INSTITUTO FEDERAL SUL-RIO-GRANDENSE - IFSUL
CAMPUS CHARQUEADAS
ENGENHARIA DE CONTROLE E AUTOMAÇÃO

ACSSA PASSOS SOUSA

EDGEECG - COMPRESSÃO DE ECG DE LONGO PRAZO APLICADO EM DISPOSITIVO IOT INTEROPERÁVEL

Orientador: Prof. Dr. André Luís del Mestre Martins

CHARQUEADAS, 2025

ACSSA PASSOS SOUSA

EdgeECG - Compressão de ECG de longo prazo aplicado em dispositivo IoT interoperável

Monografia apresentada como requisito parcial para obtenção do título de Engenheiro de Controle e Automação.

Orientador: Prof. Dr. André Luís del Mestre Martins

CHARQUEADAS, **2025**

ACSSA PASSOS SOUSA

EDGEECG - COMPRESSÃO DE ECG DE LONGO PRAZO APLICADO EM DISPOSITIVO IOT INTEROPERÁVEL

Este trabalho foi julgado adequado para a obtenção do título de Engenheiro de Controle e Automação e aprovada em sua forma final pelo Orientador e pela Banca Examinadora.

Orientador: _____
Prof. Dr. André Luís del Mestre Martins

Banca Examinadora:

Prof. Dr. Juliano Costa Machado, IFSul Câmpus Charqueadas
Doutor em Engenharia Elétrica (UFRGS-2022)

Prof. Dr. Fábio Pires Iturriet, Departamento de Eletrotécnica - UTFPR Câmpus Curitiba
Doutor em Engenharia Elétrica (UFSC-2019)

Coordenador do Curso: _____
Prof. Carlos Arthur Carvalho Sarmanho Júnior

Charqueadas, 2025.

CIP – CATALOGAÇÃO NA PUBLICAÇÃO

S587s Sousa, Acssa Passos

EdgeECG - Compressão de ECG de longo prazo aplicado em dispositivo IoT interoperável / Acssa Passos Sousa. – 2025.

74 f. : il.;color.

Monografia (Engenharia de Controle e Automação) – Instituto Federal Sul-rio-grandense - IFSul– Campus Charqueadas Charqueadas, RS, 2025.

"Orientador: André Luís del Mestre Martins."

1. ECG. 2. MCU. 3. Wavelets. 4. Compressão. 5. IoT. I. Título. del Mestre Martins, André Luís. II. Título.

CDU 004.43

RESUMO

De acordo com o Ministério da Saúde, as doenças cardiovasculares são as principais causas de morte no Brasil. Estima-se que, até 2040, haverá um aumento de até 250% nesses eventos no país. O eletrocardiograma (ECG), um dos sinais fisiológicos mais utilizados e bem-sucedidos na engenharia biomédica, é o foco deste estudo. Nesse contexto, o IFSul, campus Charqueadas, desenvolveu um eletrocardiógrafo portátil para transmitir ECGs de longa duração pela internet.

No mesmo âmbito do projeto, outros trabalhos incluem: (i) o dispositivo físico para aquisição de sinais de ECG, (ii) o EdgeECG, um sistema de compressão embarcado para exibição local e envio de dados para a internet, e (iii) o FASS-ECG, um servidor na nuvem para armazenar e processar esses dados. Dois desafios críticos no desenvolvimento do EdgeECG são a compressão eficiente dos dados para envio e a redução significativa no armazenamento.

As técnicas de compressão de dados são geralmente classificadas em dois tipos: compressão sem perdas (LLDC) e compressão com perdas (LDC). Este trabalho adota o método de transformação de dados, amplamente utilizado por sua eficiência em taxa de compressão e qualidade de reconstrução do sinal. Nesse método, os sinais são transformados do domínio do tempo para frequência, seguidos por processos de quantização e limiarização, configurando uma compressão do tipo LDC.

No projeto, a compressão dos sinais de ECG é realizada por meio da Transformada Wavelet (WT), que decompõe o sinal em múltiplas bandas de frequência de forma hierárquica. O objetivo principal é implementar o EdgeECG, permitindo a transmissão de ECGs e dados básicos de pacientes para nuvens compatíveis com o padrão FHIR, integrando um sistema de compressão para envio eficiente e armazenamento otimizado. O projeto envolve a interação de um microcontrolador (MCU) acoplado a um módulo Wi-Fi ESP8266 com um servidor online para armazenamento de ECGs. Paralelamente, foi desenvolvido um sistema de compressão de dados utilizando bancos de filtros wavelet e codificação de entropia pelo algoritmo de Huffman. O desenvolvimento da programação do MCU foi dividido em quatro etapas principais: configuração do módulo ESP, transformação dos dados para o domínio da frequência com WT, implementação da codificação Huffman e transmissão dos dados via requisições HTTP.

Para os processos de quantização e limiarização, os experimentos foram conduzidos em MATLAB, enquanto o principal desafio foi a implementação em C dos algoritmos desenvolvidos inicialmente em MATLAB e Python. Toda a programação do MCU em C, juntamente com o esquema de conexão do ESP e as especificações técnicas do MCU, está disponível em um repositório público e gratuito na internet.

Palavras-chave: ECG. MCU. Wavelets. Compressão. IoT.

ABSTRACT

According to the Ministry of Health, cardiovascular diseases are the leading causes of death in Brazil. It is estimated that by 2040, there will be up to a 250% increase in these events in the country. The electrocardiogram (ECG), one of the most widely used and successful physiological signals in biomedical engineering, is the focus of this study. In this context, IFSul, Charqueadas campus, has developed a portable electrocardiograph to transmit long-duration ECGs over the internet.

Within the same project scope, additional initiatives include: (i) a physical device for ECG signal acquisition, (ii) EdgeECG, an embedded system for local display and internet data transmission, and (iii) FASS-ECG, a cloud server for storing and processing ECG data. Two critical challenges in the development of EdgeECG are efficient data compression for transmission and significant reduction in data storage.

Data compression techniques are generally classified into two types: lossless compression (LLDC) and lossy compression (LDC). This study adopts the data transformation method, widely recognized for its efficiency in achieving high compression ratios and signal reconstruction quality. In this method, signals are transformed from the time domain to the frequency domain, followed by quantization and thresholding processes, resulting in LDC-type compression.

In this project, ECG signal compression is performed using the Wavelet Transform (WT), which decomposes the signal into multiple frequency bands in a hierarchical process. The main objective is to implement EdgeECG, enabling the transmission of ECGs and basic patient data to web clouds compatible with the FHIR standard, while incorporating a compression system to optimize transmission and storage. The project integrates a microcontroller (MCU) connected to an ESP8266 Wi-Fi module and an online server for ECG storage. Simultaneously, a data compression system was developed using wavelet filter banks and entropy coding via the Huffman algorithm. The MCU programming development was divided into four main stages: configuring the ESP module, transforming the data to the frequency domain using WT, implementing Huffman coding, and transmitting data via HTTP web requests.

For the quantization and thresholding processes, experiments were conducted in MATLAB, while the primary challenge was implementing in C the algorithms initially developed in MATLAB and Python. All MCU programming in C, the wiring diagram for connecting the ESP to the MCU, and the technical specifications of the MCU are available in a public and free repository online.

Keywords: ECG. MCU. Wavelets. Compression. IoT.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE CÓDIGOS-FONTE

LISTA DE ABREVIATURAS, SIGLAS E SÍMBOLOS

ABNT	Associação Brasileira de Normas Técnicas
IFSul	Instituto Federal Sul-rio-grandense
ECG	Eletrocardiograma
DCV	Doenças cardiovasculare
OMS	Organização Mundial da Saúde
IoT	Internet das Coisas
FHIR	o Fast Health Interoperability Resources
HL7	Health Level Seven
RNDS	Rede Nacional de Dados em Saúde
WT	Transformada Wavelet
PAS	Pressão arterial sistólica
PAD	Pressão arterial diastólica
FIR	Resposta ao Impulso Finita
CR	Taxa de compressão
LLDC	Lossless Data Compression
LDC	Lossy Data Compression
TP	Turning Pointing
FFT	Transformada Rápida de Fourier
DCT	Transformada Cosseno Discreta
LSPIHT	Particionamento de conjuntos em árvores hierárquicas
DWT	Transformada Wavelet Discreta
CWT	Transformada Wavelet Contínua
DFT	Transformada Discreta de Fourier
WP	Wavelet Packet
RLE	Run-length encoding
MIT)	Instituto de Tecnologia de Massachusetts

RES Registros Eletrônicos de Saúde

HOS Estatística de alta ordem

PRD Percentagem de Diferença Quadrática Média

CR Taxa de Compressão

QS Qualidade de Sinal

RL Run-Length

MCU Microcontrolador

LPF Passa-baixas

HPF Passa-altas

ILPF Passa-baixas inverso

IHPF Passa-altas inverso

PRD Percentual de Dife- rença Quadrática Média

IDE Ambiente de desenvolvimento integrad

RMSE Erro Quadrático Médio , do inglês *Root Mean Square Error*

SUMÁRIO

1 INTRODUÇÃO

A evolução tecnológica na área da saúde tem promovido avanços significativos na forma como dados biomédicos são processados e integrados em sistemas de monitoramento e diagnóstico. Nesse contexto, o eletrocardiograma (ECG), essencial para monitorar a atividade elétrica do coração, destaca-se como uma ferramenta central em diagnósticos médicos. Tradicionalmente, os sistemas de monitoramento de ECG eram restritos a equipamentos fixos e soluções proprietárias, enfrentando desafios como o armazenamento de grandes volumes de dados e a falta de interoperabilidade entre sistemas.

O monitoramento contínuo de ECG, especialmente para aplicações de longo prazo em dispositivos vestíveis ou monitoramento remoto, gera uma quantidade expressiva de dados que demandam processamento e armazenamento eficientes (Kaplan Berkaya et al., 2018). Este fator é visível ao se verificar o espaço necessário para armazenamento de um exame, de acordo com seu tempo de realização como ilustrado ena Figura 1.1. Isso é particularmente importante diante da crescente prevalência de doenças cardiovasculares (DCV), que, de acordo com a Organização Mundial da Saúde (OMS), são responsáveis por 17,9 milhões de mortes anuais, representando 32% das mortes globais. A doença cardíaca isquêmica, por exemplo, responde por 13% dessas fatalidades em 2021. Esses números ressaltam a necessidade de soluções tecnológicas capazes de fornecer diagnósticos mais rápidos e precisos, além de otimizar os custos associados ao tratamento de doenças crônicas.

Apesar dos avanços significativos na área, ainda existem desafios técnicos e científicos a serem superados. Entre eles, destacam-se a necessidade de algoritmos de compressão mais eficientes, capazes de lidar com as limitações de recursos computacionais de dispositivos IoT; a garantia de interoperabilidade plena entre sistemas; e a validação de métodos em cenários reais.

Nesse cenário, o desenvolvimento de Eletrocardiógrafos com conexão com a internet devem ter uma estratégia para reduzir os custos de armazenamento e transmissão de dados em dispositivos da Internet das Coisas (IoT).

A interoperabilidade entre sistemas de saúde é indispensável para integrar os dados em ecossistemas heterogêneos, e o padrão Fast Health Interoperability Resources (FHIR), desenvolvido pelo Health Level Seven (HL7), tem emergido como a solução mais robusta para o intercâmbio de informações médicas (AYAZ et al., 2021; SARIPALLE, 2019). No Brasil, iniciativas como a Rede Nacional de Dados em Saúde (RNDS) têm demonstrado o potencial do FHIR para conectar diferentes sistemas e promover um ecossistema de saúde mais eficiente (RNDS, 2022).

Este trabalho busca contribuir para a superação dessas lacunas, propondo uma abordagem que combina métodos de compressão com integração ao padrão FHIR, visando otimizar o uso de dispositivos IoT em aplicações de saúde.

Com isso, o objetivo principal é explorar a integração de métodos de compressão de si-

Figura 1.1: Tamanho dos dados de ECG para armazenamento de sinais sem EdgeECG

Duração	360 Hz	Derivações	Tamanho
24 horas	31.104.000 amostras	1	42,7MB
		12	513,2MB
1 hora	216.000 amostras	1	297 KB
		12	3,5 MB
5 min.	108.000 amostras	1	148,5 KB
		12	1,78 MB
10 s	3.600 amostras	1	4,95 KB
		12	59,4 KB

Fonte: Autoral

nais ECG em dispositivos IoT interoperáveis na prática, em um protótipo chamado EdgeECG, utilizando o padrão FHIR como base para armazenamento e intercâmbio de dados. Para isso, propõe-se uma abordagem baseada na Transformada Wavelet Daubechies 2, combinada com técnica de codificação Huffman, para alcançar um equilíbrio entre compressão eficiente e preservação da qualidade do sinal. A principal contribuição está na implementação de um algoritmo em um dispositivo que realiza a transmissão de dados para armazenamento em nuvem, compatível com o padrão de comunicação FHIR, atendendo às exigências das aplicações de saúde digital.

A abordagem metodológica inclui: (i) análise teórica de métodos de compressão; (ii) implementação de algoritmos baseados na Transformada Wavelet Daubechies 2 e codificação Huffman; (iii) integração com o padrão FHIR e plataformas IoT; e (iv) validação experimental em cenários simulados. Os resultados serão avaliados com base em métricas como taxa de compressão e compatibilidade com plataformas baseadas no padrão FHIR.

A dissertação está organizada em cinco capítulos, além desta introdução. O capítulo 2 aborda os fundamentos teóricos do ECG, métodos de compressão e o padrão FHIR. O capítulo 3 apresenta uma revisão de trabalhos relacionados. No capítulo 4, a metodologia é detalhada, com foco nos algoritmos desenvolvidos. O capítulo 5 discute os resultados experimentais, destacando as contribuições e limitações do método proposto. Por fim, o capítulo 6 conclui o trabalho, indicando direções para pesquisas futuras.

2 REFERENCIAL TEORICO

Este capítulo explora as principais técnicas e tecnologias utilizadas no processamento e compressão de sinais biomédicos, com foco especial no ECG. O objetivo é apresentar os fundamentos teóricos e práticos necessários para aprimorar a qualidade, eficiência e representatividade desses sinais, especialmente em aplicações clínicas e remotas. A discussão inclui os conceitos essenciais no processamento de sinais, a transformação e a compressão de dados, bem como métodos avançados que garantem eficiência sem comprometer a qualidade do sinal.

Inicialmente, o capítulo introduz o ECG, destacando sua relevância no monitoramento da atividade elétrica do coração, além de discutir os desafios relacionados à captura e interpretação dos sinais (2.1). Em seguida, apresenta o conceito matemático das transformadas lineares, fundamentais para a análise de sinais no domínio da frequência, e sua aplicação no processamento de dados biomédicos (2.2). O texto segue abordando os métodos de compressão na seção 2.3 e, posteriormente, na seção 2.4, que apresenta as técnicas de compressão baseadas em transformadas. Nesta última, destaca-se a Transformada Wavelet (WT), amplamente utilizada devido à sua versatilidade e eficiência em aplicações que envolvem sinais de ECG, imagens e áudio. A WT permite uma redução significativa no tamanho dos dados, mantendo a integridade e a qualidade essenciais para as aplicações.

A Transformada Wavelet é explorada em detalhes na seção 2.5, essa técnica é particularmente útil para compressão de sinais biomédicos devido à sua representação compacta e eficiente. A limiarização, técnica discutida em 2.6, complementa o processo de compressão, descartando componentes com valores abaixo de um limiar predeterminado, o que contribui para a redução do ruído e compactação dos dados.

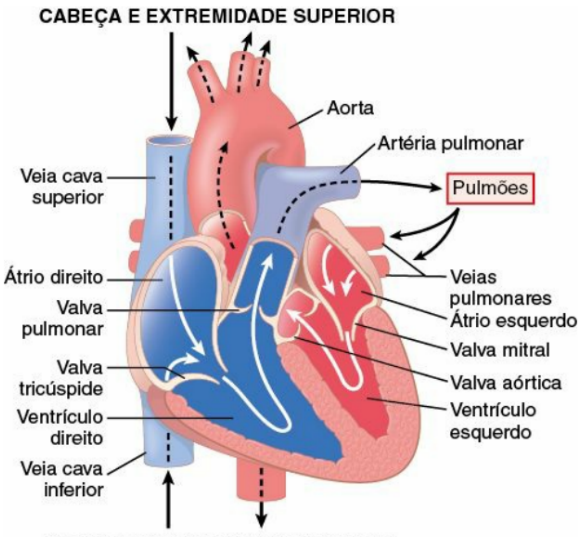
O capítulo também aborda a quantização (2.7), destacando os diferentes tipos disponíveis e como eles influenciam a qualidade do sinal comprimido e a eficiência geral do processo. Além disso, é introduzida a codificação Huffman e técnicas adaptativas em (2.7.1), que otimizam o armazenamento e a transmissão dos sinais de ECG, garantindo compressão eficiente sem comprometer a integridade do sinal. Por fim, a seção 2.8 apresenta o processo de envio e registro de dados biomédicos a partir do padrão internacional FHIR – Fast Health Interoperability Standards.

2.1 Eletrocardiograma

O coração é um órgão muscular responsável por bombear sangue para o corpo, garantindo o transporte de oxigênio e nutrientes.

Como ilustrado na Figura 2.1, ele é formado por duas bombas distintas: o coração direito, que bombeia o sangue para os pulmões, e o coração esquerdo, que bombeia o sangue através da circulação sistêmica que fornece o fluxo sanguíneo aos demais órgãos e tecidos do corpo. (GUYTON; HALL, 2017, pg.343).

Figura 2.1: Estrutura do coração e fluxo do sangue pelas câmaras e valvas cardíacas



Fonte: GUYTON; HALL (2017)

> O coração possui 4 câmaras, duas aurículas (ou átrios) e dois ventrículos. Sangue desoxigenado retorna ao átrio direito do coração, através das duas veias cava (superior e inferior). É bombeado para o ventrículo direito e depois para os pulmões onde o dióxido de carbono é libertado e o oxigénio é absorvido. O sangue oxigenado, em seguida, volta para o átrio esquerdo do coração, em seguida, para o ventrículo esquerdo a partir do qual é bombeado para dentro da aorta e a circulação arterial. (ANGOMED, 2024).

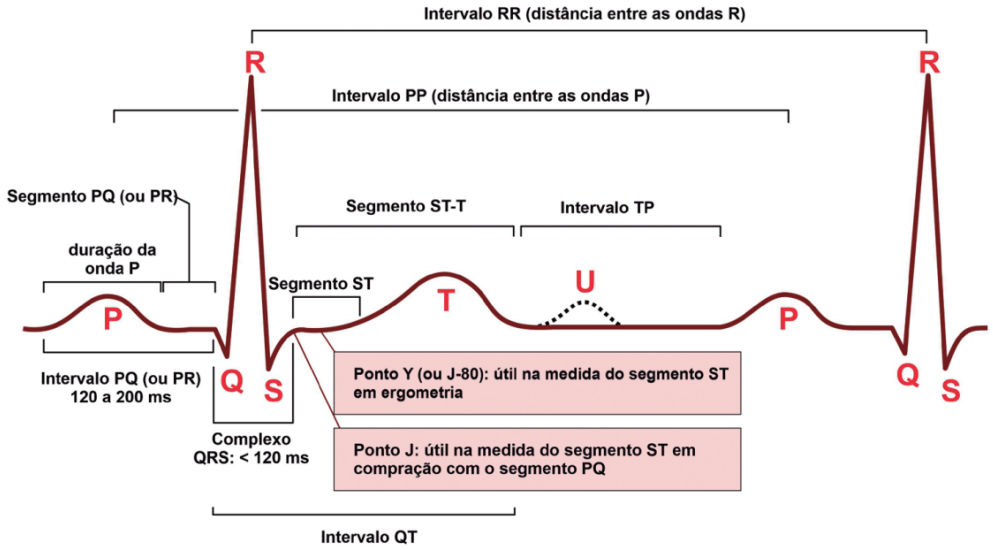
A função cardíaca é sustentada pelas propriedades de excitabilidade, condutibilidade e contractilidade das células cardíacas, essenciais para o ciclo cardíaco, composto por sístole (contração) e diástole (relaxamento). A pressão arterial sistólica (PAS) resulta da contração ventricular, enquanto a pressão arterial diastólica (PAD) ocorre no relaxamento, permitindo o enchimento ventricular ANGOMED (2024).

