 = dec2hex(out(jj));
            sync1 = dec2hex(out(jj+1));
            if sum(sync0 == 'A5') == 2 && sum(sync1 == '5A') == 2
                find_sync = 1;
            else jj = jj+1;
            end
          end

%           index_hb_ch1 = jj+14:17:length(out);
%           index_lb_ch1 = jj+15:17:length(out);

         index_hb_ch2 = jj+6:17:length(out);
         index_lb_ch2 = jj+7:17:length(out);

         hb = out(index_hb_ch2);
         lb = out(index_lb_ch2);
         Lmin = min(length(hb),length(lb));

         Novo_Buffer = bitshift(hb(1:Lmin),8) + lb(1:Lmin);
         %Novo_Buffer = Novo_Buffer-mean(Novo_Buffer);
         ECGch_2 = [ECGch_2; Novo_Buffer];
         %save('ECG_SAC_12082019','ECGch_2');
        tending2 = toc(tstart2)
        Fim_ReadSave_Buffer = [Fim_ReadSave_Buffer tending2];

        tstart3 = tic;
        Inicio_Filtra_Buffer = [Inicio_Filtra_Buffer tstart3];
        %Novo_Buffer_f = FiltroECGMuscular_WaveletFs256_TesteArduino(Novo_Buffer',sfreq);
        %Novo_Buffer_f = FiltroECGMuscular_WaveletFs256_TesteArduino(Novo_Buffer',sfreq,'FILTRO');
        tending3 = toc(tstart3)
        Fim_Filtra_Buffer = [Fim_Filtra_Buffer tending3];
        %sinalf1 = [sinalf1 Novo_Buffer_f];

        tic;
        tstart = tic;
        Inicio_Espera_Buffer = [Inicio_Espera_Buffer tstart];
         time = 0;
      end

      %pause(step)
end
save(NOME, 'DATA','x','y', 'ECGch_2', 'TEMPO1');
fclose(s);

                otherwise
                    Inicio_Espera_Buffer = [];
Fim_Espera_Buffer = [];
Inicio_ReadSave_Buffer = [];
Fim_ReadSave_Buffer = [];
Inicio_Filtra_Buffer = [];
Fim_Filtra_Buffer = [];
```

```
tic;
tstart = tic;
Inicio_Espera_Buffer = [Inicio_Espera_Buffer tstart];
serial_list = instrfind;
delete(serial_list);
s = serial(PORTA);   %% PORTA COM A QUAL O ARDUINO SE COMUNICA COM O COMPUTADOR  %%
s.InputBufferSize = 2*16384;
s.BaudRate = 57600;   %% TAXA DE TRANSMISSÃO ARDUINO - PORTA USB - COMPUTADOR
fopen(s);
parou = 0;
buffer = 0;
time = 0;
sfreq = 256;

while s.BytesAvailable < s.InputBufferSize
    time = time+1;
end
tending = toc(tstart)
Fim_Espera_Buffer = [Fim_Espera_Buffer tending];
tic;
tstart2 = tic;
Inicio_ReadSave_Buffer = [Inicio_ReadSave_Buffer tstart2];
out = fread(s,s.BytesAvailable,'char');

find_sync = 0;
jj=1;
while find_sync == 0;
    sync0 = dec2hex(out(jj));
    sync1 = dec2hex(out(jj+1));
    if sum(sync0 == 'A5') == 2 && sum(sync1 == '5A') == 2
        find_sync = 1;
    else jj = jj+1;
    end
end

% index_hb_ch1 = jj+14:17:length(out);
% index_lb_ch1 = jj+15:17:length(out);

index_hb_ch2 = jj+6:17:length(out);
index_lb_ch2 = jj+7:17:length(out);

hb = out(index_hb_ch2);
lb = out(index_lb_ch2);
Lmin = min(length(hb),length(lb));

ECGch_2 = bitshift(hb(1:Lmin),8) + lb(1:Lmin);
%ECGch_2 = ECGch_2-mean(ECGch_2);
%save('ECG_SAC_12082019','ECGch_2');   %% ARQUIVO QUE CONTERÁ O ECG EM AQUISIÇÃO %%
tending2 = toc(tstart2)
Fim_ReadSave_Buffer = [Fim_ReadSave_Buffer tending2];

tstart3 = tic;
Inicio_Filtra_Buffer = [Inicio_Filtra_Buffer tstart3];
%sinalf1 = FiltroECGMuscular_WaveletFs256_TesteArduino(ECGch_2',sfreq);


sinalf1 = FiltroECGMuscular_WaveletFs256_TesteArduino(ECGch_2',sfreq,app.FiltroWavelet256HZKnob.Value);   %% APLICAÇÃO DE FILTRO PARA ELIMINAÇ
sinalf2 = FiltroECGWandering_Wavelet3_16032018(ECGch_2',sfreq,app.FiltroWavelet256HZKnob.Value);   %% APLICAÇÃO DE FILTRO PARA ELIMINAÇÃO DE F
sinalf3 = FiltroECGMuscular_WaveletFs256_TesteArduino(sinalf2,sfreq,app.FiltroWavelet256HZKnob.Value);
global ENTERSIG;
ENTERSIG=ECGch_2;
f1=fftshift(abs(fft(ECGch_2)));
f2=fftshift(abs(fft(sinalf1)));
f3=fftshift(abs(fft(sinalf2)));
f4=fftshift(abs(fft(sinalf3)));

tending3 = toc(tstart3)
Fim_Filtra_Buffer = [Fim_Filtra_Buffer tending3];

tic
tstart = tic;
Inicio_Espera_Buffer = [Inicio_Espera_Buffer tstart];
Antigo_Buffer = 0;
Novo_Buffer = 0;
t = 0 ;
k = 1;
x = 0 ;
y = 0 ;
Z3 = 0 ;
Z4 = 0 ;
startSpot = 0;
interv = length(ECGch_2)/sfreq; % considering 1000 samples
step = 1/sfreq ; % lowering step has a number of cycles and then acquire more data
%figure;
Esperando_Buffer = 0;

TEMPO=TEMPO*10000000;
A1=tic;
while ( t < length(ECGch_2)/sfreq )
A2=tic;
A3=A2-A1;
TEMPO1=(A2-A1)/(10^7);
app.TempodeExecuoEditField.Value=double(TEMPO1);

 CZ=counter;
    if (CZ==1)
        break;
    end

    if (A3>TEMPO)
        break;
    end
```

```
    b = ECGch_2(k);
    c = sinalf1(k);
    DZ = sinalf2(k);
    EZ = sinalf3(k);


    x = [ x, b ]; % original
    y = [ y, c ]; % passa baixa
    Z3= [ Z3, DZ ]; %passa alta
    Z4= [ Z4, EZ ]; %serie


    tt = (0:1:length(x)-1)/sfreq;
    xf=linspace(-sfreq/2,sfreq/2,length(f1));
    %linspace(-sfreq/2,sfreq/2,length(d1));
    %fshift = (-length(z1)/2:length(z1)/2-1)*(sfreq/length(z1));

    if ((t/step)-1000 < 0)
          startSpot = 0;
      else
          startSpot = (t/step)-500;
      end


    switch app.DropDown_3.Value
        case 'Canal1'

   switch app.PassaBaixaButton.Value
          case 1

    switch app.SinalOriginalButton.Value
          case 1
    plot(app.UIAxes,tt,x,'b');
    end

    switch app.SinalFiltradoButton.Value
          case 1
    plot(app.UIAxes,tt,y,'r') ;
    end

    switch app.TodosButton.Value
          case 1
    plot(app.UIAxes,tt,x,'b',tt,y,'r') ;
    end

      switch app.AutomticoButton_2.Value
          case 1
     app.UIAxes.XLim=[startSpot/sfreq (t/step+50)/sfreq];
     app.UIAxes.YLim=[min(1.4*min(y)) 1.2*max(x)];
    end

        end

switch app.PassaAltaButton.Value
          case 1
     switch app.SinalOriginalButton.Value
          case 1
    plot(app.UIAxes,tt,x,'b') ;
    end

    switch app.SinalFiltradoButton.Value
          case 1
    plot(app.UIAxes,tt,Z3,'g') ;
    end

    switch app.TodosButton.Value
          case 1
    plot(app.UIAxes,tt,x,'b',tt,Z3,'g');
    end

      switch app.AutomticoButton_2.Value
          case 1
     app.UIAxes.XLim=[startSpot/sfreq (t/step+50)/sfreq];
     app.UIAxes.YLim=[min(1.4*min(Z3)) 1.2*max(x)];
    end

end

   switch app.SrieButton.Value
          case 1
     switch app.SinalOriginalButton.Value
          case 1
    plot(app.UIAxes,tt,x,'b') ;
    end