Na fibra ventricular do coração, o potencial de ação possui uma amplitude média de cerca de 105 milivolts. Durante cada batimento, o potencial intracelular varia de forma expressiva, iniciando com um valor negativo próximo a -85 milivolts, que ocorre entre os batimentos, e atingindo aproximadamente +20 milivolts em seu pico positivo. O ciclo cardíaco se caracteriza pela sequência de eventos que ocorre durante cada batimento do coração, envolvendo a contração (sístole) e o relaxamento (diástole) das câmaras cardíacas. Esse ciclo começa com a geração de um impulso elétrico no nodo sinusal, que se propaga pelos átrios, causando sua contração e depois, por meio do feixe A-V, para os ventrículos. Um ciclo completo, incluindo a sístole e a diástole define frequência cardíaca. Esse ciclo é regulado por mudanças de pressão nas câmaras cardíacas, que controlam a abertura e o fechamento das válvulas, garantindo o fluxo unidirecional de sangue. Essas etapas ocorrem de maneira sincronizada para manter uma circulação eficiente entre o coração, os pulmões e o restante do corpo (GUYTON; HALL, 2017).

O eletrocardiograma registra as voltagens geradas pelo coração na superfície do corpo, representadas pelas ondas P, Q, R, S e T (Figura 2.2). A onda P reflete a despolarização dos átrios, que antecede sua contração. O intervalo PR corresponde ao tempo necessário para o impulso elétrico atravessar os átrios e alcançar o nodo atrioventricular. O complexo QRS representa a despolarização dos ventrículos, sendo composto por: a onda Q, uma deflexão negativa inicial causada pela despolarização do septo interventricular; a onda R, uma deflexão positiva gerada pela propagação do impulso em direção ao ápice cardíaco; e a onda S, uma deflexão negativa final resultante da despolarização das paredes livres dos ventrículos. Esse processo desencadeia a contração ventricular e o aumento da pressão, antecedendo a sístole ventricular.

A repolarização dos ventrículos é representada pelo segmento ST e pela onda T, indicando o início do relaxamento ventricular. Por outro lado, a repolarização dos átrios ocorre simultaneamente ao complexo QRS, mas não é visível no ECG devido à sua baixa amplitude. Esses registros permitem uma análise detalhada do funcionamento cardíaco, auxiliando na identificação e avaliação do funcionamento cardíaco (SAúDE, 2024).

Figura 2.2: Ondas, complexos, intervalos e segmentos do eletrocardiograma de superfície



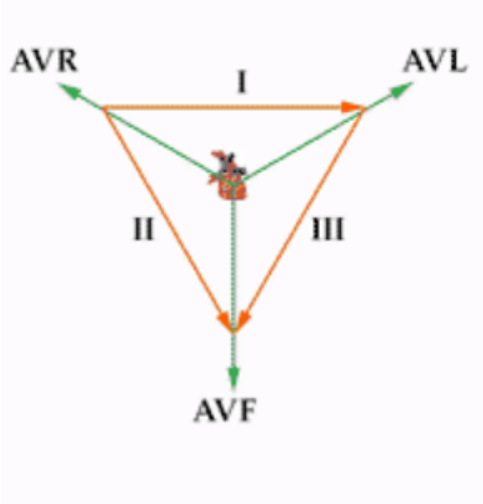
Fonte: Adaptado de SAúDE (2024)

A movimentação dos íons através das membranas celulares é o princípio de geração do sinal de ECG, a movimentação iônica induzem um potencial elétrico gerando campos detectáveis na superfície corporal o que origina o traçado do ECG. Os segmentos e intervalos de onda permitem identificar anormalidades no ritmo e na condução cardíaca registradas por meio da atividade elétrica do coração em um período de tempo. Este sinal é coletado por meio de eletrodos colocados sobre a pele, posicionados de acordo com padrões específicos que definem as derivações do ECG.

2.1.1 Derivações

As derivações são as atividades elétricas captadas de cada eletrodo, as várias posições dos eletrodos são usadas para captar diferentes ângulos do coração. O padrão clínico mais comum, conhecido como ECG de 12 derivações, fornece uma visão abrangente da atividade elétrica cardíaca a partir de eletrodos, ou seja, as derivações são 12 ângulos diferentes que acompanham simultaneamente a propagação da atividade elétrica. Elas são classificadas em unipolares e bipolares, com base no número de polos considerados durante a medição. Entre as derivações mais relevantes do ECG estão as precordiais, que são seis eletrodos fixados ao redor do tórax rotulados de V1 a V6, que fornecem informações detalhadas sobre a atividade elétrica dos ventrículos, além destes existem dois tipos de derivações periféricas: as derivações bipolares, ou de Einthoven, e as derivações unipolares aumentadas. Nas derivações V1 e V2, os registros do complexo QRS do coração normal são na maioria das vezes negativos, devido a posição do eletrodo dessas derivações está mais próximo da base cardíaca que permanece eletronegativa durante a maior parte do processo de despolarização ventricular. De modo oposto, nas derivações V4, V5 e V6, os complexos QRS são em sua maior parte positivos (My EKG, n.d.).

Figura 2.3: Triângulo de Einthoven



Fonte: Adaptado de My EKG (n.d.)

As derivações periféricas unipolares registram a diferença de potencial entre um ponto teórico no centro do triângulo de Einthoven (Figura 2.3), com um valor de 0, e os eletrodos em cada extremidade (My EKG, n.d.). Nesse tipo de registro, dois dos membros são conectados ao terminal negativo do eletrocardiógrafo por meio de resistências elétricas, e o terceiro membro é conectado ao terminal positivo. Quando o terminal positivo está no braço direito, a derivação é denominada aVR; quando está no braço esquerdo, aVL; e quando está na perna esquerda, aVF (GUYTON; HALL, 2017). Já as derivações bipolares registram a diferença de potencial entre dois eletrodos localizados em diferentes membros, sendo:

- D1: diferença de potencial entre o braço direito e o braço esquerdo. O vector é em

direcção de 0º

- D2: diferença de potencial entre o braço direito e a perna esquerda. O vector é em direcção de 60º.
- D3: diferença de potencial entre o braço esquerdo e a perna esquerda. O vector é em direcção de 120º.

.

2.2 Transformada Lineares em tempo discreto

Os sinais descrevem uma grande variedade de fenômenos físicos e podem ser representados de diferentes formas. A informação de um sinal está sempre contida em algum tipo de variação (OPPENHEIM; WILLSKY, 1997). No caso dos sinais de tempo contínuo, a variável independente é contínua e, portanto, esses sinais são definidos em um conjunto contínuo de valores. Por outro lado, os sinais de tempo discreto são definidos apenas em instantes específicos, geralmente como resultado do processo de amostragem (OPPENHEIM; WILLSKY, 1997). Como destacado em (MITRA, 2006), "sinais discretos são essenciais no contexto digital, especialmente quando representados por amostras para implementação computacional". Esses sinais possuem aplicações fundamentais em sistemas digitais, incluindo processamento de imagens, áudio e dados de comunicação.

A conexão entre sinais contínuos e discretos é mediada pela teoria da amostragem, formulada por Claude Shannon. Segundo SHANNON (1949), um sinal contínuo limitado em banda pode ser completamente reconstruído a partir de suas amostras, desde que seja amostrado a uma taxa pelo menos duas vezes maior que sua frequência máxima. Esse princípio fundamenta o uso de sinais discretos para representar sinais contínuos em sistemas digitais.

> Se uma função $f(t)$ não contém frequências superiores a W ciclos por segundo, ela é completamente determinada ao fornecer seus valores em uma série de pontos espaçados $1/(2W)$ segundos entre si SHANNON (1949).

A variável independente, denominada tempo, desempenha um papel essencial na análise e caracterização dos sinais. Quando a variável independente é contínua, o sinal resultante é classificado como de tempo contínuo. Se for discreta, é definido como de tempo discreto (OPPENHEIM; WILLSKY, 1997). Quando os instantes de definição de um sinal de tempo discreto são espaçados uniformemente, a variável independente discreta n pode ser normalizada para assumir valores inteiros. Isso é particularmente útil para representação digital, como no caso de uma função $x[n]$ definida por CAMPITELLI (2015):

$$x : \{0, 1, 2, \ldots, N-1\} \to \mathbb{R}. \tag{2.1}$$

Essa função pode ser representada de maneira mais eficiente em diversas aplicações ao ser expressa como uma combinação linear da forma:

$$x[n] = \sum_{k=0}^{N-1} \hat{X}[k] b_k[n], \tag{2.2}$$

onde k é um índice inteiro, $\hat{X}[k]$ são os coeficientes transformados e $b_k[n]$ representam as funções do conjunto de expansão. Segundo MALLAT (1999), "uma base é um conjunto de funções linearmente independentes que permite a representação única de qualquer função dentro de um espaço sinal".

Se o conjunto $\{b_k[n]\}$ permitir tal representação para todos os sinais $x[n]$, e se suas funções forem independentes, ele é denominado base. Quando o conjunto de expansão é ortogonal, ou seja, satisfaz:

$$\langle b_k[n], b_l[n] \rangle = \sum_{n=0}^{N-1} b_k[n] b_l^*[n] = 0, \quad \forall k \neq l, \tag{2.3}$$

os coeficientes $\hat{X}[k]$ podem ser calculados diretamente pelo produto interno:

$$\hat{X}[k] = \langle x[n], b_k[n] \rangle = \sum_{n=0}^{N-1} x[n] b_k^*[n]. \tag{2.4}$$

A relação entre sinais contínuos e discretos fundamenta o uso de transformadas, como a Transformada Wavelet, para decompor e reconstruir sinais com alta eficiência. No domínio digital, a convolução de sinais com filtros, como os filtros de Resposta ao Impulso Finita (FIR), desempenha um papel fundamental. A operação de convolução aplica os coeficientes do filtro ao sinal de entrada, produzindo um sinal de saída que ajusta seletivamente seu conteúdo em frequência. Como destacado por OPPENHEIM; WILLSKY (1997), filtros FIR são ideais para sistemas digitais devido à sua estabilidade, eficiência computacional e capacidade de separação de bandas.

2.3 Métodos de compressão de dados

Os principais motivos para aplicações de compressão de dados citados são a redução de requisitos de armazenamento e redução de custos de transmissão (CHANDRA; SHARMA; SINGH, 2021). No geral, a compressão de dados podem ser classificadas em dois tipos: O método LLDC (Lossless Data Compression) que refere-se a técnicas de compressão de dados sem perdas e LDC (Lossy Data Compression) refere-se a técnicas que permitem a compressão de dados com alta taxa de compressão (CR), mas com perda de informação. No LLDC os dados podem ser recuperados exatamente como os dados originais, mas em geral se é alcançada uma baixas CR. Huffman e run-length encoding (RLE) são os exemplos mais famosos de LLDC, no entanto as técnicas de LDC são mais preferidas para a compressão de dados de ritmos de ECG do que as técnicas de LLDC.

Segundo CHANDRA; SHARMA; SINGH (2021),

> os métodos de compressão de dados de ECG são classificados em três tipos: a) método direto de compressão de dados, b) métodos transformacionais de compressão de dados e c) método de compressão de dados baseado em extração de parâmetros.

De acordo com (BARBONI, 1992), nos métodos de extração de parâmetros, um conjunto de características é extraído do sinal e classificado com base em conhecimento prévio. Apesar de eficientes, esses métodos não preservam a forma do sinal e demandam maior processamento, limitando sua aplicação em certos casos. Um exemplo de aplicação desse método é a extração de parâmetros do complexo QRS, que utiliza algoritmos especializados para detectar o complexo QRS e extrair características essenciais, como os picos R, intervalos RR e outras métricas importantes para a análise do sinal de ECG.

Entre os métodos de manipulação direta de dados, destacam-se o TP (Turning Pointing) (MUELLER, 1978), AZETEC (COX; COLS, 1968, 1972), Predição Linear (RUTTIMANN; PIPBERGER, 1979), Codificação Delta (STEWART; COLS, 1973) e CORTES (ABENSTEIN; TOMPKINS, 1982). Sendo que o algoritmo TP analisa a tendência dos pontos amostrados, armazenando apenas um de cada dois pontos consecutivos, com isso sua implementação torna-se fácil, no entanto introduz distorções no sinal. O AZETEC utiliza a combinação da codificação por amplitude e por intervalo de tempo para compactar os dados, mas apresenta dificuldades em reproduzir fielmente o sinal original, especialmente nas ondas P e T. O CORTES, um sistema híbrido dos algoritmos TP e AZETEC, que por sua vez compartilha as desvantagens de ambos. Já a Codificação Delta, baseada na diferença entre amostras sucessivas, é sensível a ruídos e distorce as ondas P. E por fim a Predição Linear (RUTTIMANN; PIPBERGER, 1979), utiliza modelos matemáticos para prever os valores futuros de um sinal com base em seus valores passados. Em outras palavras, a predição linear tenta estimar o valor de uma amostra do sinal a partir de uma combinação linear das amostras anteriores.

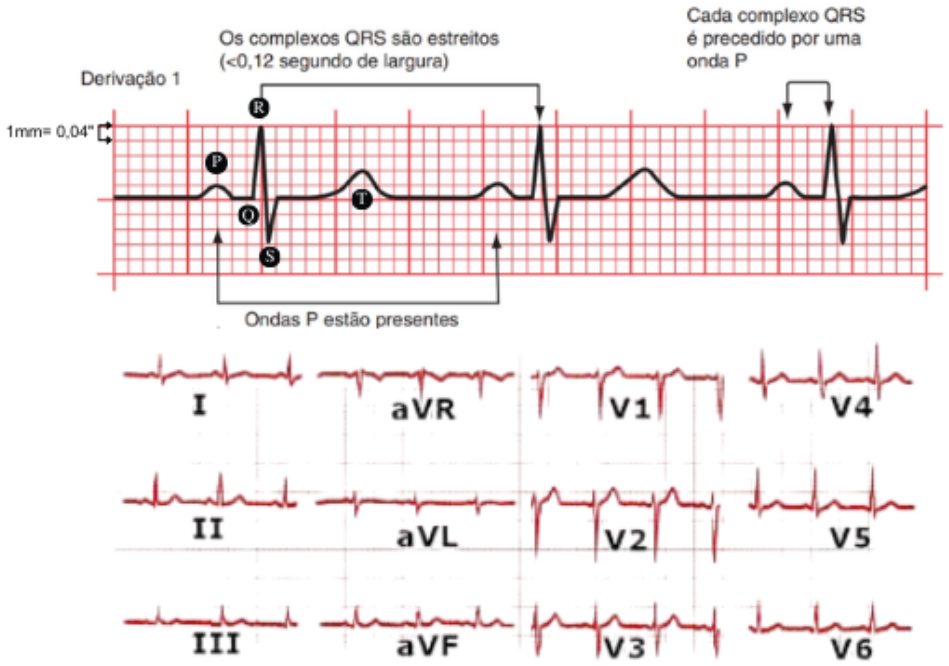
O método que utiliza transformadas é o mais utilizado por apresentar resultados eficiêntes em termos de CR e reconstituição de sinal. De acordo com (CHANDRA; SHARMA; SINGH, 2021) existem vários métodos de transformação além da WT usados para executar a compactação de dados, como; transformada rápida de Fourier (FFT), particionamento de conjuntos em árvores hierárquicas (LSPIHT), técnica de interpolação sinc discreta (DSI), DCT e método de compressão baseado em pirâmide laplaciana, dentre outros.

2.4 Compressão por transformada

Um sinal de ECG exibe uma certa quantidade de redundância, manifestada pela correlação entre amostras adjacentes, a recorrência de batimentos cardíacos com morfologia semelhante e a semelhança relativa entre diferentes derivações como demonstrado na Figura 2.4. Economias consideráveis podem ser obtidas em termos de capacidade de armazenamento e tempo de transmissão explorando os diferentes tipos de redundância para que cada amostra possa ser

representada por menos bits do que no sinal original. Assim, o algoritmo de compressão de dados deve explicar o fato de que o sinal contém batimentos cardíacos recorrentes, muitas vezes com morfologia semelhante, e que o sinal é, quase invariavelmente, uma gravação multilead (SÖRNMO LEIF E LAGUNA, 2006).

Figura 2.4: Morfologia do ECG e suas derivações



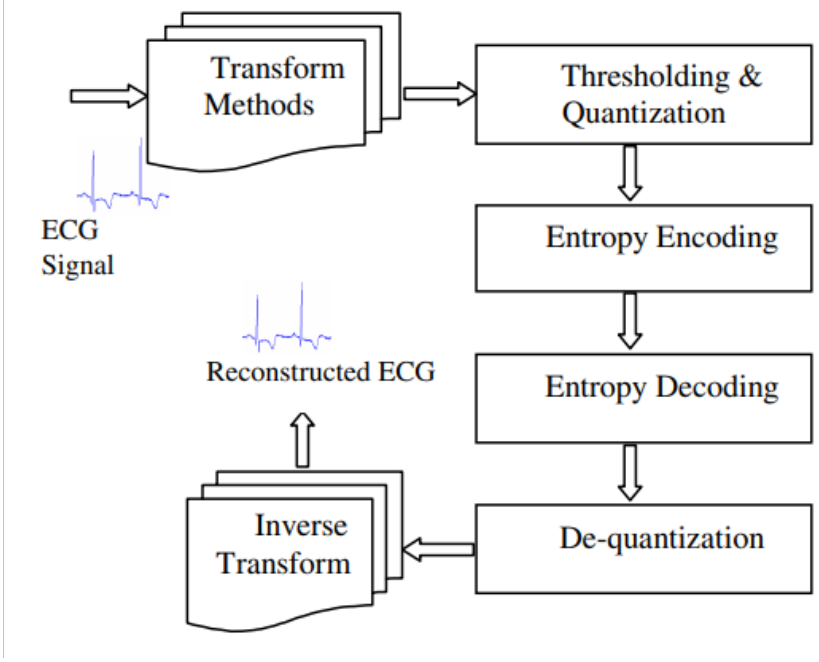
Fonte: Adaptado de ENFERMAGEM (2024)

Métodos de compressão baseados em transformadas são métodos de compressão de ECG com perdas ou quase sem perdas no domínio da frequência. A compactação e descorrelação de energia são as principais propriedades de qualquer transformada, que torna a representação do sinal adequada para compressão. Segundo (OLIVEIRA, 2024), as transformações desempenham um papel fundamental no Processamento Digital de Sinais, sendo amplamente utilizadas em áreas como reconhecimento de padrões, diagnóstico médico, biometria e, no contexto deste trabalho, compressão de dados. Certas transformadas possuem propriedades específicas que as tornam especialmente eficazes para compactação. Por exemplo, transformadas trigonométricas baseadas na função cosseno concentram a maior parte das informações essenciais de sinais contínuos digitalizados nos coeficientes de baixa frequência. Por outro lado, as wavelets oferecem uma abordagem hierárquica, separando os coeficientes de alta e baixa frequência em diferentes níveis, o que possibilita a codificação seletiva das frequências mais relevantes para minimizar erros na reconstrução do sinal.

O projeto de uma transformação para compactação deve focar na redução de redundâncias presentes entre as amostras, buscando concentrar a informação essencial em um número menor de coeficientes, permitindo maior eficiência na codificação e armazenamento dos dados. As etapas principais de grande parte dos métodos de compressão baseados em transformadas são apresentadas na Figura 2.5. Nesse processo, o sinal de ECG é inicialmente submetido a

uma transformada, sendo utilizada neste trabalho a transformada wavelet, conforme descrito na seção 2.5. Após a transformação para o domínio da frequência, aplicam-se as etapas de *thresholding* e quantização, detalhadas nas seções 2.6 e 2.7, respectivamente. Por fim, os dados são processados pela etapa de codificação de entropia 2.7.1, que finaliza o processo de compressão. O ciclo de reconstrução segue o caminho inverso, aplicando as transformações inversas em cada etapa mencionada, até a recuperação do sinal no domínio original.

Figura 2.5: Método de compactação baseado em transformação



Fonte: (KUMAR; KUMAR; PANDEY, 2010)

2.5 Transformada wavelets (WT)

Wavelets são funções base que permitem representar uma função em múltiplos níveis de detalhe. Uma transformada wavelet é construída a partir de sucessivas aproximações de um sinal de entrada, utilizando uma função base denominada wavelet-mãe. Essa transformada gera um sinal de saída estruturado conforme o domínio definido pela função base. As etapas de aproximação realizadas no sinal são denominadas passos da wavelet, e, à medida que esses passos são executados, o sinal é analisado em diferentes escalas, desde níveis mais detalhados até representações de larga escala (OLIVEIRA, 2024).

Considerando as transformada lineares em tempo discreto (Seção 2.2) para o sinal de ECG, a WT se destaca por realiza uma análise multirresolução, fornecendo informações simultâneas nos domínios do tempo e da frequência. Essa característica é especialmente útil para capturar eventos transitórios e variações locais no sinal, como a detecção de complexos QRS e ondas P e T no ECG. (MANZAN, 2006) enfatiza que a flexibilidade da WT em ajustar escalas e deslocamentos permite adaptar a análise às diferentes morfologias do ECG, algo que não é

possível com outras transformadas.

Uma função $\psi(t)$ pertencente ao espaço $L^2(\mathbb{R})$, cuja norma seja unitária, é denominada de função wavelet se a condição de admissibilidade, dada na equação (2.5), for satisfeita (DAUBECHIES, 1992).

$$C_\psi = 2\pi \int_{-\infty}^{\infty} \frac{|F_\psi(\omega)|^2}{|\omega|} d\omega < \infty, \tag{2.5}$$

onde $F_\psi(\omega)$ é a transformada de Fourier (FT) de $\psi(t)$.

Para assegurar a convergência da integral na equação (2.5), é necessário que $F_\psi(0) = 0$ (MALLAT, 1999). Essa condição estabelece que:

$$0 = F_\psi(0) = \int_{-\infty}^{\infty} \psi(t) e^{-i0t}\, dt = \int_{-\infty}^{\infty} \psi(t)\, dt. \tag{2.6}$$

De acordo com (OLIVEIRA, 2018) , a condição de média nula estabelecida na equação 2.6, que é equivalente a relação da condição de admissibilidade, e o fato de $\psi(t) \in L^2(\mathbb{R})$ garantem o comportamento ondulatório das wavelets, onde a área acima e abaixo do eixo t se anulam, e $\psi(t)$ decai rapidamente conforme $t \to \pm\infty$. Dessa forma, as wavelets possuem um suporte curto, com energia concentrada em uma região limitada, assegurando uma boa localização temporal.

A propriedade de localização das funções wavelet é uma característica que permite que elas sejam simultaneamente compactadas no domínio do tempo e no domínio da frequência, o que as diferencia das funções sinusoidais da Transformada de Fourier. Uma função wavelet $\psi(t)$ é dita localizada no tempo se sua energia total for finita e concentrada ao redor de $t = 0$. Isso é descrito pelo momento quadrático no domínio do tempo:

$$E_t = \int_{-\infty}^{+\infty} t^2 |\psi(t)|^2\, dt < \infty, \tag{2.7}$$

onde E_t mede a dispersão temporal da wavelet.

A função wavelet $\psi(t)$ é localizada na frequência se sua transformada de Fourier $\hat{\psi}(\omega)$, definida como:

$$\hat{\psi}(\omega) = \int_{-\infty}^{+\infty} \psi(t) e^{-j\omega t}\, dt, \tag{2.8}$$

também tiver energia concentrada em uma região limitada no domínio da frequência. O momento quadrático no domínio da frequência é dado por:

$$E_\omega = \int_{-\infty}^{+\infty} \omega^2 |\hat{\psi}(\omega)|^2\, d\omega < \infty, \tag{2.9}$$

onde E_ω mede a dispersão frequencial.

A WT faz uso do princípio MRA e consiste em utilizar wavelets como funções de base para realizar a transformação. O nome wavelet significa onda

> pequena. Pequena se refere a que a função de base utilizada é de duração finita. Onda se refere a que a função de base é de caráter oscilatório, ou seja, que sua integral no intervalo de duração da função é igual a 0. Existe outro termo importante denominado wavelet mãe, que é utilizado para explicar que as diferentes funções de base (de diferente comprimento) usadas no processo de transformação são obtidas a partir de uma função principal, chamada wavelet mãe (CAMPITELLI, 2015).

A WT é uma ferramenta matemática utilizada na análise de sinais não estacionários que tem como principio de fundamental dividir o sinal em diferentes funções usando a wavelet mãe (ϕ(t)), definida (CHANDRA; SHARMA; SINGH, 2021):

$$\phi_a(t) = a^{-}\frac{1}{2}\varphi\left(\frac{t-b}{a}\right), a, b \in R \tag{2.10}$$

onde a é a função de dilatação e b a de translação.

Ao calcular a Transformada Wavelet, utiliza-se a wavelet mãe como protótipo para gerar todas as funções de base, também chamadas de janelas. Inicialmente, a wavelet é comprimida, representando frequências mais altas, e multiplicada pelo sinal. Em seguida, o resultado dessa multiplicação é integrado ao longo de todo o intervalo de tempo e normalizado para garantir que a energia do sinal transformado permaneça consistente em todas as escalas (CAMPITELLI, 2015).

Após essa etapa, a wavelet comprimida é deslocada ao longo do sinal, repetindo-se o processo de multiplicação, integração e normalização em cada posição até atingir o final do sinal. Em seguida, a wavelet é ligeiramente dilatada, esse processo de dilatação, deslocamento e cálculo se repete para todas as escalas da wavelet até a conclusão da Transformada Wavelet.

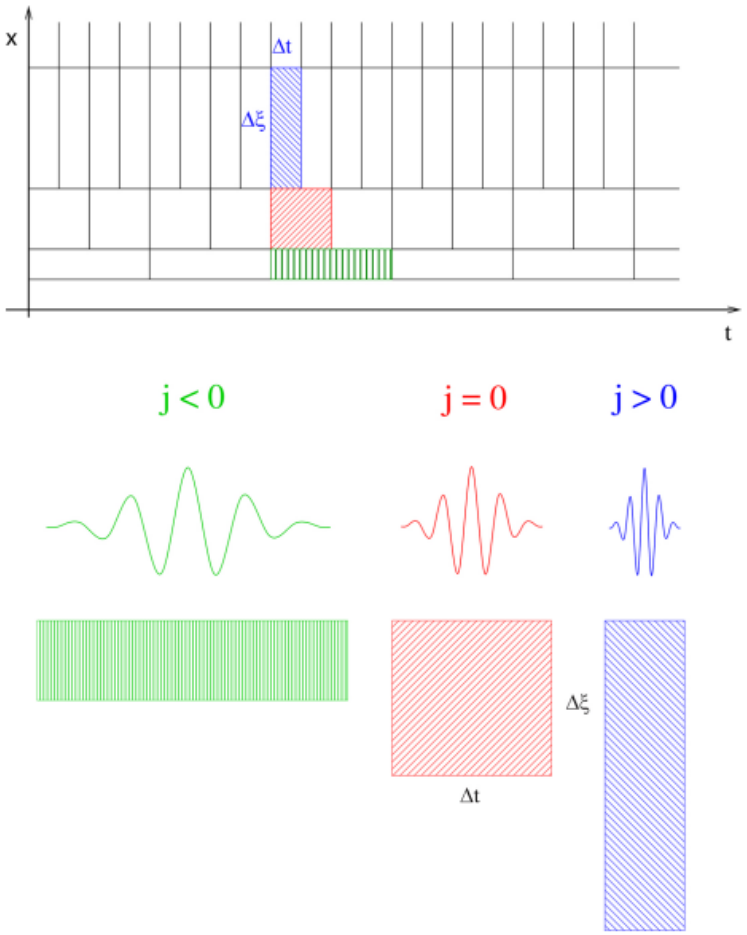
A WT pode ser dividida em dois tipos principais: a Transformada Wavelet Contínua (CWT) e a Transformada Wavelet Discreta (DWT). Estudos demonstram que a CWT é eficaz na análise de sinais de ECG, auxiliando na detecção de componentes como ondas P, complexos QRS e ondas T, essenciais para diagnósticos clínicos. Por exemplo, a pesquisa de LI; ZHANG; WANG (2022) propõe um método de identificação de sinais de ECG baseado na transformada wavelet, destacando a capacidade da CWT em detectar ondas Q, R e S, bem como ondas P e T, facilitando a identificação de anomalias cardíacas.

Além disso, a CWT é eficiente na detecção de mudanças sutis no sinal, como variações na morfologia do complexo QRS, que podem indicar arritmias ou isquemia. O estudo de MALEKI (2024) explora a aplicação de transformadas wavelet na classificação de sinais de ECG para identificar diversas condições cardiovasculares, evidenciando a eficácia da CWT na extração de características significativas que auxiliam na detecção de anomalias cardíacas.

A CWT pode ser expressa como:

$$Wf(b,a) = |a|^{-\frac{1}{2}} \int_{-\infty}^{\infty} f(t)\varphi * \left(\frac{t-b}{a}\right) dt \tag{2.11}$$

Figura 2.6: Representação de tempo e frequência para as janelas da WT



Fonte: Adaptado de DOMINGUES et al. (2016)

A multiplicação de $|a|^{-\frac{1}{2}}$ e é usado para estabilizar o sinal transformado com a mesma energia em todas as escalas. No caso da CWT, os valores de a e b são contínuos sobre o número real (R), o que aumenta a complexidade computacional.

A Transformada Wavelet Discreta é usada em uma variedade de aplicações de processamento de sinais, como compressão de vídeo, compressão de comunicações pela Internet, reconhecimento de objetos e análise numérica WEEKS (2007). Adicionamente o livro WEEKS (2007) ressalta que esta transformação é discreta em tempo e escala. Em outras palavras, os coeficientes DWT podem ter valores reais (ponto flutuante), mas os valores de tempo e escala usados para indexar esses coeficientes são inteiros. a hierarquia de decomposição da WTD permite uma análise multirresolução do sinal, além de também possibilitar a redução da redundância de dados por possibilitar maior robustez na análise do sinal.

O DWT é representado pela equação:

$$Wx(m,n) = \int_{-\infty}^{\infty} x(t)\varphi_{m,n}(t)dt = a_0^{-\frac{m}{2}} \int_{-\infty}^{\infty} x(t)\varphi_{m,n}(t)dt \tag{2.12}$$

Esta equação pode ser encontrada ao substituir os valores de $a = a_0^m$ e $b = nb_0a_0^m$ para diferentes valores de n e m. Quando, a_0 = 2 e b_0 = 1, uma nova wavelet discreta pode ser

construído, que é dado em Eqn.2.13. Esta wavelet consiste em uma base ortogonal para L2(R).

$$\varphi_{m,n}(t) = 2^{-\frac{m}{2}}\varphi(2^{-m}t - n) \tag{2.13}$$

Como desenvolvido em CHANDRA; SHARMA; SINGH (2021) na DWT, o sinal de entrada é decomposto em diferentes escalas pela seguinte expressão dada abaixo:

$$x(t) = \sum_{j=1}^{k}\sum_{k=-\infty}^{\infty} d_j(k)\varphi_{j,k}(t) + \sum_{k=-\infty}^{\infty} a_k(k)\psi k,k(t) \tag{2.14}$$

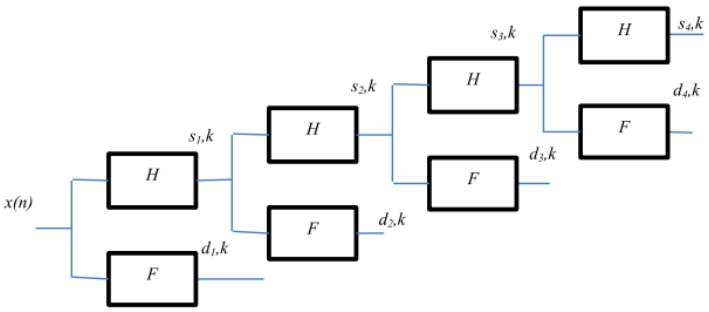
onde, $d_j(k)$, φ,k(t) e ψ K,k(t) são sinais de detalhe, de análise discreta e a escala da função discreta, respectivamente. Na transformada wavelet, apenas bandas de frequência mais baixas são decompostas, conforme mostrado na 2.7, no entanto, na decomposição de pacotes wavelet, os sinais de detalhe também são decompostos. A Figura 2.8 mostra a decomposição de um sinal usando o algoritmo de pacote wavelet.

A árvore de decomposição wavelet é descrita na Figura 2.7 e a implementação de pacotes wavelet pode ser feita usando as equações 2.15 e 2.16.

$$h(n) = \frac{1}{2}(\psi(t), \psi(2t - n)) \tag{2.15}$$

$$g(n) = \frac{1}{\sqrt{2}}(\varphi(t), \psi(2t - n)) = \int_{-\infty}^{\infty}(-1)^n h(1 - n) \tag{2.16}$$

Figura 2.7: Decomposição wavelet

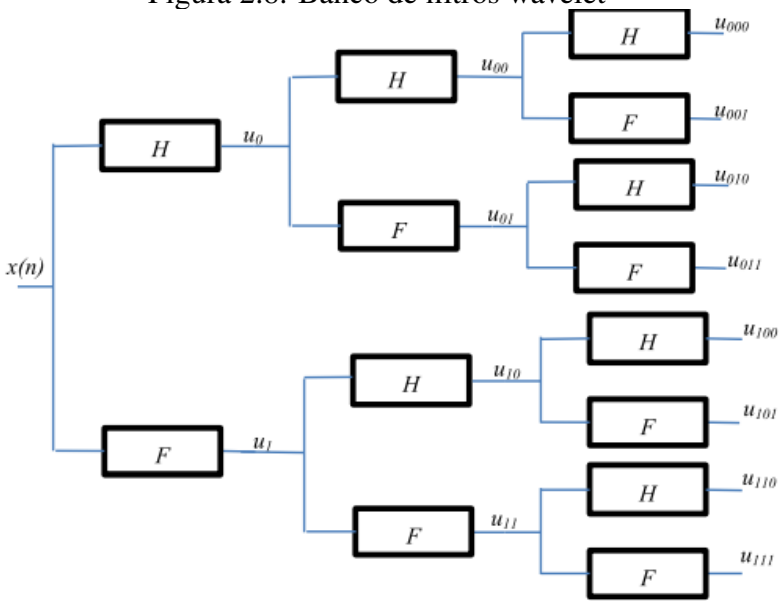


Fonte: Adaptado de CHANDRA; SHARMA; SINGH (2021)

Os filtros wavelet determinam a representação eficiente das morfologias do ECG, o que afeta diretamente no desempenho da compressão. A distorção mínima entre o sinal original e reconstruído desempenha um papel significativo na seleção ideal do filtro wavelet e no nível de decomposição para a taxa de compressão desejada.

Uma característica fundamental da DWT é sua eficiência computacional, com complexidade linear, ou seja, $O(N)$, onde N representa o número de amostras do sinal. Conforme abordado no artigo (ABREU MOREIRA; SOUZA, 2015), sua implementação eficiente baseia-se no algoritmo piramidal, que utiliza bancos de filtros passa-baixa e passa-alta, seguidos de

Figura 2.8: Banco de filtros wavelet



Fonte: Adaptado de CHANDRA; SHARMA; SINGH (2021)

subamostragem, conforme descrito nesta seção. Essa estrutura reduz progressivamente o tamanho do sinal processado em cada nível, resultando em uma soma de operações proporcional ao número total de amostras do sinal.

Comparada a outras transformadas, como a Transformada Discreta de Fourier (DFT), com complexidade $O(N^2)$, e sua versão otimizada, a FFT, com complexidade $O(N \log N)$, a DWT apresenta a menor complexidade computacional. Essa característica a torna ideal para aplicações em tempo real e para processamento de grandes volumes de dados.

Diversos estudos abordam as transformadas Wavelet's citadas, como no artigo CHANDRA; SHARMA; SINGH (2021) que apresenta um estudo cujo a compressão de dados de ECG de derivação única é realizada usando Wavelet Packet (WP) (2.8), que é um algorítmo que realiza processamento e análise de sinais, com alta eficiência e flexibilidade. Técnica realizada através da decomposição do sinal em número de bandas de frequência, em um processo em cadeia. Já em outro estudo a detecção de picos R, intervalo R-R e compressão de dados de ECG é feita usando a DWT, este trabalho proporciona uma alta taxa de compressão e baixa complexidade de implementação.

2.6 Limiarização

As técnicas de limiarização dos coeficientes de uma série wavelet têm como objetivo reduzir, ou até eliminar, o ruído presente em um sinal, contribuindo para a melhoria da qualidade e representatividade do mesmo (KOZAKEVICIUS; BAYER, 2014). Esse processo consiste em aplicar um valor de limiar λ, que define critérios para decidir quais valores serão mantidos, ajustados ou descartados. Métodos de limiarização são amplamente utilizados em áreas

como compressão de dados, análise de imagens e processamento de sinais biomédicos (ENGENHARIA, 2023), desempenhando um papel essencial no processo de compressão de sinais. A Figura 2.5 ilustra o fluxo geral desse processo, destacando a importância da limiarização como uma etapa intermediária fundamental.

Na limiarização fixa, o limiar é definido de forma estática e aplicado uniformemente ao sinal ou imagem essa metodologia é altamente aplicada em compressão JPEG, onde coeficientes de alta frequência muitas vezes são limiarizados (WALLACE, 1991). Analogamente a limiarização Adaptativa implementa uma variação de λ de acordo com as características locais do dado, ajustando-se às variações na densidade de informação. No processamento wavelet, coeficientes em diferentes níveis de detalhe podem receber limiares adaptados ao ruído local (MALLAT, 1999). Desta forma é possível preservar mais detalhes em regiões de alta variação.

O trabalho de KOZAKEVICIUS; BAYER (2014) apresenta uma metodologia de aplicação da limiarização em WT, com isso o processo de limiarização é realizado utilizando métodos como o Hard Thresholding, que elimina abruptamente coeficientes menores que λ, e o Soft Thresholding, que reduz gradualmente esses coeficientes. Embora o Hard Thresholding apresente menor viés, sua variância é maior. Já o Soft Thresholding, por sua vez, resulta em sinais mais suaves, embora com viés mais elevado. Um aspecto fundamental no sucesso do método é a seleção do valor de limiar λ. Diferentes abordagens para determinar esse limiar podem ser encontradas na literatura, variando conforme a natureza do ruído e as características do sinal.

> Considera-se o vetor T-dimensional $f = (f_1, f_2, \ldots, f_T)$, com $T = 2^n$ e $n \in \mathbb{Z}^+$, onde cada componente f_i representa uma amostra do sinal sem ruído no instante t_i, ou seja, $f_i = f(t_i)$. [...] o sinal amostrado y pode ser considerado como uma soma de uma componente limpa f e um ruído e, ou seja:
>
> $$y = f + e, \quad (y_1, y_2, \ldots, y_T) = (f_1 + e_1, f_2 + e_2, \ldots, f_T + e_T).$$
>
> [...] Como a WT é uma transformação linear e ortogonal, temos:
>
> $$\mathrm{TW}(y) = \mathrm{TW}(f + e) = \mathrm{TW}(f) + \mathrm{TW}(e).$$
>
> Para ruídos gaussianos, a média zero e a variância constante σ^2 do ruído são preservadas após a transformação. Em termos de coeficientes wavelet, isso pode ser expresso como:
>
> $$d_{j,k} = w_{j,k} + \sigma z_{j,k},$$
>
> onde $d_{j,k}$ são os coeficientes wavelet do sinal y, $w_{j,k}$ são os coeficientes wavelet do sinal limpo f e $z_{j,k} \sim N(0,1)$. A estimativa de f é obtida pela limiarização dos coeficientes $d_{j,k}$. O operador $\mathrm{thr}_\lambda(\cdot)$ reduz ou anula os coeficientes $d_{j,k}$ menores que um limiar λ KOZAKEVICIUS; BAYER (2014).

Definição matemática segundo KOZAKEVICIUS; BAYER (2014)

1. Hard Thresholding:

$$\text{thr}_{\lambda}^{H}(d_{j,k}) = \begin{cases} 0, & \text{se } |d_{j,k}| \leq \lambda, \\ d_{j,k}, & \text{se } |d_{j,k}| > \lambda. \end{cases} \tag{2.17}$$

2. Soft Thresholding:

$$\text{thr}_{\lambda}^{S}(d_{j,k}) = \begin{cases} 0, & \text{se } |d_{j,k}| \leq \lambda, \\ \text{sign}(d_{j,k})(|d_{j,k}| - \lambda), & \text{se } |d_{j,k}| > \lambda. \end{cases} \tag{2.18}$$

2.7 Quantização

De acordo com (SAYOOD, 2017), a quantização é o processo de mapear um conjunto extenso, potencialmente infinito, de valores para um conjunto significativamente menor. Este processo é essencial na compressão de sinais, pois permite a redução da quantidade de dados necessários para representar um sinal, facilitando o armazenamento e a transmissão eficientes. Neste contexto, a quantização representa uma etapa frequente nos processos de compressão com transformada, como ilustrado em 2.5.