    switch app.SinalFiltradoButton.Value
          case 1
    plot(app.UIAxes,tt,Z4,'m') ;
    end

    switch app.TodosButton.Value
          case 1
    plot(app.UIAxes,tt,x,'b',tt,Z4,'m');
    end
      switch app.AutomticoButton_2.Value
          case 1
     app.UIAxes.XLim=[startSpot/sfreq (t/step+50)/sfreq];
     app.UIAxes.YLim=[min(1.4*min(Z4)) 1.2*max(x)];
    end
        end

switch app.ComparaoButton.Value
          case 1
```

```
    switch app.SinalOriginalButton.Value
          case 1
    plot(app.UIAxes,tt,x,'b') ;
    end

    switch app.SinalFiltradoButton.Value
          case 1
    plot(app.UIAxes,tt,y,'r',tt,Z3,'g') ;
    end

    switch app.TodosButton.Value
          case 1
    plot(app.UIAxes,tt,x,'b',tt,Z3,'g',tt,y,'r');
    end
  switch app.AutomticoButton_2.Value
          case 1
     app.UIAxes.XLim=[startSpot/sfreq (t/step+50)/sfreq];
     app.UIAxes.YLim=[min(1.4*min(Z3)) 1.2*max(x)];
    end

end

    switch app.ManualButton.Value
          case 1
    app.UIAxes.XLim=[app.XEditField.Value app.XEditField_2.Value];
    app.UIAxes.YLim=[app.YEditField.Value app.XEditField_3.Value];
    end



    case 'Canal2'
        switch app.PassaBaixaButton.Value
          case 1

    switch app.SinalOriginalButton.Value
          case 1
    plot(app.UIAxes,xf,f1,'b');
    end

    switch app.SinalFiltradoButton.Value
          case 1
    plot(app.UIAxes,xf,f2,'r');
    end

    switch app.TodosButton.Value
          case 1
    plot(app.UIAxes,xf,f1,'b',xf,f2,'r') ;
    end

      switch app.AutomticoButton_2.Value
          case 1
     app.UIAxes.XLim=[0 sfreq/2];
     app.UIAxes.YLim=[min(1.4*min(f2)) 1.2*max(f1)];
    end

        end

  switch app.PassaAltaButton.Value
          case 1
     switch app.SinalOriginalButton.Value
          case 1
    plot(app.UIAxes,xf,f1,'b') ;
    end

    switch app.SinalFiltradoButton.Value
          case 1
    plot(app.UIAxes,xf,f3,'g') ;
    end

    switch app.TodosButton.Value
          case 1
    plot(app.UIAxes,xf,f1,'b',xf,f3,'g');
    end

      switch app.AutomticoButton_2.Value
          case 1
     app.UIAxes.XLim=[0 sfreq/2];
     app.UIAxes.YLim=[min(1.4*min(f3)) 1.2*max(f1)];
    end

end

 switch app.SrieButton.Value
          case 1
     switch app.SinalOriginalButton.Value
          case 1
    plot(app.UIAxes,xf,f1,'b') ;
    end

    switch app.SinalFiltradoButton.Value
          case 1
    plot(app.UIAxes,xf,f4,'m') ;
    end

    switch app.TodosButton.Value
          case 1
    plot(app.UIAxes,xf,f1,'b',xf,f4,'m');
    end
      switch app.AutomticoButton_2.Value
          case 1
     app.UIAxes.XLim=[0 sfreq/2];
     app.UIAxes.YLim=[min(1.4*min(f4)) 1.2*max(f1)];
    end
         end
```

```
          switch app.ComparaoButton.Value
           case 1

    switch app.SinalOriginalButton.Value
          case 1
    plot(app.UIAxes,xf,f1,'b') ;
    end

    switch app.SinalFiltradoButton.Value
          case 1
    plot(app.UIAxes,xf,f2,'r',xf,f3,'g');
    end

    switch app.TodosButton.Value
          case 1
    plot(app.UIAxes,xf,f1,'b',xf,f3,'g',xf,f2,'r');
    end
  switch app.AutomticoButton_2.Value
          case 1
     app.UIAxes.XLim=[0 sfreq/2];
     app.UIAxes.YLim=[min(1.4*min(f3)) 1.2*max(f1)];
    end

end

    switch app.ManualButton.Value
          case 1
    app.UIAxes.XLim=[app.XEditField.Value app.XEditField_2.Value];
    app.UIAxes.YLim=[app.YEditField.Value app.XEditField_3.Value];
    end

    end



      switch app.GridCheckBox.Value
          case 1
             grid(app.UIAxes,'on');
          case 0
              grid(app.UIAxes,'off');
      end

%       global value;
%       adr1=value;
%        if adr1==1
% app.UIAxes.XGrid='on';
% app.UIAxes.YGrid='on';
%     end

      t = t + step;
      k = k + 1;
      drawnow;
      if s.BytesAvailable < s.InputBufferSize
          time = time + 1;
      else

          tending = toc(tstart)
          Fim_Espera_Buffer = [Fim_Espera_Buffer tending];
          tic;
          tstart2 = tic;
          Inicio_ReadSave_Buffer = [Inicio_ReadSave_Buffer tstart2];
          out = fread(s,s.BytesAvailable,'char');

          find_sync = 0;
          jj=1;
          while find_sync == 0;
            sync0 = dec2hex(out(jj));
            sync1 = dec2hex(out(jj+1));
            if sum(sync0 == 'A5') == 2 && sum(sync1 == '5A') == 2
                find_sync = 1;
            else jj = jj+1;
            end
          end

%           index_hb_ch1 = jj+14:17:length(out);
%           index_lb_ch1 = jj+15:17:length(out);

         index_hb_ch2 = jj+6:17:length(out);
         index_lb_ch2 = jj+7:17:length(out);

         hb = out(index_hb_ch2);
         lb = out(index_lb_ch2);
         Lmin = min(length(hb),length(lb));

         Novo_Buffer = bitshift(hb(1:Lmin),8) + lb(1:Lmin);
         %Novo_Buffer = Novo_Buffer-mean(Novo_Buffer);
         ECGch_2 = [ECGch_2; Novo_Buffer];
        % save('ECG_SAC_12082019','ECGch_2');
        tending2 = toc(tstart2)
        Fim_ReadSave_Buffer = [Fim_ReadSave_Buffer tending2];

        tstart3 = tic;
        Inicio_Filtra_Buffer = [Inicio_Filtra_Buffer tstart3];
        %Novo_Buffer_f = FiltroECGMuscular_WaveletFs256_TesteArduino(Novo_Buffer',sfreq);

        Novo_Buffer_f1 = FiltroECGMuscular_WaveletFs256_TesteArduino(Novo_Buffer',sfreq,app.FiltroWavelet256HZKnob.Value);
        Novo_Buffer_f2 = FiltroECGWandering_Wavelet3_16032018(Novo_Buffer',sfreq,app.FiltroWavelet256HZKnob.Value);
        Novo_Buffer_f3 = FiltroECGMuscular_WaveletFs256_TesteArduino(Novo_Buffer_f2,sfreq,app.FiltroWavelet256HZKnob.Value);


        tending3 = toc(tstart3)
        Fim_Filtra_Buffer = [Fim_Filtra_Buffer tending3];
```

```
        sinalf1 = [sinalf1 Novo_Buffer_f1];
        sinalf2 = [sinalf2 Novo_Buffer_f2];
        sinalf3 = [sinalf3 Novo_Buffer_f3];
        ENTERSIG=ECGch_2;
        f1=fftshift(abs(fft(ECGch_2)));
        f2=fftshift(abs(fft(sinalf1)));
        f3=fftshift(abs(fft(sinalf2)));
        f4=fftshift(abs(fft(sinalf3)));

        tic;
        tstart = tic;
        Inicio_Espera_Buffer = [Inicio_Espera_Buffer tstart];
         time = 0;
      end

      %pause(step)
end
save(NOME, 'DATA','x','y', 'ECGch_2', 'sinalf1' , 'TEMPO1');
fclose(s);

            end
        end

        % Callback function
        function StopButtonPushed(app, event)
        global counter;
        global counter1;
        counter=1;
        counter1=1;
        end

        % Callback function
        function FinalizarButtonPushed(app, event)
            close all;
            closereq;
        end

        % Callback function
        function NovoPacienteButtonPushed(app, event)
            app.NomedoPacienteEditField.Value=' ';
            app.TemposEditField.Value=10;
        end