Considerando os achados na literatura, há diversos tipos de quantização com suas diferentes especificidades. Dentre eles há, a quantização uniforme, a qual contém um espaço igual entre cada nível existente e o intervalo se mantém constante para cada nível, ou a não uniforme que apresenta diferentes espaçamentos para cada nível de quantização. A quantização com *dead-zone* é também denominada semi-uniforme pois o intervalo próximo ao nível zero é maior em comparação com os demais que possuem o mesmo intervalo (TONIN, 2021). A quantização compacta é uma abordagem que expande a região de entrada onde os valores têm alta probabilidade de concentração, otimizando a representação dos dados em torno da origem. Esse método é particularmente útil em situações onde os valores de entrada apresentam uma distribuição não uniforme, pois ajusta os intervalos de quantização para refletir melhor a densidade de probabilidade dos dados. A principal vantagem é que, ao alocar mais níveis de quantização em regiões de alta probabilidade, minimiza-se o erro médio de quantização nessas áreas críticas (SALOMON, 2007).

Por outro lado, a quantização vetorial é uma técnica mais avançada que codifica blocos de amostras simultaneamente, em vez de processar uma única amostra por vez. Essa abordagem explora as correlações entre as amostras no bloco, permitindo uma maior eficiência na compressão de dados. Segundo (GERSHO; GRAY, 1992), "a quantização vetorial reduz significativamente o número de bits necessários para representar os dados, ao mesmo tempo que preserva informações relevantes para a reconstrução".

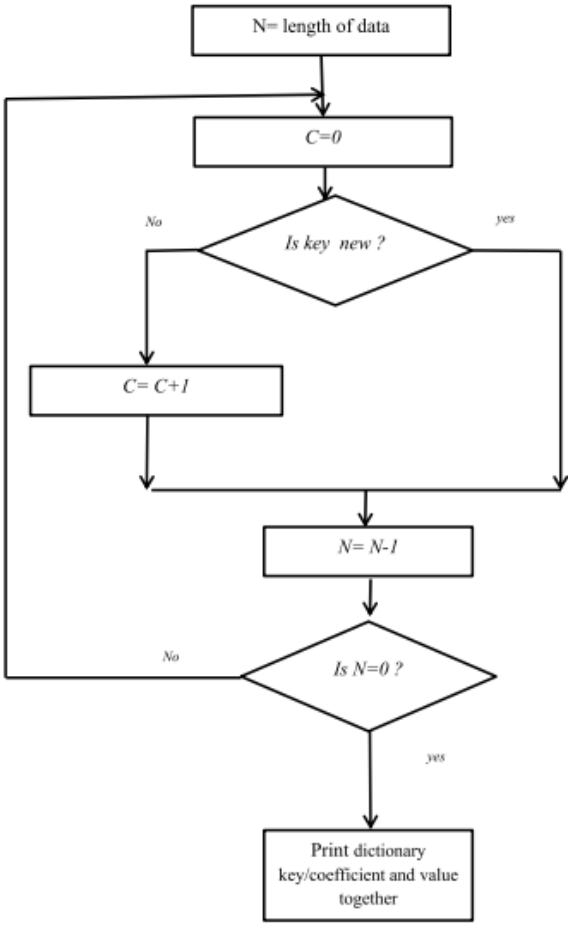
2.7.1 Codificação

Existem diversas metodologias de codificação, muitas se destacam em aplicações com WT como as abordadas em (CHANDRA; SHARMA; SINGH, 2021), sendo elas: Run-length, Huffman e codificação LZW.

2.7.1.1 Run-length encoding (RLE)

A RLE é uma técnica de LLDC, na qual os dados são representados pelo número de repetição de um dado seguido pela representação do mesmo. Por exemplo: aaaabbccccccrrrrrrffff pode ser representado como 4a2b6c6r4f.

Figura 2.9: Lógica RLE



Fonte: Adaptado de CHANDRA; SHARMA; SINGH (2021)

As etapas apresentadas no artigo CHANDRA; SHARMA; SINGH (2021) para execução do RLE, são:

- Passo 1. Localize o comprimento dos dados/vetor e, consequentemente, inicie um alongamento de loop para o comprimento da cadeia de caracteres.
- Passo 2. Aqui, exame da chave/coeficiente.

- Passo 3 (a) Se a chave for nova, a contagem será igual a zero, haverá adição na chave e o valor para o dicionário é feito.
- Passo 3 (a) Se a chave for nova, a contagem será igual a zero, haverá adição na chave e o valor para o dicionário é feito.
- Passo 3(b) Caso contrário, se a chave não for nova, haverá incremento no valor do contador.
- Passo 4. A reparação das Etapas 2 e 3 será feita continuamente até que o loop seja encerrado.
- Passo 5. Imprima a chave/coeficiente e o valor do dicionário juntos. Estes passos são dados na Fig.

2.7.1.2 Huffman

A codificação de Huffman, foi desenvolvida em 1952 durante seu doutorado no Instituto de Tecnologia de Massachusetts (MIT), é considerada um exemplo clássico de codificação estatística. O método utiliza as probabilidades de ocorrência dos símbolos em um conjunto de dados para atribuir códigos binários de tamanho variável, priorizando a eficiência na representação dos dados (SANTOS, 2024).

> "Um código de prefixo ótimo (menor comprimento esperado) para uma distribuição dada pode ser construído por um algoritmo simples descoberto por Huffman"COVER; THOMAS (2006).

Segundo HUFFMAN (1952), o algoritmo associa árvores ponderadas a símbolos, combinando iterativamente aquelas com menores pesos para criar uma estrutura hierárquica, onde os códigos binários são determinados com base na posição na árvore. A cada iteração do algoritmo, as duas árvores de menor peso são substituídas por uma nova árvore cujo peso é a soma dos pesos das primeiras. A árvore de menor peso se torna a subárvore esquerda e a outra se torna a subárvore direita da nova árvore. Na ordenação dos pesos, empates são resolvidos por qualquer regra sistemática. O procedimento pára quando resta apenas uma única árvore. A palavra-código para qualquer letra é obtida percorrendo-se esta árvore desde a raiz até a folha correspondente à letra em questão, registrando 0 para cada ramo esquerdo e 1 para cada ramo direito.

COVER; THOMAS (2006) destaca que “[...]combinando os dois símbolos menos prováveis em um símbolo até finalmente ficarmos com apenas um símbolo, e então atribuindo palavras-código aos símbolos”. A 2.10 demonstrada em COVER; THOMAS (2006), exemplifica as alocações das árvores segundo a frequência em que os elementos estão presentes vetor de entrada x a partir do método de minimização ponderada:

Em (CHANDRA; SHARMA; SINGH, 2021) essa codificação é feita com base na frequência em que os dados de entrada se repetem, logo uma estrutura de árvore é criada onde o zero

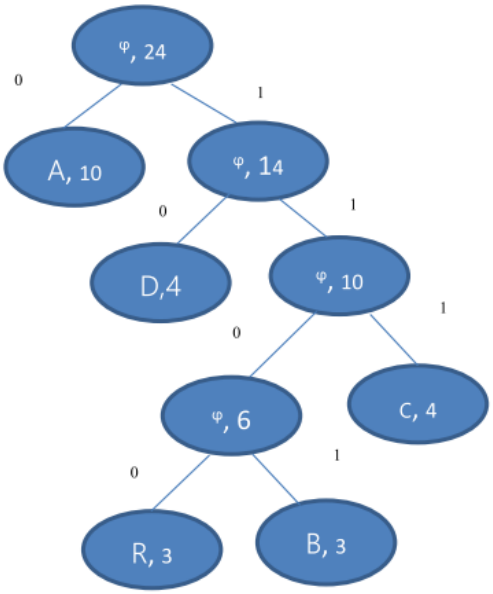
Figura 2.10: Prioridade da árvore Huffman

X	Codeword	Weights
1	00	5 8 10 18
2	01	5 5 8
3	10	4 5
4	11	4

Fonte: Adaptado de COVER; THOMAS (2006)

representa o lado esquerdo e o um o lado direito. Cada folha contêm uma letra e sua contagem de frequência, além disto em todos os outros nós, em vez de um caractere, um zero, e a contagem de sua frequência e seus descendentes é representada.

Figura 2.11: Árvore Huffman



Fonte: CHANDRA; SHARMA; SINGH (2021)

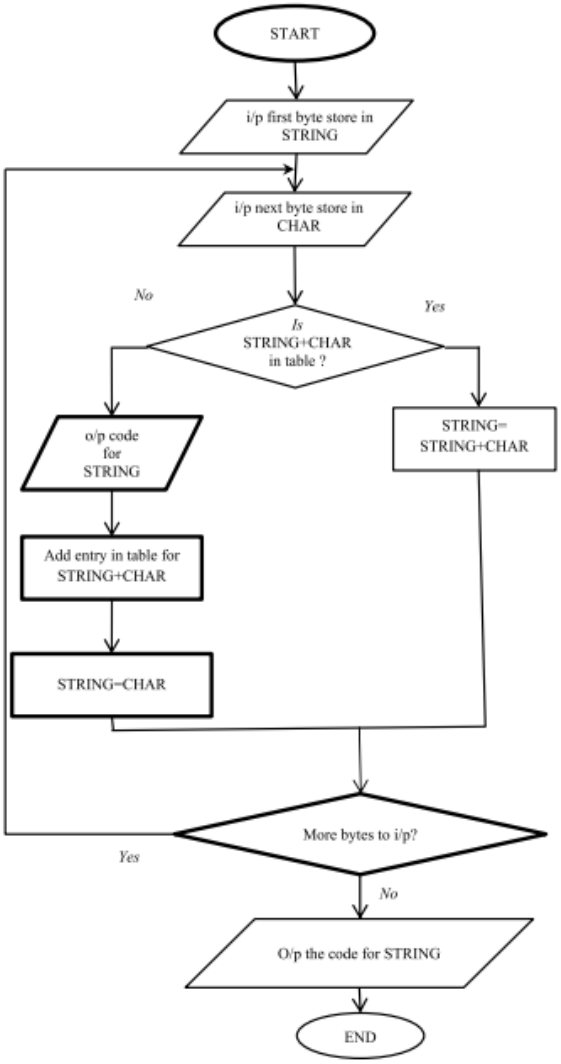
2.7.1.3 Codificação LZW

Também abordado em (CHANDRA; SHARMA; SINGH, 2021) uma das vantagens da implementação dessa codificação é sua simplicidade, normalmente esse método pode compactar dados que resultam em um arquivo com a metade do tamanho original, sendo assim em casos como números tabulados, códigos fontes do computador e sinais, este método oferece uma ótima CR. Para essa decodificação é utilizada uma tabela que usualmente fornece 4096 entradas, consistindo em códigos de 12 bits. A descompactação utiliza a tabela de códigos para converter cada caractere apartir de cada código do arquivo compactado. Os códigos 0-255 na tabela de código são atribuídos para representar bytes únicos do arquivo de entrada. As etapas

a seguir demonstram a execução do LZW segundo CHANDRA; SHARMA; SINGH (2021):

- Passo 1. A inicialização do dicionário é feita.
- Passo 2. A estimativa da cadeia de caracteres mais longa é feita no dicionário que corresponde à entrada atual.
- Passo 3. Redução do dicionário é feita, para W para saída e remover W da entrada.
- Passo 4. A adição de "W"é feita e seguida pelo coeficiente/símbolo de entrada na entrada.
- Passo 5. O passo 2 é realizado.

Figura 2.12: Lógica LZW



Fonte: Adaptado de CHANDRA; SHARMA; SINGH (2021)

Esse método funciona de forma proeminente em dados com as configurações repetitivas; portanto, as partes iniciais de qualquer informação veem pouca compactação. À medida que a informação aumenta de tamanho, no entanto, a RC tende assintoticamente ao máximo.

2.8 Registro Eletrônico e Transmissão de um ECG pela internet

O desenvolvimento e a adoção de Registros Eletrônicos de Saúde (RES) busca padrões estruturais e semânticos para representar e compartilhar dados de saúde de pacientes entre seus diferentes atores em ecossistemas de aplicações de saúde digital. Um dispositivo IoT como o proposto neste trabalho é apenas um ator neste ecossistema que precisa compartilhar dados com diversos outros sistemas de forma interoperável. Dentre os diversos padrões de interoperabilidade de RES existentes, o FHIR – (HL7, 2019; BENDER; SARTIPI, 2013) – se destaca internacionalmente porque foca no desenvolvimento de aplicações web por meio de API's (SARIPALLE, 2019).

2.8.1 Registro de um ECG no padrão FHIR

O padrão FHIR descreve os RES através de modelos chamados **Recursos**. Os RES do Recursos FHIR podem ser representados em formatos como JSON e XML. Dentre os mais 140 tipos de recursos documentados, o recurso *Observation* é destaque neste trabalho porque representa medidas em geral como pressão, temperatura e também o ECG.

Apesar de conter muitas chaves especificadas no objeto JSON, a declaração necessária para especificar um ECG no padrão FHIR depende de alguns parâmetros obrigatórios, mas podem variar de acordo com o objetivo do armazenamento dos dados. O exemplo do Código-fonte 2.1 destaca apenas os mais importantes para compreensao deste trabalho:

- `resourceType` - campo obrigatório que define o recurso;
- `status` - estado do recurso, podendo ser finalizado, preliminar, registrado ou alterado;
- `component` - é um array de objetos JSON que definem o registro de uma derivação do ECG;
    - `display` - código da derivação do ECG;
    - `period` - taxa de amostragem do ECG, em milissegundos;
    - `data` - string contendo cada amostra que compõe o ECG para determinada derivação;

2.8.2 Envio de um ECG para APIs na nuvem

O padrão FHIR foi desenvolvido com foco em aplicações web, pois facilita que APIs possam realizar a troca de dados por meio de requisições web no padrão HTTP (SARIPALLE, 2019). Essas interações seguem o paradigma RESTful de gerenciamento de estado por ações Criar/Ler/Atualizar/Excluir (CRUD - Create/Read/Update/Delete) no conjunto de recursos (FERREIRA, 2017). Considerando a troca de RES em formato JSON no padrão FHIR, as requisições HTTP mais conhecidas e que realizam as operações CRUD são:

Código-fonte 2.1: Exemplo de um *Resource* FHIR *Observation* descrevendo um ECG

```
{
    "resourceType": "Observation",
    "status": "final",
    "component": [
        {
            "code": {
                "coding": [
                    {"display": "MDC_ECG_LEAD_AVR"}
                ]
            },
            "valueSampledData": {
                "period": 10,
                "data": "2041 2043 2037 ... 2027 2034 2033"
            }
        },
        {
            "code": {
                "coding": [
                    {"display": "MDC_ECG_LEAD_AVL"}
                ]
            },
            "valueSampledData": {
                "period": 10,
                "data": "934 932 960 ... 920 950 960"
            }
        }
    ]
}
```

Fonte: Adaptado de PEREIRA et al. (2023)

- `GET` - Obtém algum(s) recurso(s) FHIR previamente cadastrado(s);
- `POST` - Cadastra/Grava um recurso FHIR enviado como corpo da requisição;
- `PUT` - Altera/Atualiza algum(s) recurso(s) FHIR já cadastrados;
- `DELETE` - Exclui algum(s) recurso(s) FHIR cadastrado.

Diversas implementações *open source* do padrão FHIR podem ser encontradas (AYAZ et al., 2021), onde a que mais se destaca é a HAPI FHIR (CDR, 2019), uma API escrita em Java que implementa completamente o padrão FHIR ao disponibilizar operações CRUD para todos os tipos de recursos do FHIR. HAPI FHIR é amplamente utilizada por pesquisadores para fazer testes e validar registros FHIR ou outras APIs.

FASS-ECG é outra API FHIR relevante para este trabalho por ser uma API especializada em biossinais e o ECG foi utilizado como estudo de caso PEREIRA et al. (2023). FASS-ECG

fornece suporte ao streaming e armazenamento de ECGs de 12 derivações de longo prazo.

Sobre o processo de envio de ECGs a longo prazo, é visto que o FASS ECG estipula uma sequência de múltiplas requisições HTTP que os dispositivos ECG precisam seguir e que permitem o envio do sinal de forma distribuída. O processo inicia-se pela requisição POST que cria um novo recurso *Observation* e inicia a transmissão de ECG, gerando um identificador `id` para registro. Em seguida, uma requisição do tipo **`PATCH`** envia um JSON (que não é um recurso FHIR) contendo novas amostras do ECG criado anteriormente para serem concatenadas as amostras ja existentes. A quantidade de requisições PATCH não tem limitação, tampouco a quantidade de amostras do sinal do ECG enviadas por requisição PATCH. A requisição PUT finaliza a transmissão ECG atualizando a chave `status` com o valor `final`.

Posto isso, FASS-ECG realiza o fluxo de gravação assíncrona e é uma API mais adequada para as exigências de dispositivos IoT, possibilitando assim que os tempos ociosos não comprometam a funcionalidade em tempo real.

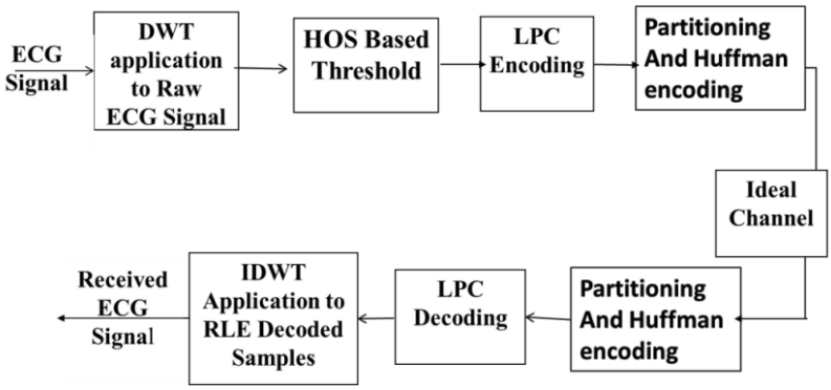
3 TRABALHOS RELACIONADOS

O tema deste trabalho é bastante específico, logo três trabalhos relacionados são revisados para, ao final desta Seção, realizar uma análise comparativa entre os trabalhos revisados e este trabalho.

3.1 Wavelet transform and Huffman coding based electrocardiogram compression algorithm: Application to telecardiology

No artigo de CHOUAKRI; DJAAFRI; TALEB-AHMED (2013) é analisado o desenvolvimento e a avaliação de um algoritimo para a compressão de sinais de eletrocardiograma ECG com foco em sua aplicação em telecardiologia, combinando a transformada wavelet e codificação de Huffman. A pesquisa busca viabilizar os dados necessários para o armazenamento e transmissão de sinais ECG, garantindo a conservação das informações coletadas.

Figura 3.1: Diagrama do método de Compressão



Fonte: Adaptado de CHOUAKRI; DJAAFRI; TALEB-AHMED (2013)

Como objetivo da compressão do sinal de ECG o artigo menciona a quantidade dos custos de transmissão de informações através dos canais de telecomunicações, ou seja, o método manipula um conjunto de dados iniciais por uma determinada codificação para economizar espaço de armazenamento. Figura 3.1 apresenta a visao geral da proposta. Para formulação do método do estudo, é utilizado a Transformada Wavelet para decompor o sinal de ECG diferentes níveis de frequência para evidenciar os componentes essenciais. Tendo em vista o objetivo proposto, os autores utilizaram um algoritmo de compressão de sinal de ECG baseado na transformada wavelet e estatísticas de alta ordem (HOS), usando a codificação de predição linear.

O limiar wavelet é um método não linear para suprimir ruído corrompido para o sinal de interesse. Um limiar é estimado e referindo-se a ele o ruído é removido. Já as medidas de Estatísticas de Alta Ordem são extensões de medidas de segunda ordem (como a função de autocorrelação e espectro de potência) para ordens mais altas. Em seguida, a codificação de Huffman é aplicada para a compressão de dados e assim reduzir o comprimento médio do volume de informações, sem comprometer a qualidade do sinal. Dessa forma, pela codificação de

Huffman o sinal comprimido pode ser transmitido por meio de um canal ideal. Como resultado, (CHOUAKRI; DJAAFRI; TALEB-AHMED, 2013) utiliza diversas métricas de desempenho. Para fins de comparação, três das principais métricas estão representadas na Tabela 3.1:

- CR (Compression Ratio), que indica a relação entre o tamanho original e o tamanho comprimido de um sinal;
- PRD (Percent Root-Mean-Square Difference), que mede a diferença percentual entre o sinal original e o reconstruído, avaliando a distorção relativa;
- MSE (Mean Squared Error), que calcula o erro médio quadrático entre o sinal original e o reconstruído, refletindo a qualidade da reconstrução.

Tabela 3.1: Métricas de compressão para os métodos analisados.

Sinal	CR	PRD (%)	MSE
100.dat	13,76	19,35	0.0719

3.2 Efective high compression of ECG signals at low level distortion

O artigo de REBOLLO-NEIRA (2019) aborda a compressão com perdas, visando alto desempenho na recuperação com baixa distorção. A metodologia utiliza a DWT e apresenta como principal inovação a etapa de "Organização e Armazenamento", que emprega o formato comprimido HDF5. O algoritmo de compressão proposto consiste em três etapas principais: Aproximação, Quantização e Organização e Armazenamento. Na Etapa de aproximação, uma FWT é aplicada ao sinal $f \in \mathbb{R}^N$, transformando-o no vetor $w \in \mathbb{R}^N$, cujos componentes são os coeficientes wavelet, ou seja, é realizado a implementação computacional eficiente da DWT, baseada em convoluções com bancos de filtros e subamostragem. O objetivo é selecionar os coeficientes mais relevantes para aproximar o sinal com a qualidade desejada. Para isso, duas abordagens são consideradas:

- Abordagem (a): Os coeficientes $w(1),\ldots,w(N)$ são ordenados em ordem crescente de valor absoluto. Em seguida, calculam-se as somas cumulativas dos quadrados dos coeficientes $t(n) = \sum_{i=1}^{n} w(\gamma_i)^2$, identificando os coeficientes que atingem o limiar definido (tol^2).
- Abordagem (b): Após a quantização, apenas os coeficientes diferentes de zero e seus índices correspondentes são retidos.

A complexidade computacional da DWT é $O(N)$. No entanto, na abordagem (a), a ordenação dos coeficientes aumenta a complexidade para $O(N \log N)$. Por outro lado, na abordagem (b), a complexidade permanece $O(N)$, pois o número de coeficientes selecionados (K) é geralmente muito menor que N. Em ambos os casos, a compressão de um registro de 30 minutos foi realizada em MATLAB em um tempo médio inferior a 0,2 segundos.

Na etapa de Quantização, esses coeficientes são convertidos em inteiros por meio de um quantizador uniforme, eliminando aqueles mapeados para zero. Por fim, na etapa de Organização e Armazenamento, os coeficientes e seus índices são reorganizados e armazenados de forma compacta no formato HDF5, utilizando diferenças consecutivas para otimizar o espaço.

Os resultados apresentados no artigo foram baseados em testes iniciais conduzidos com diferentes famílias de wavelets e níveis variados de decomposição, permitindo avaliar a influência desses fatores na eficiência e qualidade da compressão. A escolha do formato de armazenamento HDF5 foi um diferencial significativo, eliminando a necessidade de etapas adicionais de codificação de entropia para valores de PRD (Percentagem de Diferença Quadrática Média) superiores a 0,4. Isso resultou em uma redução considerável no tempo de processamento, com compressão de registros de ECG de 30 minutos realizada em menos de 0,2 segundos.

No artigo, são realizados quatro testes numéricos que avaliam a eficácia do método proposto em diferentes cenários. Esses testes analisam aspectos relacionados à escolha de wavelets, comparações com outros métodos de compressão, resultados em benchmarks e características distintivas do algoritmo.

- 1º Teste: Analisa o desempenho de diferentes famílias de wavelets, como Daubechies, Coiflets e Symlets, ajustadas a diversos níveis de decomposição. A escolha ideal da wavelet variou conforme o PRD alvo e a estrutura do sinal. Os níveis de decomposição foram otimizados para balancear compactação e distorção, alcançando alta eficiência sem comprometer a qualidade do sinal.
- 2º Teste: Compara os resultados do método proposto com os obtidos pela abordagem SPHIT (Set Partitioning in Hierarchical Trees). Esta comparação avalia a eficiência em termos de compressão e recuperação, destacando a superioridade do método proposto em certos cenários.
- 3º Teste: Avalia o desempenho do método no banco de dados completo de arritmia do MIT-BIH, com valores médios de PRD variando entre $[0.23, 1.71]$. Os resultados foram comparados com benchmarks de estudos anteriores.
  Segundo (REBOLLO-NEIRA, 2019) o método proposto reproduz o mesmo PRD médio dos benchmarks, mas apresenta ganhos significativos na Taxa de Compressão (CR) e Qualidade de Sinal (QS).
- 4º Teste: Este teste destaca dois aspectos exclusivos do método:

  1. Armazenamento Compactado em HDF5: O algoritmo salva diretamente as saídas no formato HDF5 compactado, eliminando a necessidade de etapas adicionais, como a codificação de entropia (exemplo: codificação de Huffman). Comparações mostraram que o HDF5 reduz significativamente o tamanho do arquivo final e simplifica o processo de compressão.
  2. Organização e Armazenamento: A abordagem (b) do método foi comparada com a codificação Run-Length (RL) tradicional. A organização proposta, que armazena

índices dos coeficientes diferentes de zero, mostrou-se superior em termos de eficiência e flexibilidade, especialmente em cenários que exigem compressão elevada.

Figura 3.2: Comparação de diferentes métodos de armazenamento. CR_a e CR_b são os CRs das abordagens (a) e (b) quando as saídas são salvas diretamente no formato HFD5. CR_a^{Huff} e CR_b^{Huff} são os valores correspondentes quando a etapa de codificação de Hufman é aplicada antes de salvar os dados no formato HFD5. CR_{RL} fornece o CR se as saídas do método (b) são armazenadas usando o algoritmo RL e os arrays são salvos no formato HFD5. CR_{RL}^{Huff} é o CR correspondente se a codificação de Hufman for aplicada antes de salvar os arrays no formato HFD5.

PRD	1.0	0.9	0.8	0.7	0.6	0.5	0.4	0.3	0.2
CR_a	40.51	37.12	33.09	29.70	25.50	22.00	16.80	10.33	6.64
CR_a^{Huff}	43.57	40.41	36.32	32.96	28.80	25.13	20.25	14.62	9.53
Gain %	8	9	10	11	13	14	20	42	43
t^c (s)	0.13	0.13	0.13	0.14	0.14	0.14	0.15	1.15	1.15
t^c (s)	4.3	4.5	5.0	5.4	5.5	6.2	8.3	10.22	15.4
Δ a)	71	64	50	45	35	30	24	15	8.5
PRD_0 a)	0.750	0.675	0.640	0.550	0.484	0.400	0.300	0.230	0.150
CR_b	38.64	35.41	31.86	28.53	24.93	21.03	16.21	10.07	6.56
CR_b^{Huff}	42.56	39.20	35.65	32.10	28.37	24.32	19.60	14.32	9.40
Gain %	10	11	12	13	14	16	21	42	43
t^c (s)	0.11	0.11	0.11	0.11	0.11	0.11	0.12	0.12	0.12
t^c (s)	4.1	4.0	4.5	5.2	5.4	6.7	8.1	10.7	15.5
Δ b)	92	81	69	58	47	36	25	15.5	8.5
CR_{RL}	26.63	24.41	22.42	20.34	18.14	15,68	12.50	8.32	5.61
CR_{RL}^{Huff}	35.06	31.78	28.80	25.93	22.91	19.66	15.85	11.63	7.82
Gain %	32	30	28	27	26	25	26	40	40
t^c (s)	0.12	0.12	0.12	0.13	0.13	0.13	0.13	0.14	0.16
t^c (s)	4.5	4.9	6.2	6.4	7.1	7.4	9.0	12.8	19.5

Fonte: Adaptado de REBOLLO-NEIRA (2019)

A abordagem baseada na seleção dos maiores coeficientes wavelet mostrou-se mais flexível para diferentes PRD, enquanto a organização proposta manteve alta eficiência e simplificou o processo. Comparado a métodos tradicionais como a codificação Run-Length, a estratégia foi superior em compactação e qualidade de recuperação, sendo ideal para aplicações de monitoramento remoto que exigem alta compressão e baixa distorção, podendo ser visualizado na figura 3.2.

3.3 ECG Signal Compression Using Different Techniques

No artigo (RANJEET; KUMAR; PANDEY, 2011), são analisadas as etapas de limiarização, quantização e codificação Huffman aplicadas a três métodos de transformação: DWT,

FFT e DCT. O objetivo é identificar o método mais eficiente para a compressão de sinais de ECG, considerando tanto a taxa de compactação quanto a preservação da qualidade do sinal reconstruído.

A metodologia empregada no artigo utiliza registros do banco de dados MIT-BIH Arrhythmia Database. Após a aplicação das transformadas DWT, FFT e DCT, o processo de compressão é conduzido conforme as etapas descritas na Figura 2.5, detalhadas a seguir.

A etapa de limiarização é realizada para reduzir o número de coeficientes a serem processados, descartando aqueles que apresentam valores insignificantes. No artigo, utiliza-se um limiar global, no qual o valor limite é definido manualmente com base nos coeficientes transformados. Este limite é escolhido dentro do intervalo $(0, \text{xmax}_j)$, onde xmax_j representa o coeficiente máximo da transformação aplicada. Seja T o valor do limiar escolhido e c_i os coeficientes transformados. A regra de limiarização é expressa como:

$$\text{Se } |c_i| < T, \text{ então } c_i = 0. \tag{3.1}$$

Os coeficientes restantes, com $|c_i| \geq T$, são considerados relevantes e são mantidos para as etapas subsequentes.

Essa abordagem garante que a maior parte da energia do sinal seja preservada, ao mesmo tempo em que reduz significativamente o número de coeficientes processados, otimizando o desempenho da compressão.

Além disso, o quantizador uniforme é empregado nesses coeficientes. No processo de quantização, os coeficientes wavelet são quantizados usando um tamanho de passo uniforme. O cálculo do tamanho do passo (Δ) depende de três parâmetros: valores máximos (M_{max}) e mínimos (M_{min}) na matriz de coeficientes e o número de níveis de quantização (L). O tamanho do passo é definido como:

$$\Delta = \frac{M_{\text{max}} - M_{\text{min}}}{L}. \tag{3.2}$$

A entrada é então dividida em $L+1$ níveis com tamanhos de intervalo iguais, variando de M_{min} a M_{max}, para construir a tabela de quantização. Durante a etapa de reconstrução, esses três parâmetros (M_{max}, M_{min}, L) são armazenados em um arquivo, pois são necessários para recriar a tabela de quantização.

Os valores quantizados são então compactados utilizando a codificação Huffman. Este método atribui códigos binários de comprimento variável aos coeficientes, com base em sua frequência de ocorrência como discutido na Seção 2.7.1.

A análise demonstra que a DWT oferece um melhor equilíbrio entre taxa de compressão e fidelidade do sinal reconstruído, devido à sua capacidade de representar características localizadas do sinal, enquanto as transformadas FFT e DCT apresentam maior eficiência de compactação o que as torna mais adequadas para cenários em que a compactação máxima é prioritária. A utilização de um limiar global definido manualmente, combinado com a quantização uniforme baseada em tamanho de passo e codificação Huffman, revelou-se eficaz na redução de

dados, preservando informações clínicas essenciais. A tabela a seguir apresenta as métricas de taxa de compressão e qualidade do sinal reconstruído (PRD e SNR) para cada método.

Tabela 3.2: Métricas de compressão para os métodos analisados.

Método	CR	PRD (%)	SNR (dB)	MSE
DWT (db7)	3,86	9,55	20,43	0,000547
FFT	5,31	9,34	20,62	0,000532
DCT	6,58	9,53	20,45	0,000545

3.4 Comparativo entre os trabalhos relacionados

Tabela 3.3: Comparação dos trabalhos relacionados com o EdgeECG

Critério	**3.1**	**3.2**	**3.3**	**EdgeECG**
Transformada	Sim (DWT)	Sim (DWT)	Sim (DWT, FFT e DCT)	Sim (DWT)
Quantização/ Limiarização	Sim (quantização uniforme)	Sim (quantização uniforme)	Sim (quantização uniforme e limiarização global)	Não (quantização uniforme e Soft-Threshold não aplicados no sistema)
Código de compressão	Sim (codificação de entropia Huffman)	Não (utiliza formato HDF5 em vez de codificação de entropia)	Sim (codificação SPIHT para compressão)	Sim (codificação de entropia Huffman)
Transmissão de ECG comprimido para nuvem	Sim (projeto otimizado para transmissão via rede em telemedicina)	Não (Testes realizados no ambiente MATLAB)	Não	Sim (Transmissão parcial do código)
Registro Eletrônico de Saúde padronizado	Não	Não	Não	FHIR
Processamento em dispositivo físico	Não	Não	Não	Sim

Seis critérios foram avaliados com o objetivo de posicionar o EdgeECG em relação a outros trabalhos voltados à compressão de dados biomédicos utilizando transformadas. A partir da Tabela 3.3, observa-se que todos os trabalhos analisados utilizam a Transformada Wavelet como, pelo menos, uma das técnicas aplicadas no processo de compressão.

Quanto às etapas de quantização e limiarização, destaca-se que apenas o artigo mencionado na Seção 3.3 incorpora a etapa de limiarização em sua metodologia. A tabela também compara os tipos de codificação utilizados, evidenciando que o trabalho descrito na Seção 3.2 se diferencia por empregar um formato de armazenamento e gerenciamento de dados, em vez de um algoritmo de compressão convencional.

No quesito transmissão de ECG comprimido para a nuvem, além do EdgeECG, que realiza as etapas de processamento no microcontrolador STM32F411 e transmite os dados via um dispositivo ESP acoplado ao MCU, apenas o artigo sumarizado na Seção 3.1 aborda a viabilidade da transmissão. Contudo, esse artigo não detalha os mecanismos que garantem a eficiência ou apresenta evidências claras de desempenho.

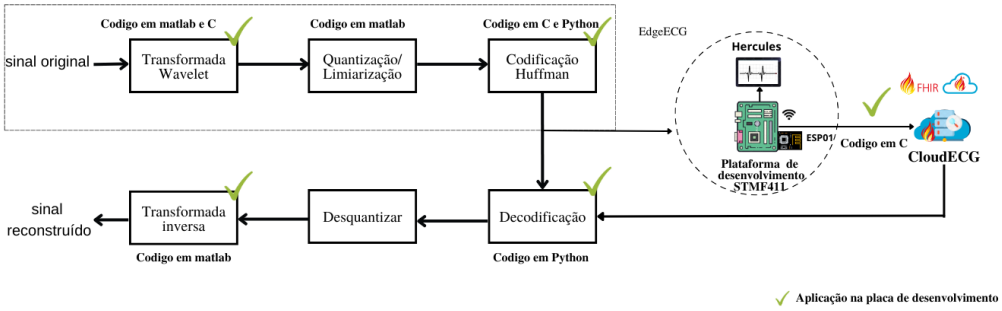
Por fim, dois parâmetros são exclusivos da metodologia proposta neste trabalho: a padronização dos registros segundo o modelo FHIR e o processamento diretamente no MCU, o que

demonstra a viabilidade prática da aplicação dos métodos desenvolvidos.

4 METODOLOGIA

A Figura 4.1 apresenta uma visão geral das etapas de compressão e descompressão do sinal de ECG consideradas neste trabalho. O fluxo é dividido em três etapas principais de compressão, todas associadas ao dispositivo IoT. Os símbolos de *check* destacam que o EdgeECG implementa as etapas de transformada (seção 4.1) e codificação de entropia Huffman (seção 4.3), com suas respectivas linguagens de programação. É importante ressaltar que as etapas que não possuem o *check* descritas nas seções 2.7 e 2.6 não estão implementadas no desenvolvimento do microcontrolador (MCU) que possui a capacidade de transmissão eficiente dos dados comprimidos para a nuvem, sendo este o grande diferencial deste trabalho. Essa abordagem integra processamento local e conectividade, otimizando o uso de recursos e a acessibilidade das informações.

Figura 4.1: Sistema EdgeECG



Fonte: Autoria Própria

4.1 DWT e Daubechies 2

O sistema de compressão desenvolvido neste trabalho é baseado na implementação do algoritmo DWT, que permite a análise e compressão do sinal no domínio da frequência. Para sua execução, foi necessário desenvolver um código compatível com a linguagem de programação utilizada na placa de desenvolvimento mencionada na seção 5.1.2. Dada as limitações de processamento da placa, especialmente em tarefas complexas como as associadas à Transformada Wavelet, adotou-se uma abordagem em dois ambientes distintos.

Inicialmente, os desafios dessa aplicação foram abordados no ambiente MATLAB, onde a lógica foi desenvolvida e validada. Posteriormente, o código foi adaptado para a linguagem C e modificado para operar no MCU. Após testes em compiladores online, o código ajustado demonstrou desempenho satisfatório no dispositivo. Assim, a etapa de aplicação da transformada wavelet foi organizada em dois códigos complementares, conforme ilustrado na Figura 4.1. A mesma lógica foi seguida para a etapa de codificação Huffman.

Dessa forma, o desenvolvimento do sistema EdgeECG envolveu o uso de três linguagens de programação distintas (MATLAB, C e Python), permitindo uma implementação eficiente e

adaptada às necessidades e restrições do dispositivo.

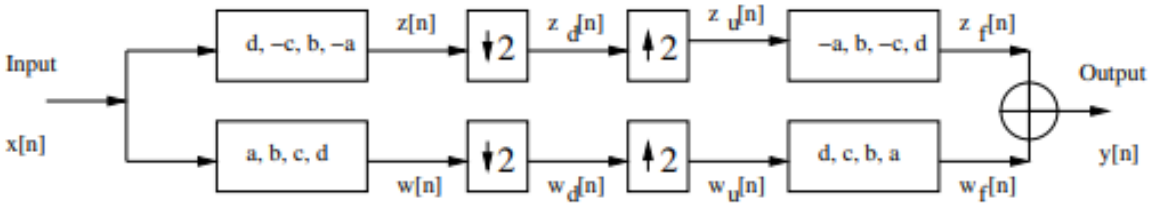
O primeiro código foi implementado no MATLAB, devido as ferramentas robustas de processamento de sinais e funções prontas para realizar a decomposição do sinal com a Transformada Wavelet. Esse ambiente serviu como referência para validar e comprovar os resultados. Já o segundo código foi adaptado para o microcontrolador, permitindo a execução eficiente do processamento diretamente na placa. Dessa forma, os resultados obtidos no MATLAB foram utilizados como parâmetro para avaliar a precisão e o desempenho do código implementado no microcontrolador, garantindo a integridade dos dados processados.

A DWT realiza uma decomposição do sinal em sub-bandas utilizando filtros passa-alta e passa-baixa, permitindo a extração de informações detalhadas em diferentes níveis de resolução (WEEKS, 2007). Como o sinal analisado é unidimensional uma Oitava pode ser definida como um nível de resolução, onde cada oitava pode ser imaginada como um par de filtros FIR. Um dos filtros é um passa-baixas (LPF), que utiliza a sequencia de coeficientes (d, -c, b, -a), enquanto o outro é um filtro passa-altas (HPF), com a sequencia de coeficientes (a, b, c, d) Figura 4.2, sendo que para decomposição desse sinal o par de filtros é definido como analise, já o par de síntese (transformada wavelet inversa), consistindo de um filtro passa-baixas inverso (ILPF) e um filtro passa-altas inverso (IHPF). Cada filtro de análise tem um down-sampler depois dele, para tornar a transformada eficiente que por sua vez é precedido por um up-sampler pra garantir a reconstrução do sinal.

> O passa-baixa produz o sinal de aproximação, enquanto um filtro passa-alta produz o sinal de detalhe. Por exemplo, um filtro passa-baixa simples pode ter coeficientes $\{\frac{1}{2}, \frac{1}{2}\}$, produzindo saídas da forma $(x[n] + x[n-1])/2$, que correspondem claramente à média de duas amostras. Um filtro passa-alta simples correspondente teria coeficientes $\{\frac{1}{2}, -\frac{1}{2}\}$, produzindo saídas da forma $(x[n] - x[n-1])/2$, ou seja, metade da diferença das amostras. Enquanto o sinal de média se assemelha muito ao original, os detalhes são necessários para que o sinal reconstruído corresponda ao sinal original. A análise multirresolução alimenta o sinal de média em outro conjunto de filtros, que produz os sinais de média e detalhe na próxima oitava. Os sinais de detalhe são mantidos, mas as médias das oitavas superiores podem ser descartadas, pois podem ser recalculadas durante o processo de síntese. As saídas de cada oitava contêm apenas metade da quantidade de dados da entrada (mais alguns coeficientes adicionais devido ao filtro). Assim, a representação wavelet possui aproximadamente o mesmo tamanho que o sinal original. WEEKS (2007)

A família de wavelets Daubechies, nomeada em homenagem à matemática belga Ingrid Daubechies, é amplamente utilizada no processamento de sinais e análise multirresolução por demonstrar eficiência em propriedades matemáticas robustas. Na aplicação do código de compressão foi utilizado a wavelet db2, sendo essa capaz de representar polinômios de grau até 1 (constantes e lineares) sem erro. Essa propriedade torna a db2 particularmente útil na com-

Figura 4.2: Operação de down e up-sampler



Fonte: Adaptado de WEEKS (2007)

pressão de sinais e remoção de ruídos, pois permite capturar informações de baixa frequência de forma eficaz enquanto detalha mudanças rápidas no sinal. Os filtros utilizados foram projetados com base nos coeficientes específicos da wavelet Daubechies de ordem 2, garantindo propriedades como ortogonalidade e compacidade no suporte. Desta forma, a aplicação direta da WT no MATLAB pode ser realizada como descrito no Código fonte 4.1.