        % Callback function
        function GridCheckBoxValueChanged(app, event)
           global value;
            value = app.GridCheckBox.Value;
            global value1;
            value1 = app.GridCheckBox.Value;

        end

        % Callback function
        function ProcurarPortaCOMButtonPushed(app, event)
%               if ~usejava('jvm')
%          error(message('instrument:instrhwinfo:nojvm'));
%      end

     % Determine the jar file version.
%      jarFileVersion = com.mathworks.toolbox.instrument.Instrument.jarVersion;
%
%           fields = {'AvailableSerialPorts',...
%                     'JarFileVersion',...
%                     'ObjectConstructorName',...
%                     'SerialPorts'};
%
%      try
%          s = javaObject('com.mathworks.toolbox.instrument.SerialComm','temp');
%          tempOut = hardwareInfo(s);
%          dispose(s)
%      catch
%          tempOut = {{'COM1'}, '', {}, {}}'
%      end
%
%      list = cell(tempOut);
%      list = list{1};
%      [r,c] = size(list);
%      if r==0
%           list = {'COM1'}; % if there are no ports leave something in the menu
%      end
 list =seriallist;
 num=length( list);
 app.aux1=app.aux1+1;
 if app.aux1>num
     app.aux1=1;
 end
     a=list{1,app.aux1};
     % update the ports menu to only contaim valid COMs
     app.PortaSerialDropDown.Value=a;
        end

        % Button pushed function: IniciarButton_2
        function IniciarButton_2Pushed(app, event)
%%   ATUALIZA AS LISTAS DA 1º ITERAÇÃO
   profile on

NIVEL1=str2num(app.NveldeDecompDropDown.Value);

strA={'cAr1','cAr2','cAr3','cAr4','cAr5','cAr6','cAr7','cAr8','cAr9','cAr10'};
strD={'cDr1','cDr2','cDr3','cDr4','cDr5','cDr6','cDr7','cDr8','cDr9','cDr10'};
data={};
for i=1:NIVEL1
    jusd={strA{1,i},false,strD{1,i},false};
    for k=1:4
```

```
 data{i,k} = jusd{1,k};
    end
end
data{1,2}=true;
app.UITable.Data = data;

tems1={};
tems2={};
for i=1:NIVEL1
 tems1{1,i}= strA{1,i};
 tems2{1,i}= strD{1,i};
end

switch app.AproximaoButton.Value
    case 1
app.opesDropDown.Items=tems1;
end

switch app.DetalheButton.Value
    case 1
app.opesDropDown.Items=tems2;
end


%% DECLARANDO VARIÁVEIS
conter=0;
global conter1;
conter1=0;

global ENTERSIG;

%% INÍCIO DO LOOP
  while 1

 %% ATUALIZANDO O FILTRO A SER UTILIZADO
       switch app.DaubechiesButton.Value
          case 1
            app.FiltrosKnob.Items={'db1','db2','db3','db4','db5','db6','db7','db8','db9','db10','db11','db12','db13','db14','db15','db16'};
            end

            switch app.CoifletsButton.Value
          case 1
            app.FiltrosKnob.Items={'coif1','coif2','coif3','coif4','coif5'};
            end

            switch app.SymletsButton.Value
          case 1
            app.FiltrosKnob.Items={'sym2','sym3','sym4','sym6','sym7','sym8','sym9','sym10','sym11','sym12','sym13','sym14','sym15','sym16'};
            end

           switch app.FejerKorovkinfiltersButton.Value
          case 1
            app.FiltrosKnob.Items={'fk4','fk6','fk8','fk14','fk22'};
            end

            switch app.DiscreteMeyerButton.Value
          case 1
            app.FiltrosKnob.Items={'dmey','dmey'};
            end

           switch app.BiorthogonalButton.Value
          case 1
            app.FiltrosKnob.Items={'bior1.1', 'bior1.3', 'bior1.5','bior2.2', 'bior2.4', 'bior2.6', 'bior2.8','bior3.1', 'bior3.3', 'bior3.5',
            end

         switch app.ReverseBiorthogonalButton.Value
          case 1
            app.FiltrosKnob.Items={'rbio1.1', 'rbio1.3', 'rbio1.5','rbio2.2', 'rbio2.4', 'rbio2.6', 'rbio2.8','rbio3.1', 'rbio3.3', 'rbio3.5',
         end



 %% VERIFICA SE O STOP FOI ACIONADO
conter=conter1 ;


% %% VERIFICA SE O SINAL É INSERIDO NO MODO MANUAL
%     switch app.ManualButton_2.Value
%         case 1
%          load(app.DiretriodoarquivoEditField.Value);
%          ENTRADA=eval(app.NomedaVarivelEditField.Value);
%
%         switch app.DiretriodoarquivoEditField.Value
%             case ''
%         conter=1;
%         end
%     end
%
% %% VERIFICA SE O SINAL É INSERIDO NO MODO AUTOMÁTICO
%     switch app.AutomticoButton_2.Value
%         case 1
%          load(app.DiretriodoarquivoEditField.Value);
%          ENTRADA=eval(app.NomedaVarivelEditField.Value);
%
%     end
%



 load(app.DiretriodoarquivoEditField.Value);
 ENTRADA=eval(app.NomedaVarivelEditField.Value);

 if coder.internal.isConst(iscolumn(ENTRADA)) && iscolumn(ENTRADA)
    ENTRADA = ENTRADA';
```

```
else
    ENTRADA = ENTRADA;
end



%% PARAR CASO STOP FOR ACIONADO OU ERRO DE DIRETÓRIO NO MODO MANUAL

    if (conter==1)
        break;
    end


%% INÍCIO DA DECOMPOSIÇÃO MULTINÍVEL

NIVEL=str2num(app.NveldeDecompDropDown.Value);
Signal=ENTRADA;
TAMA=length(Signal);
app.TamanhoEditField.Value=TAMA;
DB=app.FiltrosKnob.Value;

[c,l] = wavedec(Signal,NIVEL,DB);
for i=1:NIVEL
cD{i,:} = detcoef(c,l,i);
cA{i,:} = appcoef(c,l,DB,i);

cDr{i,:} = wrcoef('d',c,l,DB,i);
cAr{i,:} = wrcoef('a',c,l,DB,i);
end


%% PLOTANDO OS RESULTADOS DO PROCEDIMENTO DE RECUPERAÇÃO
strA={'cAr1','cAr2','cAr3','cAr4','cAr5','cAr6','cAr7','cAr8','cAr9','cAr10'};
strD={'cDr1','cDr2','cDr3','cDr4','cDr5','cDr6','cDr7','cDr8','cDr9','cDr10'};

switch app.AproximaoButton.Value
    case 1
app.opesDropDown.Items=tems1;
end

switch app.DetalheButton.Value
    case 1
app.opesDropDown.Items=tems2;
end


data= app.UITable.Data;



%% PLOTANDO OS RESULTADOS DO PROCEDIMENTO DE RECUPERAÇÃO
switch app.cArecDrCheckBox.Value
    case 1
figure;
for i=1:2*NIVEL

switch  rem(i,2)==0
    case 0
subplot(NIVEL,2,i);
plot(cAr{((i+1)/2)});
title(strA{1,((i+1)/2)})
    case 1
subplot(NIVEL,2,i)
plot(cDr{(i/2)})
title(strD{1,(i/2)})
end
end
app.cArecDrCheckBox.Value=0;
end

% PLOTANDO AS COMPONENTES
for i=1:NIVEL
switch app.opesDropDown.Value
    case strA{1,i}
   compt=cAr{i,:};
   asef1=strA{1,i};
    case strD{1,i}
   compt=cDr{i,:};
   asef1=strD{1,i};
end
end
asef=strcat('Componente (',asef1,')');
plot(app.UIAxes2,compt,'b');
  legend(app.UIAxes2,{asef});
switch app.AutomticoButton_3.Value
          case 1
    app.UIAxes2.XLim=[0 length(compt)];
    app.UIAxes2.YLimMode = 'auto' ;
    %app.UIAxes2.YLim='auto'
    end

    switch app.ManualButton_3.Value
          case 1
    app.UIAxes2.XLim=[app.XEditField_6.Value app.XEditField_4.Value];
    app.UIAxes2.YLim=[app.YEditField_2.Value app.XEditField_5.Value];
    end



%% PLOTANDO TODAS COMPONENTES DO PROCEDIMENTO DE DECOMPOSIÇÃO
switch app.cAecDCheckBox.Value
    case 1
strA1={'cA1','cA2','cA3','cA4','cA5','cA6','cA7','cA8','cA9','cA10'};
strD1={'cD1','cD2','cD3','cD4','cD5','cD6','cD7','cD8','cD9','cD10'};
```

```
figure;
for i=1:2*NIVEL

switch  rem(i,2)==0
    case 0
subplot(NIVEL,2,i);
plot(cA{((i+1)/2)});
title(strA1{1,((i+1)/2)})
    case 1
subplot(NIVEL,2,i)
plot(cD{(i/2)});
title(strD1{1,(i/2)})
end
end
app.cAecDCheckBox.Value=0;
end


%% ATUALIZANDO AS LISTAS

switch NIVEL
    case NIVEL1
        QWER='OK';
otherwise

data={};
for i=1:NIVEL
    jusd={strA{1,i},false,strD{1,i},false};
    for k=1:4
 data{i,k} = jusd{1,k};
    end
end
data{1,2}=true;
app.UITable.Data = data;

tems1={};
tems2={};
for i=1:NIVEL
 tems1{1,i}= strA{1,i};
 tems2{1,i}= strD{1,i};
end

switch app.AproximaoButton.Value
    case 1
app.opesDropDown.Items=tems1;
end

switch app.DetalheButton.Value
    case 1
app.opesDropDown.Items=tems2;
end

NIVEL1=NIVEL;
end

%% OPERAÇÕES
sd=1;
sd1=1;
for i=1:NIVEL
    switch data{i,2}
        case 1
          plotar1=cAr{i,:};
          plotaA{sd,:}=plotar1;
%           save('sinalA','plotaA')
    sd=sd+1;
    end

    switch data{i,4}
        case 1
          plota1=cDr{i,:};
          plotaD{sd1,:}=plota1;
%           save('sinalD','plotaD')
    sd1=sd1+1;
    end
end


Soma1=0;
if sd>1
MA=sd-1;
for i=1:MA
 Soma1= Soma1+plotaA{i,:};
end
end

Soma2=0;
if sd1>1
MD=sd1-1;
for i=1:MD
 Soma2= Soma2+plotaD{i,:};
end
end


saida=Soma1+Soma2;
saida1=ENTRADA-saida;

switch app.SomaButton.Value
    case 1
    hold(app.UIAxes2_2,'off');
  plot(app.UIAxes2_2,saida,'b');
  legend(app.UIAxes2_2,{'Sinal Resultante(soma)'});
```

```
  switch app.SinalOriginalCheckBox.Value
          case 1
        hold(app.UIAxes2_2,'on');
    plot(app.UIAxes2_2,ENTRADA,'c');
    legend(app.UIAxes2_2,{'Sinal Resultante(soma)','Sinal Original'});
  end
 %###
  switch app.Grfico1CheckBox.Value
          case 1
        hold(app.UIAxes2_2,'on');
    plot(app.UIAxes2_2,compt,'g');
    switch app.SinalOriginalCheckBox.Value
          case 1
      legend(app.UIAxes2_2,{'Sinal Resultante(soma)','Sinal Original','Sinal Gráfico 1'});
        case 0
         legend(app.UIAxes2_2,{'Sinal Resultante(soma)','Sinal Gráfico 1'});
    end

  end
  %##

   switch app.AutomticoButton_4.Value
          case 1
    app.UIAxes2_2.XLim=[0 length(saida)];
    %app.UIAxes2_2.YLim=[1.2*min(saida) 1.2*max(saida)];
    app.UIAxes2_2.YLimMode = 'auto' ;
    end
end




switch app.SubtraoButton.Value
    case 1
          hold(app.UIAxes2_2,'off');
  plot(app.UIAxes2_2,saida1,'r');
  legend(app.UIAxes2_2,{'Sinal Resultante(subtração)'});

  switch app.SinalOriginalCheckBox.Value
          case 1
        hold(app.UIAxes2_2,'on');
    plot(app.UIAxes2_2,ENTRADA,'c');
    legend(app.UIAxes2_2,{'Sinal Resultante(subtração)','Sinal Original'});
  end

  switch app.Grfico1CheckBox.Value
          case 1
        hold(app.UIAxes2_2,'on');
    plot(app.UIAxes2_2,compt,'g');
    switch app.SinalOriginalCheckBox.Value
          case 1
      legend(app.UIAxes2_2,{'Sinal Resultante(subtração)','Sinal Original','Sinal Gráfico 1'});
        case 0
         legend(app.UIAxes2_2,{'Sinal Resultante(subtração)','Sinal Gráfico 1'});
    end

  end

  switch app.AutomticoButton_4.Value
          case 1
    app.UIAxes2_2.XLim=[0 length(saida1)];
    %app.UIAxes2_2.YLim=[1.2*min(saida1) 1.2*max(saida1)];
    app.UIAxes2_2.YLimMode = 'auto' ;
  end
end



switch app.ManualButton_4.Value
          case 1
    app.UIAxes2_2.XLim=[app.XEditField_9.Value app.XEditField_7.Value];
    app.UIAxes2_2.YLim=[app.YEditField_3.Value app.XEditField_8.Value];
end

drawnow;

 profile viewer  % view the timing report
  p = profile("info");
   profile off
  profsave(p,"myresults")
break;
break;
break;
%% GRAFICO COMPONENTE
  end



        end

        % Button pushed function: BrowseButton
        function BrowseButtonPushed(app, event)

           [file,path] = uigetfile;
           strs = strcat(path,file);
           if path==0
           app.SelecioneoarquivomatcomosinalLabel.Text='Selecione o arquivo ".mat" com o sinal';
           app.DiretriodoarquivoEditField.Value='';
           else
           app.SelecioneoarquivomatcomosinalLabel.Text='';
           app.DiretriodoarquivoEditField.Value=strs;
           end

        end
```

```
        % Callback function: AtualizarButton, IniciarButton_3
        function AtualizarButtonValueChanged(app, event)
             switch app.DaubechiesButton.Value
          case 1
            app.FiltrosKnob.Items={'db1','db2','db3','db4','db5','db6','db7','db8','db9','db10','db11','db12','db13','db14','db15','db16'};
            end

            switch app.CoifletsButton.Value
          case 1
            app.FiltrosKnob.Items={'coif1','coif2','coif3','coif4','coif5'};
            end

            switch app.SymletsButton.Value
          case 1
            app.FiltrosKnob.Items={'sym2','sym3','sym4','sym6','sym7','sym8','sym9','sym10','sym11','sym12','sym13','sym14','sym15','sym16'};
            end

           switch app.FejerKorovkinfiltersButton.Value
          case 1
            app.FiltrosKnob.Items={'fk4','fk6','fk8','fk14','fk22'};
            end

            switch app.DiscreteMeyerButton.Value
          case 1
            app.FiltrosKnob.Items={'dmey','dmey'};
            end

           switch app.BiorthogonalButton.Value
          case 1
            app.FiltrosKnob.Items={'bior1.1', 'bior1.3', 'bior1.5','bior2.2', 'bior2.4', 'bior2.6', 'bior2.8','bior3.1', 'bior3.3', 'bior3.5',
            end

         switch app.ReverseBiorthogonalButton.Value
          case 1
            app.FiltrosKnob.Items={'rbior1.1', 'rbior1.3', 'rbior1.5','rbior2.2', 'rbior2.4', 'rbior2.6', 'rbior2.8','rbior3.1', 'rbior3.3', '
         end


NIVEL=str2num(app.NveldeDecompDropDown.Value);

strA={'cAr1','cAr2','cAr3','cAr4','cAr5','cAr6','cAr7','cAr8','cAr9','cAr10'};
strD={'cDr1','cDr2','cDr3','cDr4','cDr5','cDr6','cDr7','cDr8','cDr9','cDr10'};
data={};
for i=1:NIVEL
    jusd={strA{1,i},false,strD{1,i},false};
    for k=1:4
 data{i,k} = jusd{1,k};
    end
end
data{1,2}=true;
app.UITable.Data = data;

tems1={};
tems2={};
for i=1:NIVEL
 tems1{1,i}= strA{1,i};
 tems2{1,i}= strD{1,i};
end

switch app.AproximaoButton.Value
    case 1
app.opesDropDown.Items=tems1;
end

switch app.DetalheButton.Value
    case 1
app.opesDropDown.Items=tems2;
end
        end

        % Value changed function: StopButton_2
        function StopButton_2ValueChanged(app, event)
        global conter1;
        conter1=1;

        end
    end

    % Component initialization
    methods (Access = private)

        % Create UIFigure and components
        function createComponents(app)

            % Create SinalECGUIFigure and hide until all components are created
            app.SinalECGUIFigure = uifigure('Visible', 'off');
            app.SinalECGUIFigure.Position = [100 100 1381 720];
            app.SinalECGUIFigure.Name = 'Sinal ECG';

            % Create TabGroup
            app.TabGroup = uitabgroup(app.SinalECGUIFigure);
            app.TabGroup.Position = [-3 -1 1385 722];