Código-fonte 4.1: Trecho da implementação da wavelet em MATLAB

```
1 %% Sample MATLAB code snippet
2 load('100m');
3 xn=val(2,:)/200;
4
5 [J,l] = wavedec(xn,3,'db2');
6 approx = appcoef(J,l,'db2');
7 [cd1,cd2,cd3] = detcoef(J,l,[1 2 3]);
8
9 x = waverec(J,l,'db2');
```

Fonte: Autoria Própria

No MATLAB, a wavelet db2 pode ser utilizada por meio de funções como `wavedec` para a decomposição de sinais, e `waverec` para a reconstrução (Código-fonte 4.1). Essas funções facilitam a aplicação da Transformada Wavelet Discreta em diversas áreas, como compressão de dados, análise de séries temporais e processamento de imagens. O MATLAB também fornece ferramentas para visualizar a wavelet db2 e seus filtros associados, permitindo uma análise detalhada de suas propriedades matemáticas e aplicações práticas.

No entanto, para desenvolver o código e MATLAB com maior similaridade possível para que o mesmo fosse compatível a linguagem de programação do microcontrolador, funções prontas como Waverec [1] foram formuladas de maneira simplificada, através de laços de for que implementam a convolução do sianal. Para isso toda base de cálculo foi desenvolvida seguindo a lógica dos exemplos demonstrados em WEEKS (2007, pg.314). O trecho de código 4.2 exemplifica, por meio da função `calculate_filterl`, como foi possível implementar as

[1] https://www.mathworks.com/help/wavelet/ref/waverec.html

Código-fonte 4.2: Trecho da implementação da wavelet em C no uControlador

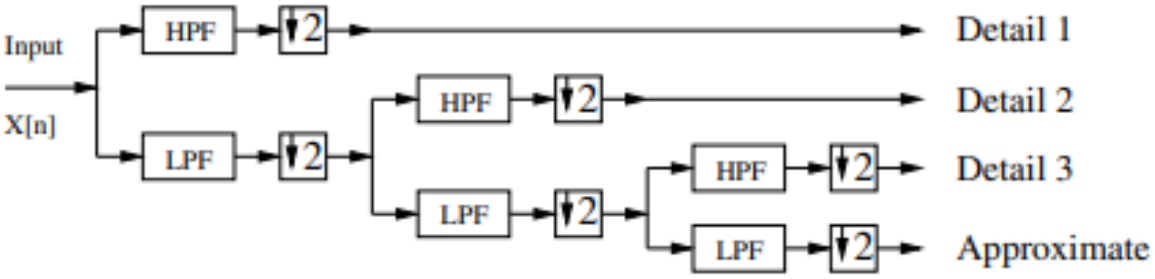
```
void calculate_filterl(int length, float *xn, float *wn, float a, float b, float c, float d,
float *wd, float *w2, float *w2d,float *w3, float *w3d) {
        int cont = 0;

    for (int i = 0; i < length; i++) {
        if (i == 0) {
            wn[i] = a * xn[i];
        } else if (i == 1) {
            wn[i] = a * xn[i] + b * xn[i - 1];
        } else if (i == 2) {
            wn[i] = a * xn[i] + b * xn[i - 1] + c * xn[i - 2];
        } else {
            wn[i] = a * xn[i] + b * xn[i - 1] + c * xn[i - 2] + d * xn[i - 3];
        }
    }
    for (int i = 0; i < length; i++) {
            if (i % 2 != 0) {
                wd[cont] = wn[i];
                cont++;
            }
    }
```

Fonte: Autoria Própria

decomposições do sinal de entrada *xn*. Essa implementação foi realizada ao convoluir o sinal com o filtro LPF, seguida pela aplicação de um *down-sampler*, sem utilizar funções prontas, como as mostradas no código 4.1.

Essa abordagem foi utilizada para realizar decomposições até o terceiro nível, conforme ilustrado na Figura 4.3. O Código-fonte 4.2 realiza convoluções sucessivas até que os parâmetros de detalhe (zd, $z2d$ e $z3d$) e de aproximação ($w3d$) sejam calculados. Com isso, foi possível comparar esses componentes com os gerados pela função `detcoef` no código 4.1, que extrai os parâmetros de detalhe ($cd1$, $cd2$ e $cd3$) e de aproximação ($approx$).

Figura 4.3: Operação de Analise



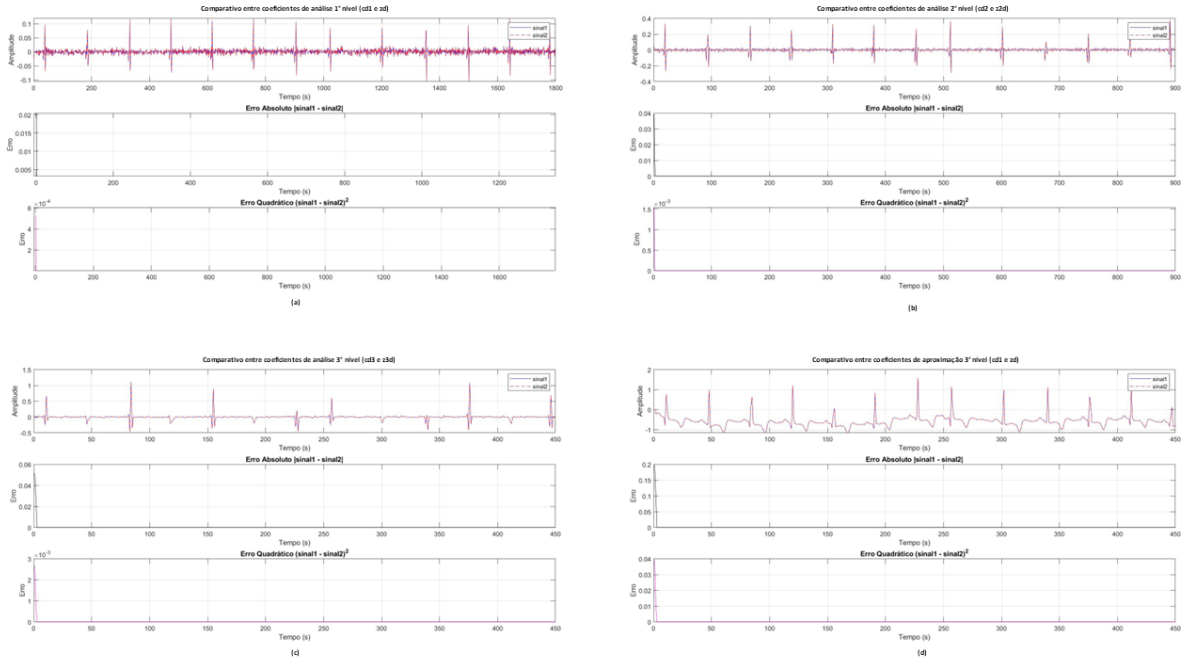
Fonte: Adaptado de WEEKS (2007)

A Figura 4.4 apresenta a comparação entre os sinais obtidos com as funções adaptadas para implementação em C e aqueles gerados pela função pronta do MATLAB. Na imagem, ambos os sinais são plotados em um gráfico, juntamente com a representação de duas métricas: o erro absoluto e o erro quadrático.

O erro absoluto mede a diferença absoluta ponto a ponto entre os sinais, proporcionando uma análise direta e robusta sem amplificar discrepâncias maiores. Já o erro quadrático, por sua

vez, eleva as diferenças ao quadrado, destacando discrepâncias mais significativas, o que é útil para identificar variações mais críticas. Além disso, os sinais foram analisados detalhadamente, identificando as posições onde apresentam diferenças e calculando o PRD (Percentual de Diferença Quadrática Média) entre os componentes correspondentes. Esse parâmetro fornece uma avaliação quantitativa precisa da qualidade da implementação e da fidelidade entre os sinais.

Figura 4.4: Comparação das decomposições até o terceiro nível entre a convolução manual e a função wavedec



Fonte: Autoria Própria

No primeiro código desenvolvido em MATLAB, a forma com que foi implementada a convolução não é efetiva quanto a reconstrução do sinal. Esse fator se dá ao fato de que o tamanho do vetor que armazena os dados entre convolução, não foi adequada conforme a documentação [2]. A convolução de dois vetores u e v representa a área de sobreposição sob os pontos à medida que v desliza sobre u. Algebraicamente, a convolução é a mesma operação que multiplicar polinômios cujos coeficientes são os elementos de u e v.

Se $m = \text{length}(u)$ e $n = \text{length}(v)$, então o vetor w, de comprimento $m+n-1$, tem seu k-ésimo elemento dado por:

$$w(k) = \sum_{j} u(j) \cdot v(k-j+1), \tag{4.1}$$

onde a soma é realizada sobre todos os valores de j que resultam em índices válidos para $u(j)$ e $v(k-j+1)$. Especificamente:

$$j = \max(1, k+1-n) : \min(k, m). \tag{4.2}$$

[2] https://www.mathworks.com/help/MATLAB/ref/conv.html

Quando $m = n$, temos:

$$
\begin{aligned}
w(1) &= u(1) \cdot v(1), \\
w(2) &= u(1) \cdot v(2) + u(2) \cdot v(1), \\
w(3) &= u(1) \cdot v(3) + u(2) \cdot v(2) + u(3) \cdot v(1), \\
&\vdots \\
w(n) &= u(1) \cdot v(n) + u(2) \cdot v(n-1) + \cdots + u(n) \cdot v(1), \\
&\vdots \\
w(2n-1) &= u(n) \cdot v(n).
\end{aligned}
$$

Além disso, devido à forma como o loop for foi declarado, o Código-Fonte 4.2 não trata adequadamente índices fora do intervalo válido do sinal (problema ajustado na linha 5 do Código-Fonte 4.3). Outro fator que compromete a reconstrução correta é o cálculo explícito de cada índice, sem realizar a acumulação generalizada necessária para a convolução. Em contrapartida, a segunda versão do Código-Fonte 4.1, implementada em MATLAB, adapta uma abordagem originalmente desenvolvida para sinais 2D, corrigindo os problemas mencionados. Essa versão se mostra mais eficiente na reconstrução do sinal, apresentando maior precisão e robustez [3].

Código-fonte 4.3: Trecho da implementação da wavelet em MATLAB (Código corrigido)

```
1
2 % Conv. manual
3 for i = 1:length(wn)
4     for j = 1:m
5         if (i - j + 1 > 0) && (i - j + 1 <= n)
6             wn(i) = wn(i) + lp(j) * xn(i - j + 1);
7         end
8     end
9 end
```

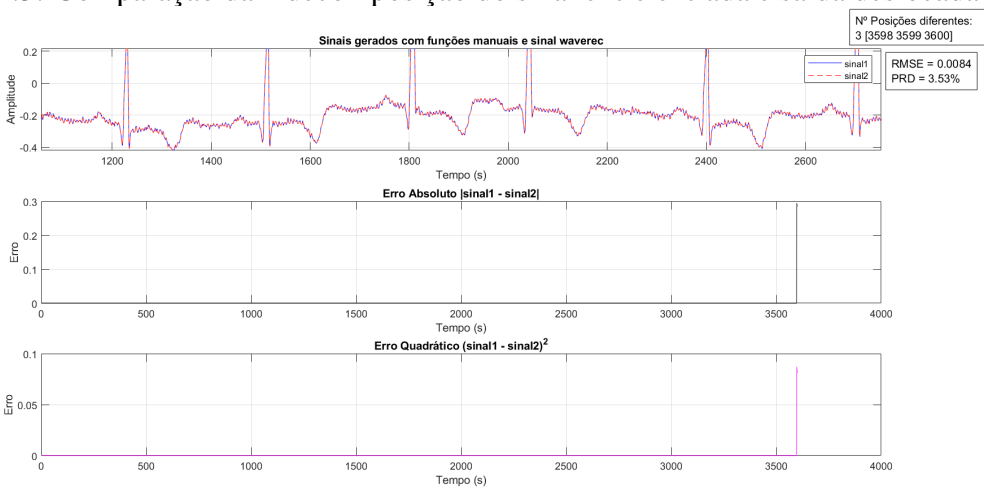
Fonte: Autoria Própria

Visto isso é demonstrado que para o código implementado na máquina, ao realizar a primeira decomposição do sinal 4.5 o mesmo apresenta pouca distorção comparado a reconstrução no último nível de decomposição 4.6, isso se dá pois operações de convolução, multiplicação e soma acumulam erros de precisão inerentes ao uso de ponto flutuante em cálculos computacionais e ao fato de que a operação de convolução insere um atraso na representação do sinal como demonstrado na 4.7. Esses erros tornam-se mais pronunciados à medida que o número de

[3] https://www.youtube.com/watch?v=1UGHvbs4hS4

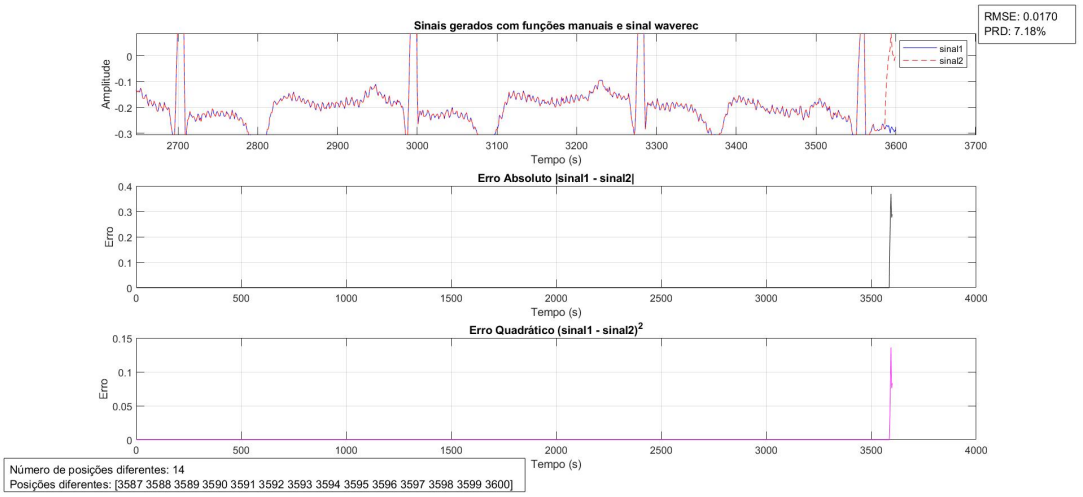
níveis de decomposição aumenta, devido à repetição das operações em cada nível (PROAKIS; MANOLAKIS, 2006).

Figura 4.5: Comparação da 1 decomposição do sinal entre entrada e saída deslocada



Fonte: Autoral

Figura 4.6: Comparação da 3 decomposição do sinal entre entrada e saída deslocada

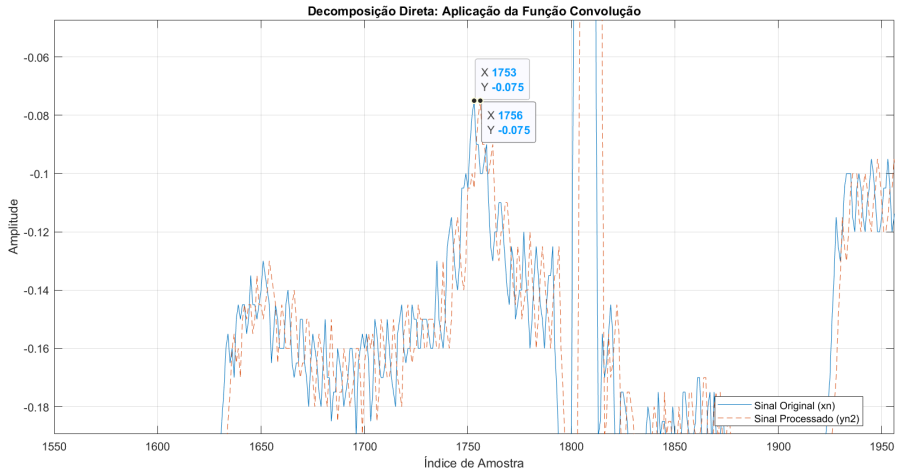


Fonte: Autoral

Os resultados apresentados na Figura referem-se ao primeiro código desenvolvido no MATLAB, posteriormente adaptado para execução no MCU (4.2). No contexto da reconstrução do sinal, após o processamento com funções manuais, foi necessário concatenar os coeficientes de aproximação e detalhes em um único vetor J para viabilizar a reconstrução do sinal utilizando a função `waverec`, conforme demonstrado no Código-fonte 4.1.

Devido às especificidades de implementação discutidas neste capítulo, o tamanho dos vetores de aproximação e detalhe foi ajustado ao final do processo. Esse ajuste se deu para compensar o excedente gerado pela convolução do sinal, preenchendo as posições restantes do vetor com zeros. Essa abordagem garantiu a comparabilidade entre os sinais reconstruídos utilizando as funções prontas e aqueles obtidos com as funções desenvolvidas manualmente.

Figura 4.7: Atraso gerado na 1° decomposição do sinal entre entrada e saída



Fonte: Autoral

Esse ajuste, contudo, torna-se desnecessário ao implementar o trecho de código apresentado em 4.3, que aprimora significativamente a eficiência do método. Com três níveis de decomposição, a implementação proposta demonstra melhorias consideráveis nos índices de PRD e RMSE, conforme os resultados abaixo:

- Código 4.2: PRD = 7,18% RMSE = 0,0170
- Código 4.3: PRD = 2,32% RMSE = 0,0055

Esses valores evidenciam a eficácia do ajuste proposto em 4.3, reduzindo consideravelmente os erros de reconstrução e garantindo maior precisão na preservação das características do sinal comprimido.

Para comparar os dados entre o sinal coletado na API FASS-ECG e o sinal obtido pela decomposição implementada no MATLAB, utiliza-se uma função que identifica as posições no vetor de dados que não correspondem ao vetor de referência. Essa abordagem permite verificar se o processamento das funções manuais no microcontrolador está consistente com as implementações realizadas no MATLAB. Após essa comparação, outras métricas são calculadas para avaliar o desempenho da compressão dos dados. Contudo, essas métricas são desenvolvidas exclusivamente nos códigos no MATLAB.

4.2 Quantização e Limiarização das sub-bandas

O desenvolvimento desse código utiliza apenas os teste no MATLAB como métodos de análise de compressão do sinal. Para isso, é realizada a decomposição do sinal utilizando a transformada wavelet discreta (DWT) com a wavelet db2. A decomposição é realizada em três níveis, retornando os coeficientes da decomposição (“J”) e os limites de nível (“l”). A partir desses coeficientes, extraem-se os coeficientes de aproximação e os coeficientes de detalhe para os três níveis.

4.2.1 Metodologia de Quantização Uniforme

A quantização uniforme implementada no código foi inspirada na abordagem descrita no artigo de CHOUAKRI; DJAAFRI; TALEB-AHMED (2013), que utiliza essa técnica para reduzir a amplitude e a precisão dos coeficientes wavelet, facilitando a compressão e preservando a estrutura do sinal. Para a implementação do código uma sequência lógica foi seguida:

1. Identificação do Intervalo dos Coeficientes: Para cada conjunto de coeficientes wavelet (detalhes nos níveis $cd1$, $cd2$, $cd3$), foi determinado o intervalo $[\text{mínimo}, \text{máximo}]$, correspondente aos valores extremos de cada conjunto.
2. Definição do Passo de Quantização (Δ):O intervalo total dos coeficientes é dividido em Q níveis uniformemente espaçados:

$$\Delta = \frac{\text{máximo} - \text{mínimo}}{Q} \tag{4.3}$$

   No código, $Q = 16$, o que significa que os coeficientes são discretizados em 16 níveis.
3. Arredondamento para Níveis Discretos: Cada coeficiente foi normalizado pelo passo Δ, arredondado para o inteiro mais próximo, e escalado de volta para o espaço original:

$$x_{\text{quant}} = \text{round}\left(\frac{x}{\Delta}\right) \cdot \Delta \tag{4.4}$$

   Esse processo reduz a precisão dos coeficientes ao restringi-los a valores discretos.
4. Redução de Amplitude: Os coeficientes menores em magnitude tornam-se mais próximos de zero, contribuindo para a compactação sem distorcer significativamente os coeficientes mais importantes.

O trabalho de CHOUAKRI; DJAAFRI; TALEB-AHMED (2013) enfatiza a importância de uma quantização eficiente antes da codificação de entropia. A quantização uniforme, ao reduzir os valores para níveis discretos, diminui a entropia dos dados, tornando-os mais fáceis de codificar e compactar, especialmente em aplicações de telemedicina.