            % Create AnliseWaveletTab
            app.AnliseWaveletTab = uitab(app.TabGroup);
            app.AnliseWaveletTab.Title = 'Análise Wavelet ';

            % Create UIAxes2
            app.UIAxes2 = uiaxes(app.AnliseWaveletTab);
            title(app.UIAxes2, 'Componente do Sinal')
            xlabel(app.UIAxes2, 'Tempo')
            ylabel(app.UIAxes2, 'Amplitude')
            app.UIAxes2.FontSize = 14;
```

```
app.UIAxes2.Position = [552 307 496 223];

% Create UIAxes2_2
app.UIAxes2_2 = uiaxes(app.AnliseWaveletTab);
title(app.UIAxes2_2, 'Sinal Filtrado')
xlabel(app.UIAxes2_2, 'Tempo')
ylabel(app.UIAxes2_2, 'Amplitude')
app.UIAxes2_2.FontSize = 14;
app.UIAxes2_2.Position = [382 73 663 223];

% Create Panel
app.Panel = uipanel(app.AnliseWaveletTab);
app.Panel.BackgroundColor = [0.902 0.902 0.902];
app.Panel.FontSize = 14;
app.Panel.Position = [1 1 373 616];

% Create FiltrosKnobLabel
app.FiltrosKnobLabel = uilabel(app.Panel);
app.FiltrosKnobLabel.HorizontalAlignment = 'center';
app.FiltrosKnobLabel.FontSize = 16;
app.FiltrosKnobLabel.FontWeight = 'bold';
app.FiltrosKnobLabel.FontColor = [0.149 0.149 0.149];
app.FiltrosKnobLabel.Position = [160 22 55 22];
app.FiltrosKnobLabel.Text = 'Filtros';

% Create FiltrosKnob
app.FiltrosKnob = uiknob(app.Panel, 'discrete');
app.FiltrosKnob.Items = {'db1', 'db2', 'db3', 'db4', 'db5', 'db6', 'db7', 'db8', 'db9', 'db10', 'db11', 'db12', 'db13', 'db14', 'c
app.FiltrosKnob.FontSize = 14;
app.FiltrosKnob.FontWeight = 'bold';
app.FiltrosKnob.FontColor = [0 0 1];
app.FiltrosKnob.Position = [94 59 184 184];
app.FiltrosKnob.Value = 'db1';

% Create Panel_4
app.Panel_4 = uipanel(app.Panel);
app.Panel_4.BackgroundColor = [0.8 0.8 0.8];
app.Panel_4.Position = [209 411 149 69];

% Create NveldeDecompDropDownLabel
app.NveldeDecompDropDownLabel = uilabel(app.Panel_4);
app.NveldeDecompDropDownLabel.HorizontalAlignment = 'right';
app.NveldeDecompDropDownLabel.FontSize = 14;
app.NveldeDecompDropDownLabel.FontWeight = 'bold';
app.NveldeDecompDropDownLabel.Position = [15 38 122 22];
app.NveldeDecompDropDownLabel.Text = 'Nível de Decomp.';

% Create NveldeDecompDropDown
app.NveldeDecompDropDown = uidropdown(app.Panel_4);
app.NveldeDecompDropDown.Items = {'1', '2', '3', '4', '5', '6', '7', '8', '9', '10'};
app.NveldeDecompDropDown.FontSize = 14;
app.NveldeDecompDropDown.FontWeight = 'bold';
app.NveldeDecompDropDown.FontColor = [0 0 1];
app.NveldeDecompDropDown.Position = [44 10 46 22];
app.NveldeDecompDropDown.Value = '1';

% Create GerarGrficosPanel
app.GerarGrficosPanel = uipanel(app.Panel);
app.GerarGrficosPanel.TitlePosition = 'centertop';
app.GerarGrficosPanel.Title = 'Gerar Gráficos';
app.GerarGrficosPanel.BackgroundColor = [0.8 0.8 0.8];
app.GerarGrficosPanel.FontWeight = 'bold';
app.GerarGrficosPanel.FontSize = 14;
app.GerarGrficosPanel.Position = [209 312 149 85];

% Create cAecDCheckBox
app.cAecDCheckBox = uicheckbox(app.GerarGrficosPanel);
app.cAecDCheckBox.Text = 'cA e cD';
app.cAecDCheckBox.FontSize = 14;
app.cAecDCheckBox.FontWeight = 'bold';
app.cAecDCheckBox.FontColor = [0 0 1];
app.cAecDCheckBox.Position = [9 33 73 22];
app.cAecDCheckBox.Value = true;

% Create cArecDrCheckBox
app.cArecDrCheckBox = uicheckbox(app.GerarGrficosPanel);
app.cArecDrCheckBox.Text = 'cAr e cDr';
app.cArecDrCheckBox.FontSize = 14;
app.cArecDrCheckBox.FontWeight = 'bold';
app.cArecDrCheckBox.FontColor = [0 0 1];
app.cArecDrCheckBox.Position = [9 7 84 22];
app.cArecDrCheckBox.Value = true;

% Create FamliasWaveletButtonGroup
app.FamliasWaveletButtonGroup = uibuttongroup(app.Panel);
app.FamliasWaveletButtonGroup.TitlePosition = 'centertop';
app.FamliasWaveletButtonGroup.Title = 'Famílias Wavelet';
app.FamliasWaveletButtonGroup.BackgroundColor = [0.8 0.8 0.8];
app.FamliasWaveletButtonGroup.FontWeight = 'bold';
app.FamliasWaveletButtonGroup.FontSize = 16;
app.FamliasWaveletButtonGroup.Position = [18 307 182 285];

% Create DaubechiesButton
app.DaubechiesButton = uiradiobutton(app.FamliasWaveletButtonGroup);
app.DaubechiesButton.Text = 'Daubechies';
app.DaubechiesButton.FontSize = 14;
app.DaubechiesButton.FontWeight = 'bold';
app.DaubechiesButton.FontColor = [0.6392 0.0784 0.1804];
app.DaubechiesButton.Position = [11 223 100 22];
app.DaubechiesButton.Value = true;

% Create CoifletsButton
app.CoifletsButton = uiradiobutton(app.FamliasWaveletButtonGroup);
app.CoifletsButton.Text = 'Coiflets';
```

```
app.CoifletsButton.FontSize = 14;
app.CoifletsButton.FontWeight = 'bold';
app.CoifletsButton.FontColor = [0.6392 0.0784 0.1804];
app.CoifletsButton.Position = [11 195 73 22];

% Create SymletsButton
app.SymletsButton = uiradiobutton(app.FamliasWaveletButtonGroup);
app.SymletsButton.Text = 'Symlets';
app.SymletsButton.FontSize = 14;
app.SymletsButton.FontWeight = 'bold';
app.SymletsButton.FontColor = [0.6392 0.0784 0.1804];
app.SymletsButton.Position = [11 169 75 22];

% Create FejerKorovkinfiltersButton
app.FejerKorovkinfiltersButton = uiradiobutton(app.FamliasWaveletButtonGroup);
app.FejerKorovkinfiltersButton.Text = 'Fejer-Korovkin filters';
app.FejerKorovkinfiltersButton.FontSize = 14;
app.FejerKorovkinfiltersButton.FontWeight = 'bold';
app.FejerKorovkinfiltersButton.FontColor = [0.6392 0.0784 0.1804];
app.FejerKorovkinfiltersButton.Position = [11 142 163 22];

% Create DiscreteMeyerButton
app.DiscreteMeyerButton = uiradiobutton(app.FamliasWaveletButtonGroup);
app.DiscreteMeyerButton.Text = 'Discrete Meyer';
app.DiscreteMeyerButton.FontSize = 14;
app.DiscreteMeyerButton.FontWeight = 'bold';
app.DiscreteMeyerButton.FontColor = [0.6392 0.0784 0.1804];
app.DiscreteMeyerButton.Position = [11 112 122 22];

% Create BiorthogonalButton
app.BiorthogonalButton = uiradiobutton(app.FamliasWaveletButtonGroup);
app.BiorthogonalButton.Text = 'Biorthogonal';
app.BiorthogonalButton.FontSize = 14;
app.BiorthogonalButton.FontWeight = 'bold';
app.BiorthogonalButton.FontColor = [0.6392 0.0784 0.1804];
app.BiorthogonalButton.Position = [11 83 109 22];

% Create ReverseBiorthogonalButton
app.ReverseBiorthogonalButton = uiradiobutton(app.FamliasWaveletButtonGroup);
app.ReverseBiorthogonalButton.Text = 'Reverse Biorthogonal';
app.ReverseBiorthogonalButton.FontSize = 14;
app.ReverseBiorthogonalButton.FontWeight = 'bold';
app.ReverseBiorthogonalButton.FontColor = [0.6392 0.0784 0.1804];
app.ReverseBiorthogonalButton.Position = [11 53 167 22];

% Create AtualizarButton
app.AtualizarButton = uibutton(app.FamliasWaveletButtonGroup, 'state');
app.AtualizarButton.ValueChangedFcn = createCallbackFcn(app, @AtualizarButtonValueChanged, true);
app.AtualizarButton.Text = 'Atualizar';
app.AtualizarButton.BackgroundColor = [0 0 1];
app.AtualizarButton.FontWeight = 'bold';
app.AtualizarButton.FontColor = [1 1 1];
app.AtualizarButton.Position = [41 13 100 22];

% Create SinalInseridoButtonGroup
app.SinalInseridoButtonGroup = uibuttongroup(app.Panel);
app.SinalInseridoButtonGroup.TitlePosition = 'centertop';
app.SinalInseridoButtonGroup.Title = ' Sinal  Inserido ';
app.SinalInseridoButtonGroup.BackgroundColor = [0.8 0.8 0.8];
app.SinalInseridoButtonGroup.FontWeight = 'bold';
app.SinalInseridoButtonGroup.FontSize = 16;
app.SinalInseridoButtonGroup.Position = [209 497 149 93];

% Create ManualButton_2
app.ManualButton_2 = uiradiobutton(app.SinalInseridoButtonGroup);
app.ManualButton_2.Text = 'Manual';
app.ManualButton_2.FontSize = 14;
app.ManualButton_2.FontWeight = 'bold';
app.ManualButton_2.FontColor = [0 0 1];
app.ManualButton_2.Position = [11 42 71 22];
app.ManualButton_2.Value = true;

% Create AutomticoButton_2
app.AutomticoButton_2 = uiradiobutton(app.SinalInseridoButtonGroup);
app.AutomticoButton_2.Text = 'Automático';
app.AutomticoButton_2.FontSize = 14;
app.AutomticoButton_2.FontWeight = 'bold';
app.AutomticoButton_2.FontColor = [0 0 1];
app.AutomticoButton_2.Position = [8 11 99 22];

% Create Panel_2
app.Panel_2 = uipanel(app.AnliseWaveletTab);
app.Panel_2.BackgroundColor = [0.902 0.902 0.902];
app.Panel_2.Position = [1050 0 335 617];

% Create ConfiguraesGrfico1Panel
app.ConfiguraesGrfico1Panel = uipanel(app.Panel_2);
app.ConfiguraesGrfico1Panel.TitlePosition = 'centertop';
app.ConfiguraesGrfico1Panel.Title = 'Configurações Gráfico 1';
app.ConfiguraesGrfico1Panel.BackgroundColor = [0.502 0.502 0.502];
app.ConfiguraesGrfico1Panel.FontWeight = 'bold';
app.ConfiguraesGrfico1Panel.FontSize = 16;
app.ConfiguraesGrfico1Panel.Position = [20 346 300 267];

% Create ContedoaserExibidoButtonGroup
app.ContedoaserExibidoButtonGroup = uibuttongroup(app.ConfiguraesGrfico1Panel);
app.ContedoaserExibidoButtonGroup.TitlePosition = 'centertop';
app.ContedoaserExibidoButtonGroup.Title = 'Conteúdo a ser Exibido';
app.ContedoaserExibidoButtonGroup.BackgroundColor = [0.8 0.8 0.8];
app.ContedoaserExibidoButtonGroup.FontWeight = 'bold';
app.ContedoaserExibidoButtonGroup.FontSize = 14;
app.ContedoaserExibidoButtonGroup.Position = [20 145 263 91];

% Create AproximaoButton
```

```
app.AproximaoButton = uiradiobutton(app.ContedoaserExibidoButtonGroup);
app.AproximaoButton.Text = 'Aproximação';
app.AproximaoButton.FontSize = 14;
app.AproximaoButton.FontWeight = 'bold';
app.AproximaoButton.FontColor = [0 0 1];
app.AproximaoButton.Position = [23 41 111 22];
app.AproximaoButton.Value = true;

% Create DetalheButton
app.DetalheButton = uiradiobutton(app.ContedoaserExibidoButtonGroup);
app.DetalheButton.Text = 'Detalhe';
app.DetalheButton.FontSize = 14;
app.DetalheButton.FontWeight = 'bold';
app.DetalheButton.FontColor = [0 0 1];
app.DetalheButton.Position = [151 41 72 22];

% Create opesDropDownLabel
app.opesDropDownLabel = uilabel(app.ContedoaserExibidoButtonGroup);
app.opesDropDownLabel.HorizontalAlignment = 'right';
app.opesDropDownLabel.FontWeight = 'bold';
app.opesDropDownLabel.Position = [53 13 47 22];
app.opesDropDownLabel.Text = 'opções';

% Create opesDropDown
app.opesDropDown = uidropdown(app.ContedoaserExibidoButtonGroup);
app.opesDropDown.Items = {};
app.opesDropDown.FontWeight = 'bold';
app.opesDropDown.FontColor = [0.6392 0.0784 0.1804];
app.opesDropDown.Position = [107 13 100 22];
app.opesDropDown.Value = {};

% Create VisualizaodosEixosButtonGroup
app.VisualizaodosEixosButtonGroup = uibuttongroup(app.ConfiguraesGrfico1Panel);
app.VisualizaodosEixosButtonGroup.TitlePosition = 'centertop';
app.VisualizaodosEixosButtonGroup.Title = 'Visualização dos Eixos';
app.VisualizaodosEixosButtonGroup.BackgroundColor = [0.8 0.8 0.8];
app.VisualizaodosEixosButtonGroup.FontWeight = 'bold';
app.VisualizaodosEixosButtonGroup.FontSize = 14;
app.VisualizaodosEixosButtonGroup.Position = [20 15 263 120];

% Create AutomticoButton_3
app.AutomticoButton_3 = uiradiobutton(app.VisualizaodosEixosButtonGroup);
app.AutomticoButton_3.Text = 'Automático';
app.AutomticoButton_3.FontSize = 14;
app.AutomticoButton_3.FontWeight = 'bold';
app.AutomticoButton_3.FontColor = [0 0 1];
app.AutomticoButton_3.Position = [23 70 99 22];
app.AutomticoButton_3.Value = true;

% Create ManualButton_3
app.ManualButton_3 = uiradiobutton(app.VisualizaodosEixosButtonGroup);
app.ManualButton_3.Text = 'Manual';
app.ManualButton_3.FontSize = 14;
app.ManualButton_3.FontWeight = 'bold';
app.ManualButton_3.FontColor = [0 0 1];
app.ManualButton_3.Position = [151 70 71 22];

% Create XEditField_4
app.XEditField_4 = uieditfield(app.VisualizaodosEixosButtonGroup, 'numeric');
app.XEditField_4.FontWeight = 'bold';
app.XEditField_4.FontColor = [0 0 1];
app.XEditField_4.Position = [148 38 62 22];
app.XEditField_4.Value = 1000;

% Create XEditField_5
app.XEditField_5 = uieditfield(app.VisualizaodosEixosButtonGroup, 'numeric');
app.XEditField_5.FontWeight = 'bold';
app.XEditField_5.FontColor = [0 0 1];
app.XEditField_5.Position = [148 9 62 22];
app.XEditField_5.Value = 1.5;

% Create XEditField_6Label
app.XEditField_6Label = uilabel(app.VisualizaodosEixosButtonGroup);
app.XEditField_6Label.HorizontalAlignment = 'center';
app.XEditField_6Label.FontWeight = 'bold';
app.XEditField_6Label.Position = [40 38 25 22];
app.XEditField_6Label.Text = 'X';

% Create XEditField_6
app.XEditField_6 = uieditfield(app.VisualizaodosEixosButtonGroup, 'numeric');
app.XEditField_6.FontWeight = 'bold';
app.XEditField_6.FontColor = [0 0 1];
app.XEditField_6.Position = [72 38 62 22];

% Create YEditField_2Label
app.YEditField_2Label = uilabel(app.VisualizaodosEixosButtonGroup);
app.YEditField_2Label.HorizontalAlignment = 'center';
app.YEditField_2Label.FontWeight = 'bold';
app.YEditField_2Label.Position = [40 9 25 22];
app.YEditField_2Label.Text = 'Y';

% Create YEditField_2
app.YEditField_2 = uieditfield(app.VisualizaodosEixosButtonGroup, 'numeric');
app.YEditField_2.FontWeight = 'bold';
app.YEditField_2.FontColor = [0 0 1];
app.YEditField_2.Position = [72 9 62 22];
app.YEditField_2.Value = -1.5;