4.2.2 Soft- Threshold

Para reduzir o impacto de ruídos e preservar informações significativas, é aplicado um limiar adaptativo aos coeficientes de detalhe após a quantização dos mesmos. Em [4], é mencionado que "uma boa escolha assintótica para o parâmetro de limiar é o limiar universal clássico definido como:

$$\lambda = \sigma\sqrt{2\log(N)} \tag{4.5}$$

[4] `https://verso.mat.uam.es/~fernando.chamizo/dark/d_denoisingw.html`

onde σ é o desvio padrão do ruído e N é o número de amostras.

A regra de soft-thresholding é utilizada para ajustar os coeficientes de detalhe. Essa regra funciona da seguinte forma: se o módulo de um coeficiente for maior que o limiar, ele é reduzido pelo valor do limiar, mantendo seu sinal. Caso contrário, o coeficiente é zerado. Essa abordagem é aplicada individualmente aos coeficientes de detalhe de cada nível.

Após o ajuste dos coeficientes de detalhe, os coeficientes suavizados substituem os originais na matriz "J". Em seguida, a função waverec é utilizada para reconstruir o sinal a partir dos coeficientes ajustados. O sinal reconstruído é então comparado ao sinal original, permitindo a visualização dos efeitos da aplicação do limiar adaptativo.

Para avaliar a eficácia do método, são calculadas várias métricas quantitativas. A taxa de compressão é definida como a razão entre o número total de coeficientes e os coeficientes diferentes de zero. A razão de compressão de energia PRD mede a distorção do sinal reconstruído em relação ao original. Além disso, a energia retida é calculada como o percentual de energia preservada após a aplicação do limiar. Por fim, o erro médio quadrático (RMSE) é utilizado para quantificar a diferença entre o sinal reconstruído e o original.

O desvio padrão é uma métrica que indica a dispersão ou variabilidade dos valores de um conjunto em relação à sua média. Na aplicação do código o desvio padrão pode ser calculada com a função `std(cd)`.

Matematicamente, o desvio padrão para um conjunto de valores $x_1, x_2, \ldots, x_n$ pode ser definido de duas formas:

- Desvio padrão populacional:

$$\text{std}(x) = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - \mu)^2} \tag{4.6}$$

- Desvio padrão amostral (padrão no MATLAB):

$$\text{std}(x) = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(x_i - \mu)^2} \tag{4.7}$$

Onde:

- n é o número total de elementos no vetor;
- x_i são os valores individuais no vetor;
- μ é a média dos valores, definida como:

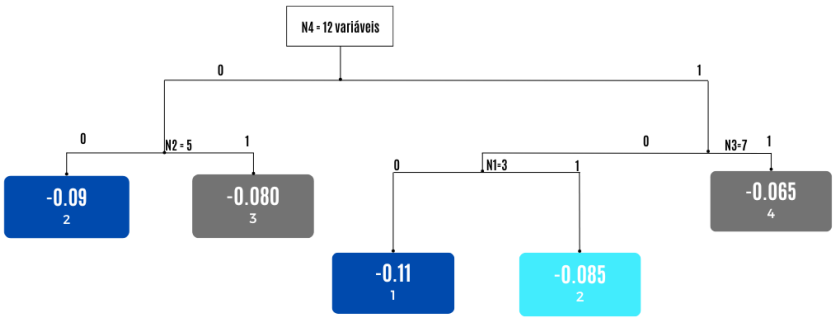
$$\mu = \frac{1}{n}\sum_{i=1}^{n} x_i \tag{4.8}$$

4.3 Código Huffman

Para a implementação do código de Huffman, foi desenvolvido um programa adaptado da lógica e trechos de codigo disponiveis na internet (SIDHWA, 2023; CODING, 2024). As alterações principais incluíram adaptações para processar um vetor de entrada com tipos de dados diferentes e calcular a frequência dos números presentes nesse vetor. Além disso, foi implementada uma funcionalidade para comparar a compressão do vetor de entrada em relação à string de dados binários gerada pela codificação.

Outra modificação significativa foi a substituição e adaptação da função pronta de *heap*, permitindo maior controle sobre os critérios de organização, especialmente em casos de empate de frequência, garantindo consistência no processo de construção da árvore de Huffman. As Figuras 4.8 e 4.9 evidenciam o problema em questão ao mostrar duas arvores de Huffman diferentes para o mesmo conjunto de valores (*[-0.065, -0.08, -0.085, -0.09, -0.065, -0.08, -0.065, -0.11, -0.065, -0.08, -0.085, -0.09]*). O primeiro código em Python utilizava a biblioteca `heapq` do Python (CODING, 2024), que segue um critério específico de organização do heap, que é realizado pela ordenação, pela frequência e, em caso de empate, manutenção da ordem de inserção. Devida a problemas encontrados para implementar essa função na placa, um segundo código foi desenvolvido com o código da *heap* implementado manualmente em Python e em C, que não considerava o critério de desempate pela ordem de inserção, o que resultava em árvores de Huffman diferentes e, consequentemente, em códigos distintos.

Figura 4.8: Arvore Huffman gerado com funções da biblioteca `heapq` do Python (CODING, 2024)

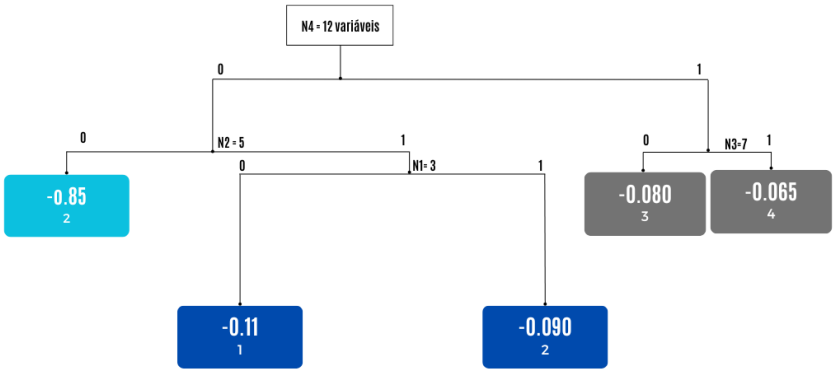


Fonte: Autoria própria

Essa inconsistência surgia porque, durante a construção da árvore de Huffman, a seleção dos dois menores nós (com mesma frequência) era feita de forma arbitrária no, enquanto no primeiro código a ordem de inserção era respeitada. Isso alterava a estrutura da árvore, mudando a ordem dos filhos e gerando códigos Huffman incompatíveis.

Para resolver isso, foi necessário modificar a lógica do heap manual, tanto em Python quanto em C. No heap manual, foi introduzido um critério de desempate explícito que prioriza a ordem de inserção dos nós com a mesma frequência. Essa modificação foi implementada

Figura 4.9: Arvore Huffman gerada sem o auxilio das funções de bibliotecas prontas do do Python



Fonte: Autoria própria

adicionando uma estrutura auxiliar, o array ‘*insertion_order*‘, que rastreia a ordem em que os nós foram adicionados ao heap. Com base nesse critério, a função ‘compareNodes‘ foi ajustada para garantir que, em caso de empate na frequência, o nó inserido primeiro fosse considerado menor.

Essa alteração foi crucial porque a estrutura da árvore de Huffman depende diretamente da sequência em que os nós são combinados. Com o critério de desempate alinhado ao comportamento do ‘heapq‘, a construção da árvore de Huffman passou a gerar a mesma estrutura que no primeiro código, garantindo compatibilidade entre os códigos gerados.

Além disso, foi implementada a função de decodificação mesmo no mcu para validar a compatibilidade entre os dois sistemas. Essa função percorre a árvore de Huffman a partir da raiz, utilizando a sequência codificada para navegar entre os nós filhos (esquerdo para ’0’ e direito para ’1’). Sempre que um nó folha é alcançado, o valor correspondente é adicionado à lista de decodificação.

Dessa forma foi possível realizar a codificação com um exemplo didático que comprovasse a integridade da aplicação da codificação Huffman em ambos os casos. Por fim, o código em C foi alterado para ser compatível no ambiente de desenvolvimento da STM32 gerando uma saída para o exemplo ditático cuja o vetor de entrada é definido como:

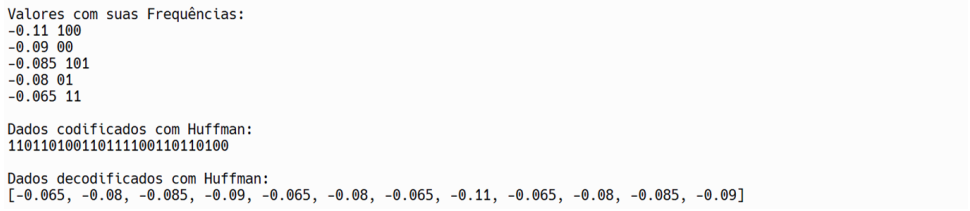
```
float originalValues[] = { -0.065, -0.080, -0.085, -0.090, -0.065,
-0.080, -0.065, -0.110, -0.065, -0.080, -0.085, -0.090 };
```

que resultou em uma codificação exata para ambos os códigos aplicados de acordo com as Figuras 4.10, 4.11 e 4.12.

4.4 Métricas de desempenho

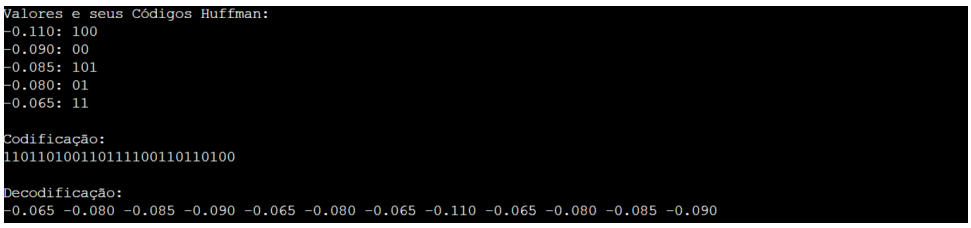
Para avaliar o desempenho do método proposto, foram implementadas algumas métricas de desempenho. Essas métricas são amplamente utilizadas no processamento de sinais e fornecem uma análise quantitativa da qualidade da reconstrução e da redução de armazenamento.

Figura 4.10: Saída do código para o exemplo didático do primeiro código em Python



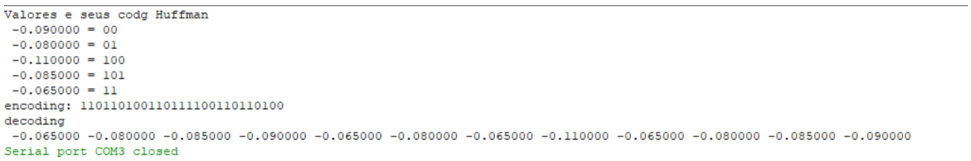
Fonte: Autoria própria

Figura 4.11: Saída do código para o exemplo didático do código em Python modificado



Fonte: Autoria própria

Figura 4.12: Saída do código para o exemplo didático do código Aplicado no mcu



Fonte: Autoria própria

A Taxa de Compressão foi utilizada para medir a eficiência do método de compressão. Segundo CAMPITELLI (2015), a taxa de compressão é definida como:

$$CR = \frac{\text{Tamanho do Sinal Original}}{\text{Tamanho do Sinal Comprimido}}, \tag{4.9}$$

onde valores elevados de CR indicam maior eficiência de compressão, proporcionando menor armazenamento ou custo de transmissão.

A fidelidade da reconstrução foi avaliada com a métrica Percentage Root-Mean-Square Difference (PRD), que mede a diferença percentual entre o sinal original e o sinal reconstruído. De acordo com CAMPITELLI (2015), a métrica é definida como:

$$PRD = \sqrt{\frac{\sum_{n=1}^{N}(x[n]-\hat{x}[n])^2}{\sum_{n=1}^{N}x[n]^2}} \cdot 100, \tag{4.10}$$

onde $x[n]$ representa o sinal original, $\hat{x}[n]$ o sinal reconstruído e N o número de amostras. Valores baixos de PRD indicam alta fidelidade na reconstrução do sinal.

Outra métrica implementada foi a Energia Retida, que mede a proporção da energia total do sinal original preservada após a reconstrução. Baseando-se em OLIVEIRA; SANTOS; SOUZA

(2017), a fórmula utilizada é:

$$EnergiaRetida = \left(\frac{\sum_{n=1}^{N} |\hat{x}[n]|^2}{\sum_{n=1}^{N} |x[n]|^2} \right) \cdot 100 \tag{4.11}$$

onde:

$x[n]$ é o sinal original; $\hat{x}[n]$ é o sinal reconstruído; N é o número total de amostras do sinal.

Valores próximos a 100% indicam que a maior parte da energia do sinal foi mantida, o que reflete uma reconstrução eficiente.

A métrica Erro Quadrático Médio (Root-Mean-Square Error, RMSE) foi implementada para medir a magnitude média do erro entre o sinal original e o reconstruído. Como definido em CHOUAKRI; DJAAFRI; TALEB-AHMED (2013), a fórmula é:

$$RMSE = \sqrt{\frac{1}{N} \sum_{n=1}^{N} (x[n] - \hat{x}[n])^2}. \tag{4.12}$$

Essa métrica fornece uma medida absoluta da discrepância entre os sinais, útil para avaliar diferenças locais.

Todas essas métricas foram implementadas no ambiente MATLAB para avaliar quantitativamente os sinais processados. Os resultados são apresentados na seção de resultados, juntamente com uma discussão detalhada.

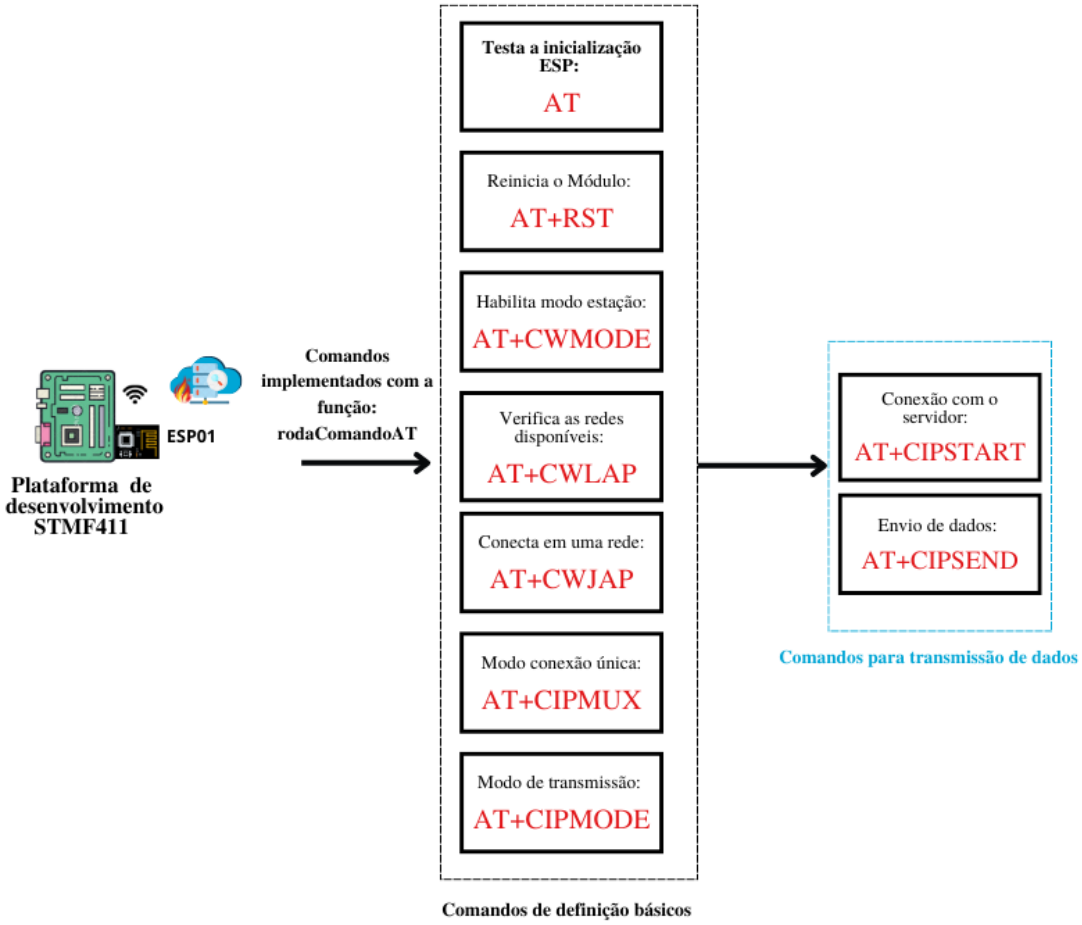
4.5 Registro e Transmissão do ECG para a Nuvem

Para exemplificar o funcionamento da transmissão do sinal de ECG através do processamento no MCU e envio do sinal pelo dispositivo ESP acoplado a placa de desenvolvimento, o conjunto de etapas do código desenvolvido pode ser dividido em 5 partes principais:

1. Comunicação com o ESP;
2. Transformada;
3. Huffman;
4. Registro FHIR;
5. Sequência de requisições FHIR.

Etapa 1: O código é implementado a partir das configurações iniciais do módulo Wi-Fi, seguindo uma sequência de comandos configurados pelo padrão AT (*Attention Command*). Esses comandos são amplamente utilizados para controlar o módulo Wi-Fi, configurar conexões de rede e realizar diversas outras funcionalidades. Os comandos AT formam uma interface baseada em texto que permite interagir com o módulo por meio de comandos simples enviados via comunicação serial.

Figura 4.13: Sequência de comandos AT implementados



Fonte: Autoria própria

A Figura 4.13 demonstra a sequência de comandos aplicados no sistema. Inicialmente, é testada a comunicação com o módulo, a fim de verificar possíveis erros, como a recepção de caracteres indevidos, frequentemente causados por configurações incorretas da taxa de transmissão. O comando `AT` é utilizado nessa etapa, retornando um "OK"caso tudo esteja funcionando corretamente.

Os dois primeiros comandos configuram a inicialização e o reinício do módulo, seguidos pelo comando que define o modo Wi-Fi. O ESP possui três modos Wi-Fi que podem ser configurados via comandos AT:

- Modo *SoftAP*: O ESP atua como um ponto de acesso, permitindo que outros dispositivos se conectem a ele.
- Modo Estação: O módulo conecta-se a uma rede Wi-Fi existente, permitindo acesso à internet.
- Modo *SoftAP+Estação*: Combina as funcionalidades de ponto de acesso e estação simultaneamente.

Para esta aplicação, é utilizado o modo Estação, configurado pelo comando `AT+CWMODE`. Em sequência, o comando `AT+CWLAP` é utilizado para listar as redes disponíveis, facilitando

o monitoramento de possíveis problemas de conexão. A Espressif disponibiliza um manual abrangente com todos os comandos AT e suas descrições detalhadas (Manual AT da Espressif).

Para conectar-se a uma rede local, utiliza-se o comando `AT+CWJAP`, que requer o SSID e a senha da rede. Na conexão com o servidor, é utilizado o modelo de comunicação cliente-servidor, em que um dispositivo baseado em Internet das Coisas (IoT) atua como cliente, enviando ou recebendo informações de um servidor. Esse processo é baseado no protocolo de comunicação TCP/IP, formado por uma pilha de protocolos organizados em quatro camadas, cada uma responsável por funções específicas para gerenciar a comunicação.

Seguindo a sequência de comandos, o `AT+CIPMUX` configura o módulo no modo de conexão única (*single connection*), onde apenas uma conexão TCP ou UDP é permitida por vez. Em seguida, o comando `AT+CIPMODE` define o modo de transmissão normal, no qual os dados são manipulados pelo firmware do módulo antes de serem enviados.

Para estabelecer uma conexão com a nuvem, utiliza-se o comando `AT+CIPSTART`, que requer os seguintes argumentos: o tipo de conexão (neste caso, *TCP*), o endereço IP ou domínio do servidor (convertido para um IP pelo DNS) e a porta de comunicação, que neste caso é a porta reservada para o protocolo HTTP. Após o sucesso desse comando, torna-se possível realizar requisições ao servidor.

Por fim, o comando `AT+CIPSEND` é usado para enviar dados ao servidor. Esse comando especifica o número de bytes que serão enviados, iniciando a transmissão de dados após uma confirmação do módulo. É um passo essencial para estabelecer comunicação bidirecional com o ESP, permitindo que o dispositivo envie informações ao servidor de maneira eficiente.