% Create ConfiguraesGrfico2Panel
app.ConfiguraesGrfico2Panel = uipanel(app.Panel_2);
app.ConfiguraesGrfico2Panel.TitlePosition = 'centertop';
app.ConfiguraesGrfico2Panel.Title = 'Configurações Gráfico 2';
app.ConfiguraesGrfico2Panel.BackgroundColor = [0.502 0.502 0.502];
app.ConfiguraesGrfico2Panel.FontWeight = 'bold';
```

```
app.ConfiguraesGrfico2Panel.FontSize = 16;
app.ConfiguraesGrfico2Panel.Position = [20 2 300 340];

% Create VisualizaodosEixosButtonGroup_2
app.VisualizaodosEixosButtonGroup_2 = uibuttongroup(app.ConfiguraesGrfico2Panel);
app.VisualizaodosEixosButtonGroup_2.TitlePosition = 'centertop';
app.VisualizaodosEixosButtonGroup_2.Title = 'Visualização dos Eixos';
app.VisualizaodosEixosButtonGroup_2.BackgroundColor = [0.8 0.8 0.8];
app.VisualizaodosEixosButtonGroup_2.FontWeight = 'bold';
app.VisualizaodosEixosButtonGroup_2.FontSize = 14;
app.VisualizaodosEixosButtonGroup_2.Position = [12 9 279 130];

% Create AutomticoButton_4
app.AutomticoButton_4 = uiradiobutton(app.VisualizaodosEixosButtonGroup_2);
app.AutomticoButton_4.Text = 'Automático';
app.AutomticoButton_4.FontSize = 14;
app.AutomticoButton_4.FontWeight = 'bold';
app.AutomticoButton_4.FontColor = [0 0 1];
app.AutomticoButton_4.Position = [23 74 99 22];
app.AutomticoButton_4.Value = true;

% Create ManualButton_4
app.ManualButton_4 = uiradiobutton(app.VisualizaodosEixosButtonGroup_2);
app.ManualButton_4.Text = 'Manual';
app.ManualButton_4.FontSize = 14;
app.ManualButton_4.FontWeight = 'bold';
app.ManualButton_4.FontColor = [0 0 1];
app.ManualButton_4.Position = [151 74 71 22];

% Create XEditField_7
app.XEditField_7 = uieditfield(app.VisualizaodosEixosButtonGroup_2, 'numeric');
app.XEditField_7.FontWeight = 'bold';
app.XEditField_7.FontColor = [0 0 1];
app.XEditField_7.Position = [148 41 62 22];
app.XEditField_7.Value = 1000;

% Create XEditField_8
app.XEditField_8 = uieditfield(app.VisualizaodosEixosButtonGroup_2, 'numeric');
app.XEditField_8.FontWeight = 'bold';
app.XEditField_8.FontColor = [0 0 1];
app.XEditField_8.Position = [148 9 62 22];
app.XEditField_8.Value = 1.5;

% Create XEditField_9Label
app.XEditField_9Label = uilabel(app.VisualizaodosEixosButtonGroup_2);
app.XEditField_9Label.HorizontalAlignment = 'center';
app.XEditField_9Label.FontWeight = 'bold';
app.XEditField_9Label.Position = [40 41 25 22];
app.XEditField_9Label.Text = 'X';

% Create XEditField_9
app.XEditField_9 = uieditfield(app.VisualizaodosEixosButtonGroup_2, 'numeric');
app.XEditField_9.FontWeight = 'bold';
app.XEditField_9.FontColor = [0 0 1];
app.XEditField_9.Position = [72 41 62 22];

% Create YEditField_3Label
app.YEditField_3Label = uilabel(app.VisualizaodosEixosButtonGroup_2);
app.YEditField_3Label.HorizontalAlignment = 'center';
app.YEditField_3Label.FontWeight = 'bold';
app.YEditField_3Label.Position = [40 9 25 22];
app.YEditField_3Label.Text = 'Y';

% Create YEditField_3
app.YEditField_3 = uieditfield(app.VisualizaodosEixosButtonGroup_2, 'numeric');
app.YEditField_3.FontWeight = 'bold';
app.YEditField_3.FontColor = [0 0 1];
app.YEditField_3.Position = [72 9 62 22];
app.YEditField_3.Value = -1.5;

% Create ContedoaserExibidoPanel
app.ContedoaserExibidoPanel = uipanel(app.ConfiguraesGrfico2Panel);
app.ContedoaserExibidoPanel.TitlePosition = 'centertop';
app.ContedoaserExibidoPanel.Title = 'Conteúdo a ser Exibido';
app.ContedoaserExibidoPanel.BackgroundColor = [0.8 0.8 0.8];
app.ContedoaserExibidoPanel.FontWeight = 'bold';
app.ContedoaserExibidoPanel.FontSize = 14;
app.ContedoaserExibidoPanel.Position = [12 187 279 122];

% Create UITable
app.UITable = uitable(app.ContedoaserExibidoPanel);
app.UITable.ColumnName = {'cAr'; 'Adicionar'; 'cDr'; 'Adicionar'};
app.UITable.RowName = {};
app.UITable.ColumnEditable = [false true false true];
app.UITable.ForegroundColor = [0 0 1];
app.UITable.FontWeight = 'bold';
app.UITable.Position = [0 -1 279 97];

% Create ButtonGroup
app.ButtonGroup = uibuttongroup(app.ConfiguraesGrfico2Panel);
app.ButtonGroup.TitlePosition = 'centertop';
app.ButtonGroup.BackgroundColor = [0.502 0.502 0.502];
app.ButtonGroup.FontWeight = 'bold';
app.ButtonGroup.Position = [53 144 201 35];

% Create SomaButton
app.SomaButton = uiradiobutton(app.ButtonGroup);
app.SomaButton.Text = 'Soma';
app.SomaButton.FontSize = 14;
app.SomaButton.FontWeight = 'bold';
app.SomaButton.FontColor = [0.6392 0.0784 0.1804];
app.SomaButton.Position = [11 8 60 22];
app.SomaButton.Value = true;
```

```
% Create SubtraoButton
app.SubtraoButton = uiradiobutton(app.ButtonGroup);
app.SubtraoButton.Text = 'Subtração';
app.SubtraoButton.FontSize = 14;
app.SubtraoButton.FontWeight = 'bold';
app.SubtraoButton.FontColor = [0.6392 0.0784 0.1804];
app.SubtraoButton.Position = [103 6 91 22];

% Create Panel_3
app.Panel_3 = uipanel(app.AnliseWaveletTab);
app.Panel_3.BackgroundColor = [0.902 0.902 0.902];
app.Panel_3.Position = [1 616 1384 82];

% Create text18_2
app.text18_2 = uilabel(app.Panel_3);
app.text18_2.BackgroundColor = [0 0.450980392156863 0.741176470588235];
app.text18_2.HorizontalAlignment = 'center';
app.text18_2.VerticalAlignment = 'top';
app.text18_2.FontSize = 22;
app.text18_2.FontWeight = 'bold';
app.text18_2.FontColor = [1 1 1];
app.text18_2.Position = [330 43 720 30];
app.text18_2.Text = 'Programa de Pós-Graduação em Energia e Ambiente(PGEA)';

% Create text19_2
app.text19_2 = uilabel(app.Panel_3);
app.text19_2.BackgroundColor = [1 1 0];
app.text19_2.HorizontalAlignment = 'center';
app.text19_2.VerticalAlignment = 'top';
app.text19_2.FontSize = 16;
app.text19_2.FontWeight = 'bold';
app.text19_2.Position = [360 12 645 31];
app.text19_2.Text = 'Universidade da Integração Internacional da Lusofonia Afro-Brasileira(UNILAB) ';

% Create BrowseButton
app.BrowseButton = uibutton(app.AnliseWaveletTab, 'push');
app.BrowseButton.ButtonPushedFcn = createCallbackFcn(app, @BrowseButtonPushed, true);
app.BrowseButton.BackgroundColor = [0 1 1];
app.BrowseButton.FontSize = 14;
app.BrowseButton.FontWeight = 'bold';
app.BrowseButton.Position = [877 580 100 25];
app.BrowseButton.Text = 'Browse';

% Create DiretriodoarquivoEditField
app.DiretriodoarquivoEditField = uieditfield(app.AnliseWaveletTab, 'text');
app.DiretriodoarquivoEditField.Editable = 'off';
app.DiretriodoarquivoEditField.Position = [436 581 425 24];

% Create SelecioneoarquivomatcomosinalLabel
app.SelecioneoarquivomatcomosinalLabel = uilabel(app.AnliseWaveletTab);
app.SelecioneoarquivomatcomosinalLabel.FontSize = 14;
app.SelecioneoarquivomatcomosinalLabel.FontWeight = 'bold';
app.SelecioneoarquivomatcomosinalLabel.FontColor = [0.651 0.651 0.651];
app.SelecioneoarquivomatcomosinalLabel.Position = [497 581 267 22];
app.SelecioneoarquivomatcomosinalLabel.Text = 'Selecione o arquivo ".mat" com o sinal';

% Create IniciarButton_2
app.IniciarButton_2 = uibutton(app.AnliseWaveletTab, 'push');
app.IniciarButton_2.ButtonPushedFcn = createCallbackFcn(app, @IniciarButton_2Pushed, true);
app.IniciarButton_2.BackgroundColor = [0 1 0];
app.IniciarButton_2.FontSize = 14;
app.IniciarButton_2.FontWeight = 'bold';
app.IniciarButton_2.Position = [450 32 100 26];
app.IniciarButton_2.Text = 'Iniciar';

% Create StopButton_2
app.StopButton_2 = uibutton(app.AnliseWaveletTab, 'state');
app.StopButton_2.ValueChangedFcn = createCallbackFcn(app, @StopButton_2ValueChanged, true);
app.StopButton_2.Text = 'Stop';
app.StopButton_2.BackgroundColor = [1 0 0];
app.StopButton_2.FontSize = 14;
app.StopButton_2.FontWeight = 'bold';
app.StopButton_2.Position = [567 32 100 26];

% Create NomedaVarivelEditFieldLabel
app.NomedaVarivelEditFieldLabel = uilabel(app.AnliseWaveletTab);
app.NomedaVarivelEditFieldLabel.HorizontalAlignment = 'right';
app.NomedaVarivelEditFieldLabel.FontSize = 14;
app.NomedaVarivelEditFieldLabel.FontWeight = 'bold';
app.NomedaVarivelEditFieldLabel.Position = [439 552 121 22];
app.NomedaVarivelEditFieldLabel.Text = 'Nome da Variável';

% Create NomedaVarivelEditField
app.NomedaVarivelEditField = uieditfield(app.AnliseWaveletTab, 'text');
app.NomedaVarivelEditField.FontSize = 14;
app.NomedaVarivelEditField.FontWeight = 'bold';
app.NomedaVarivelEditField.FontColor = [0 0 1];
app.NomedaVarivelEditField.Position = [575 552 100 22];
app.NomedaVarivelEditField.Value = 'ECGch_3';

% Create SinalOriginalCheckBox
app.SinalOriginalCheckBox = uicheckbox(app.AnliseWaveletTab);
app.SinalOriginalCheckBox.Text = 'Sinal Original';
app.SinalOriginalCheckBox.FontWeight = 'bold';
app.SinalOriginalCheckBox.FontColor = [0.6392 0.0784 0.1804];
app.SinalOriginalCheckBox.Position = [932 274 99 22];

% Create TamanhoEditFieldLabel
app.TamanhoEditFieldLabel = uilabel(app.AnliseWaveletTab);
app.TamanhoEditFieldLabel.HorizontalAlignment = 'right';
app.TamanhoEditFieldLabel.FontSize = 14;
app.TamanhoEditFieldLabel.FontWeight = 'bold';
app.TamanhoEditFieldLabel.Position = [759 552 67 22];
app.TamanhoEditFieldLabel.Text = 'Tamanho';
```

```
        % Create TamanhoEditField
        app.TamanhoEditField = uieditfield(app.AnliseWaveletTab, 'numeric');
        app.TamanhoEditField.Position = [841 552 100 22];

        % Create Grfico1CheckBox
        app.Grfico1CheckBox = uicheckbox(app.AnliseWaveletTab);
        app.Grfico1CheckBox.Text = 'Gráfico 1';
        app.Grfico1CheckBox.FontWeight = 'bold';
        app.Grfico1CheckBox.FontColor = [0.6392 0.0784 0.1804];
        app.Grfico1CheckBox.Position = [841 274 74 22];

        % Create DetecodeRudosButtonGroup
        app.DetecodeRudosButtonGroup = uibuttongroup(app.AnliseWaveletTab);
        app.DetecodeRudosButtonGroup.TitlePosition = 'centertop';
        app.DetecodeRudosButtonGroup.Title = 'Detecção de Ruídos';
        app.DetecodeRudosButtonGroup.BackgroundColor = [0.8 0.8 0.8];
        app.DetecodeRudosButtonGroup.FontWeight = 'bold';
        app.DetecodeRudosButtonGroup.FontSize = 16;
        app.DetecodeRudosButtonGroup.Position = [382 290 168 246];

        % Create FLouTVNButton
        app.FLouTVNButton = uiradiobutton(app.DetecodeRudosButtonGroup);
        app.FLouTVNButton.Text = 'FL ou TVN';
        app.FLouTVNButton.FontSize = 14;
        app.FLouTVNButton.FontWeight = 'bold';
        app.FLouTVNButton.FontColor = [0 0 1];
        app.FLouTVNButton.Position = [11 195 92 22];
        app.FLouTVNButton.Value = true;

        % Create BWButton
        app.BWButton = uiradiobutton(app.DetecodeRudosButtonGroup);
        app.BWButton.Text = 'BW';
        app.BWButton.FontSize = 14;
        app.BWButton.FontWeight = 'bold';
        app.BWButton.FontColor = [0 0 1];
        app.BWButton.Position = [12 164 45 22];

        % Create ABButton
        app.ABButton = uiradiobutton(app.DetecodeRudosButtonGroup);
        app.ABButton.Text = 'AB';
        app.ABButton.FontSize = 14;
        app.ABButton.FontWeight = 'bold';
        app.ABButton.FontColor = [0 0 1];
        app.ABButton.Position = [13 136 42 22];

        % Create MAPLIAWGNButton
        app.MAPLIAWGNButton = uiradiobutton(app.DetecodeRudosButtonGroup);
        app.MAPLIAWGNButton.Text = 'MA, PLI, AWGN ';
        app.MAPLIAWGNButton.FontSize = 14;
        app.MAPLIAWGNButton.FontWeight = 'bold';
        app.MAPLIAWGNButton.FontColor = [0 0 1];
        app.MAPLIAWGNButton.Position = [13 111 128 22];

        % Create IntervaloButtonGroup
        app.IntervaloButtonGroup = uibuttongroup(app.DetecodeRudosButtonGroup);
        app.IntervaloButtonGroup.TitlePosition = 'centertop';
        app.IntervaloButtonGroup.Title = 'Intervalo';
        app.IntervaloButtonGroup.BackgroundColor = [0 1 1];
        app.IntervaloButtonGroup.FontWeight = 'bold';
        app.IntervaloButtonGroup.FontSize = 14;
        app.IntervaloButtonGroup.Position = [1 42 158 63];

        % Create XEditField_12
        app.XEditField_12 = uieditfield(app.IntervaloButtonGroup, 'numeric');
        app.XEditField_12.FontWeight = 'bold';
        app.XEditField_12.FontColor = [0 0 1];
        app.XEditField_12.Position = [3 13 62 22];

        % Create XEditField_10
        app.XEditField_10 = uieditfield(app.IntervaloButtonGroup, 'numeric');
        app.XEditField_10.FontWeight = 'bold';
        app.XEditField_10.FontColor = [0 0 1];
        app.XEditField_10.Position = [78 14 62 22];
        app.XEditField_10.Value = 1000;

        % Create IniciarButton_3
        app.IniciarButton_3 = uibutton(app.DetecodeRudosButtonGroup, 'push');
        app.IniciarButton_3.ButtonPushedFcn = createCallbackFcn(app, @AtualizarButtonValueChanged, true);
        app.IniciarButton_3.BackgroundColor = [0 0 1];
        app.IniciarButton_3.FontSize = 14;
        app.IniciarButton_3.FontWeight = 'bold';
        app.IniciarButton_3.Position = [27 1 100 26];
        app.IniciarButton_3.Text = 'Iniciar';

        % Create Tab
        app.Tab = uitab(app.TabGroup);
        app.Tab.Title = 'Tab';

        % Show the figure after all components are created
        app.SinalECGUIFigure.Visible = 'on';
    end
end

% App creation and deletion
methods (Access = public)

    % Construct app
    function app = app1_waveletMATLAB

        % Create UIFigure and components
        createComponents(app)

        % Register the app with App Designer
```

```
            registerApp(app, app.SinalECGUIFigure)

            if nargout == 0
                clear app
            end
        end

        % Code that executes before app deletion
        function delete(app)

            % Delete UIFigure when app is deleted
            delete(app.SinalECGUIFigure)
        end
    end
end
```