Etapa 2: Nesta etapa, os códigos discutidos na Seção 4.1 foram implementados por meio da criação de duas funções principais. A primeira função é responsável pelo cálculo dos coeficientes resultantes da aplicação do filtro passa-baixas como demonstrado no trecho de Código 4.2, enquanto a segunda realiza o cálculo dos coeficientes provenientes do filtro passa-altas. Ambas as funções possuem a funcionalidade de enviar, por meio do comunicador serial, os resultados de cada etapa da decomposição do sinal.

Após a obtenção dos coeficientes, as funções também realizam uma organização dos dados, permitindo a construção adequada do vetor J, descrito no Código-fonte 4.2. Esse vetor é de suma importância para a reconstrução do sinal no ambiente de simulação MATLAB, garantindo a consistência e a fidelidade dos resultados obtidos no processo de decomposição e reconstrução do sinal.

Etapa 3: Após a etapa anterior, é realizada a criação da árvore de Huffman para o sinal integral com base nos dados contidos no vetor J. Em seguida, o sinal é codificado utilizando essa árvore, e posteriormente decodificado, como forma de validar o funcionamento completo do algoritmo. Esse processo é detalhado na seção 4.3, que demonstra a eficiência do código de Huffman desenvolvido na placa de desenvolvimento, sendo esta avaliada por meio da comparação com implementações dos códigos semelhates, desenvolvidas em outras lingagens de programação e realizadas em compiladores online, comprovando o desempenho da solução

implementada ao gerar o mesmo binário codificado.

Etapa 4: Nesta etapa, com base nos conceitos e definições apresentados no Código-fonte 2.1, é criado um arquivo JSON contendo as informações básicas de uma requisição. Inicialmente, é gerado um POST com um dado do tipo `float`, definido como "0"no campo de dados. Posteriormente, são realizadas requisições do tipo PATCH, calculando-se a quantidade necessária para o envio completo do sinal que define a quantidade de dados a serem enviados por requisição. Esta etapa é importante devido ao fato de que o comando `AT+CIPSEND` tem uma limitação quanto ao tamanho de dados que podem ser enviados por requisição (2048 bytes).

Para isso, foi desenvolvida uma função genérica capaz de realizar requisições de diferentes tipos (POST, GET, PUT e PATCH), bastando alterar o parâmetro que define o tipo de requisição a ser enviado. Para otimizar o código e melhorar a reutilização, essa função foi implementada em uma biblioteca separada, dado que é chamada diversas vezes ao longo da implementação.

Etapa 5: Como última etapa, é realizada a chamada de uma função que monta a requisição no protocolo HTTP, utilizando o JSON construído na etapa anterior. Para isso, são enviados os seguintes parâmetros em destaque:

1. o método da requisição (POST, GET, PATCH, etc.), conforme abordado na Seção 2.8;
2. a rota da requisição, que no caso do POST inicial é `/baseR4/Observation`, enquanto para os PATCHs e PUT a rota é concatenada com o `ID` retornado pelo POST inicial. Esse `ID`, fornecido como resposta à requisição POST, permite que os PATCHs subsequentes sejam enviados para o mesmo endpoint, garantindo que o sinal seja reconstruído de forma integral na nuvem.
3. o tipo de dado (neste caso, um `jsonobservation`);
4. o tamanho do JSON e;
5. o conteúdo construído na Etapa 4.

Essa abordagem garante que o processo de envio seja consistente e eficiente, permitindo a integração dos dados no ambiente de nuvem.

5 RESULTADOS

Os experimentos realizados para a validação do método envolvem dois tipos de avaliação. O primeiro (seção 5.2) consiste na análise dos resultados obtidos com os códigos implementados em MATLAB para os algoritmos mencionados no Capítulo 4. O segundo (seção 5.3) valida a transmissão do vetor *J* após a aplicação das etapas ilustradas na Figura 4.1.

5.1 Setup do experimento

5.1.1 Banco de Dados - physionet

O PhysioNet é um repositório de dados médicos e fisiológicos, amplamente utilizado na pesquisa e desenvolvimento de algoritmos em áreas como bioengenharia, aprendizado de máquina e análise de sinais biomédicos. PhysioNet fornece acesso gratuito a uma ampla variedade de bancos de dados, incluindo os registros de sinais eletrocardiográficos (ECG) utilizados neste trabalho Essa plataforma é amplamente reconhecida por sua contribuição para o avanço da pesquisa em saúde digital e análise de dados clínicos (GOLDBERGER et al., 2000).Para realizar os experimentos explicitados, a plataforma oferece diversas opções de download. No MATLAB, a função `load` acessa diretamente o arquivo salvo. Já para as implementações em outras linguagens de programação e no ambiente de programação do MCU, o sinal foi declarado em um vetor.

5.1.2 Microcontrolador

Para viabilizar a transmissão de um sinal, é essencial utilizar uma plataforma de desenvolvimento capaz de converter os dados adquiridos e se comunicar com um dispositivo que utilize a internet como meio de envio para um servidor web. Além disso, para facilitar a visualização dos dados transmitidos, foi empregado um software (hercules) que exibe, em tempo real, as informações processadas pelo microcontrolador, através de comunicação serial.

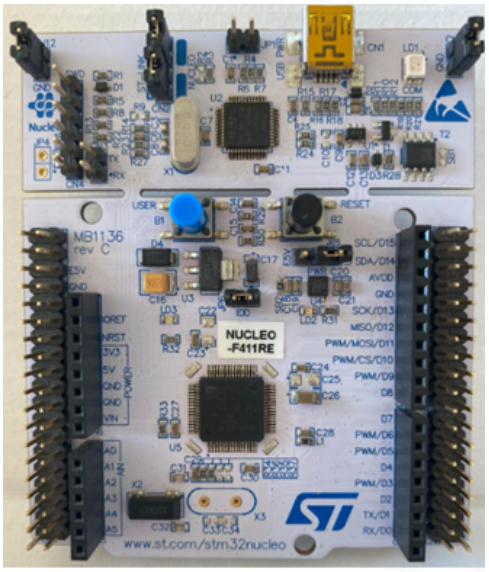
5.1.2.1 Plataforma de desenvolvimento STM32F411

O microcontrolador STM32F411, ilustrado na Figura 5.1, foi desenvolvido pela STMicroelectronics e possibilita o processamento de todas as etapas (apresentadas no Capítulo 4), além de permitir sua integração para o módulo ESP-01. A demais, a plataforma possui:

- CPU ARM®32-bit Cortex®-M4 com FPU
- Dois botões de pressão: USER e RESET
- VDD de 1,7 V a 3,6 V

- Conversor A/D de 1 × 12 bits, 2,4 MSPS: até 16 canais
- DMA de uso geral: controladores DMA de 16 fluxos com FIFOs

Figura 5.1: Plataforma de desenvolvimento *STM32F411*



Fonte: Autoria própria

O STM32CubeIDE é um ambiente de desenvolvimento integrado (IDE) criado pela STMicroelectronics, projetado especificamente para projetos baseados nos microcontroladores da família STM32. Essa ferramenta oferece uma solução completa ao combinar configuração de periféricos, geração de código e depuração em um único ambiente. Integrado ao STM32CubeMX, o STM32CubeIDE possibilita configurar graficamente periféricos e middlewares, simplificando o desenvolvimento de sistemas embarcados. Baseado na plataforma Eclipse, o IDE é multiplataforma, compatível com Windows, macOS e Linux, além de oferecer suporte a diversas ferramentas de depuração, como ST-LINK e JTAG (STMICROELECTRONICS, 2024).

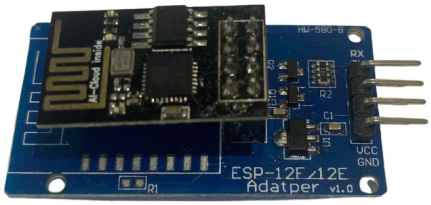
5.1.2.2 Módulo ESP-01

Para viabilizar a conexão com a internet e o envio de dados do microcontrolador para o servidor web, foi utilizado o módulo ESP-01, fabricado pela Espressif Systems. Este dispositivo, é equipado com o microcontrolador ESP8266 de 32 bits, como o módulo opera com uma tensão de trabalho de 3,3 V, foi necessário incluir um adaptador que converte essa tensão para 5 V, compatível com as linhas de comunicação da placa de desenvolvimento.

Além de realizar a conversão de tensão, o adaptador também simplifica a conexão, reduzindo os pinos do ESP-01 apenas aos de alimentação (VCC e GND) e comunicação serial (TX e RX). Os pinos de comunicação são conectados ao microcontrolador por meio de linhas alternadas com o protocolo Universal Síncrono/Assíncrono (USART), garantindo a transmissão e recepção dos dados. A configuração do módulo ESP-01 com o adaptador pode ser visualizada

na Figura 5.2.

Figura 5.2: Módulo ESP-01 com o adaptador



Fonte: Autoria própria

O ESP-01 utiliza os comandos AT como modo de operação, isso facilita a interação com o microcontrolador, desta forma é possível desenvolver um código dentro do CubeIDE. Visto isso, a Espressif disponibiliza um manual com todos os comandos AT e suas descrições, além de informações referente a comunicação [1]. Os principais comandos AT executados pelo ESP-01 para realizar a conexão com a internet e as requisições web constam na seção 4.5.

5.2 Resultados dos algoritmos de EdgeECG em MATLAB

A implementação em MATLAB do EdgeECG inclui Transformada, Quantização e Limiarização e suas operações inversas. A entropia (Huffman) não está inclusa neste sequência de experimentos.

Tabela 5.1: Métricas de desempenho para o Código com quantização e limiarização, sem codificação Huffman

Sinal	CR	PRD (%)	RMSE	Energia retida
100 (10 segundos)	6,96	15,25	0,0362	96,7
100 (1 hora)	7,25	19,34	0,0468	94,94
101 (1 hora)	7,95	12,26	0,0113	98,43
102 (1 hora)	7,39	5,85	0,0669	99,6

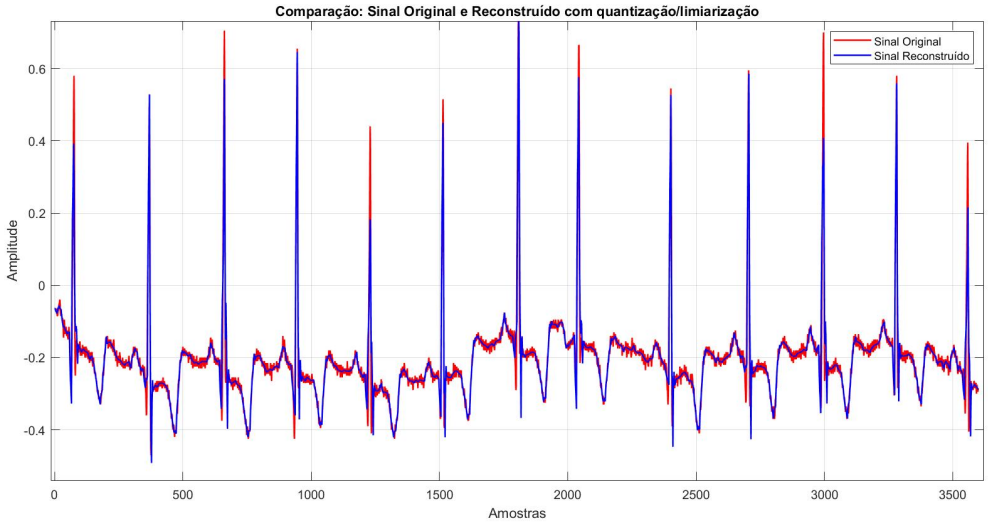
A Tabela 5.1 apresenta os valores das métricas descritas na Seção 4.4 para três sinais distintos do banco de dados MIT-BIH Arrhythmia Database da PhysioNet. O sinal 100 foi avaliado em duas condições: uma amostra de 10 segundos e um período de 1 hora de dados. Essa análise permite relacionar os resultados obtidos com os experimentos anteriores, que foram realizados utilizando a amostra de 10 segundos do sinal.

Os resultados mostram que, para a mesma família wavelet e o mesmo número de níveis de decomposição, o sinal com maior fidelidade de reconstrução foi o sinal 102, com um PRD de 5,85%. Por outro lado, a melhor taxa de compressão foi alcançada pelo sinal 101, com CR = 7,39. Além disso, para ambos os sinais, a energia retida permaneceu acima de 90%.

Na Figura 5.3, é apresentado o sinal de entrada (amostra 100 de 10s) sobreposto ao sinal resultante após a aplicação das etapas de quantização e limiarização. Observa-se que as formas

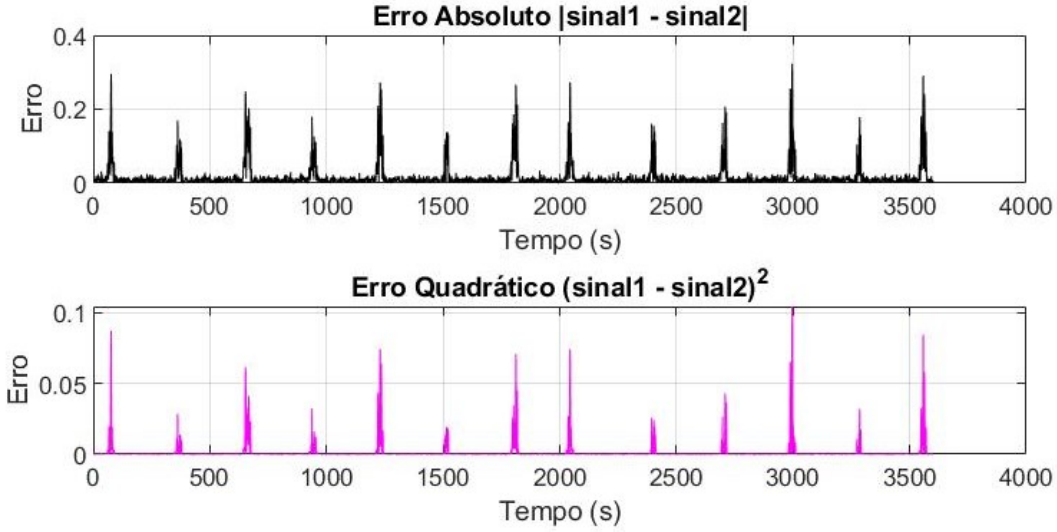
[1] `https://www.espressif.com/sites/default/files/documentation/4aesp8266_at_instruction_set_en.pdf`

Figura 5.3: Comparação do sinal (100 de 10s) de entrada com o sinal reconstruído após quantização/limiarização



Fonte: Autoral

Figura 5.4: Erro presente entre o sinal (100 de 10s) de entrada e o sinal reconstruído após quantização/limiarização



Fonte: Autoral

do segmento do sinal original foram preservadas, evidenciando a eficácia do método. Complementarmente, a Figura 5.4 ilustra o erro entre os dois sinais, cujo valor foi quantificado com um RMSE de 0,0362 o que indica um nível de distorção reduzido e satisfatório para as aplicações propostas.

5.2.1 Adicionando Huffman ao EdgeECG em MATLAB

Para avaliar o potencial de compressão do ECG, o algoritmo de Huffman descrito em Python foi aplicado ao ECG proveniente do MatLab após as etapas transformada, quantiza-

ção e limiarização. Após a construção do novo vetor J que aplica quantizado e com a aplicação do limiar, a taxa de compressão do sinal com 360 amostras (sinal 100 de 1s) obtida foi de CR=6,36.

Quando comparado ao armazenamento sem Quantização e Limiarização, observa-se que a compressão com a codificação Huffman resultou em um CR=3,25. Isso indica que o dado comprimido apenas com as etapas de transformada e compressão é cerca de 2 vezes maior que o dado que contém quantização e transformadas.

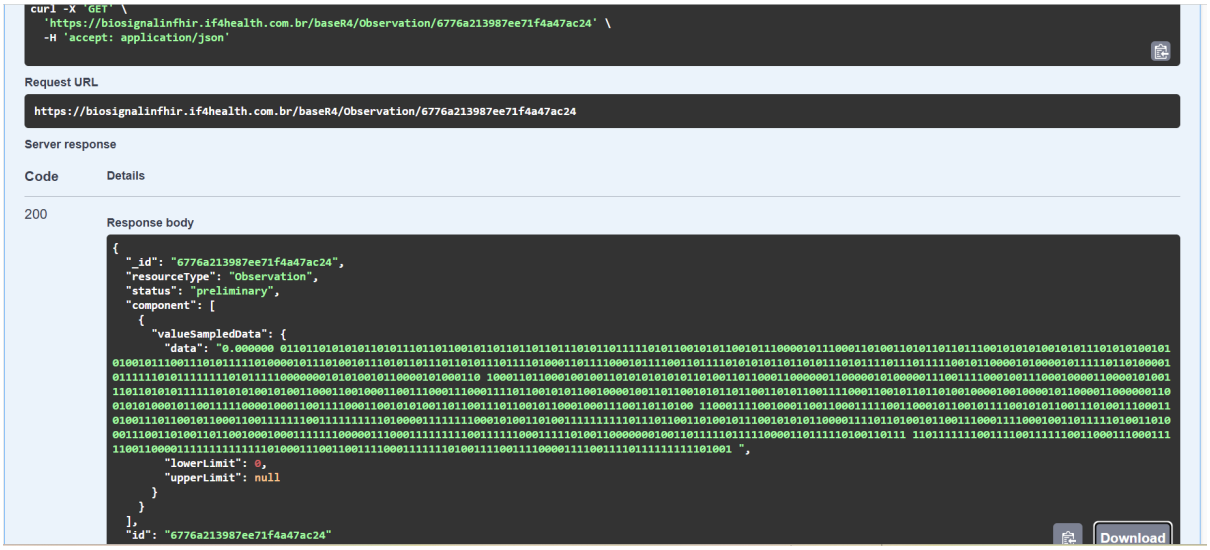
5.3 Resultados do EdgeECG

Os códigos em C que implementam as 5 etapas do EdgeECG (Seção 4.5) no microcontrolador STM32F411 não contemplam Quantização e Limiarização.

Para validar o processamento efetivo do sistema EdgeECG, os dados gerados pela etapa de transformada foram comparados com aqueles obtidos pela execução do código em MATLAB. Essa comparação assegurou que todos os coeficientes de detalhe e aproximação correspondiam fielmente aos valores processados no MATLAB.

A Figura 5.5 ilustra a forma como os dados foram armazenados na nuvem para um envio de uma parcela de 360 amostras (sinal 100 de 1s). O ECG binarizado em FHIR na nuvem é o resultado final do processamento das 5 etapas do EdgeECG. Foi adicionado espaços entre os códigos binários para indicar a quantidade de PATCHs realizados, quatro requisições PATCHs no caso da Figura 5.5.

Figura 5.5: GET com ID direto do API *Biossignal in FHIR* (PEREIRA et al., 2023)



Fonte: Autoral

Uma segunda validação foi realizada utilizando o codificação Huffman em Python combinada com a transformada em MATLAB (Subseção 5.2.1) para checar se o binário gerado pelo MCU corresponde ao binário do Python. Os resultados mostraram que o vetor codificado

em ambos os casos apresenta 100% de fidelidade, o que valida a possibilidade de realizar a decodificação do sinal diretamente na nuvem. Para isso, o mesmo código de decodificação desenvolvido em Python deve ser implementado no ambiente da nuvem.

O armazenamento final do sinal testado no MCU revelou que, após a aplicação da Transformada Wavelet (WT) e a codificação Huffman, o vetor binário armazenado contém 1215 caracteres, com cada caractere representando um único bit.

Como os códigos em C e em MATLAB são equivalentes (subseção 5.2.1), o CR=3,25 atingido para sinal com 360 amostras é o mesmo. Considerando que os dados utilizados da codificação Huffman foram apenas o sinal de 1s, essa eficiência é particularmente relevante para sinais de ECG, pois a codificação Huffman é mais eficaz quanto maior a quantidade de amostras similares, um cenário comum em sinais de ECG devido à sua natureza cíclica e repetitiva. Além disso, considerando a proposta de compressão de ECGs de longo prazo, espera-se que a codificação Huffman apresente resultados ainda mais efetivos ao ser aplicada a amostras maiores de dados.

5.4 Reprodutibilidade e Disponibilidade dos Experimentos

Considera-se como principal resultado a própria implementação do EdgeECG na placa com o envio do ECG para a nuvem. Os Códigos-Fonte estão disponíveis para reprodução dos experimentos e para a realização dos trabalhos futuros no GitHub[2]. Enquanto a API Biossignal in FHIR estiver disponível online[3], o resultado apresentado na Figura 5.5 pode ser consultado com um comando no terminal:

```
curl -X 'GET' \
    'https://biosignalinfhir.if4health.com.br/baseR4/Observation/6776a213987ee71f4a47ac24' \
    -H 'accept: application/json'
```

[2] https://github.com/if4health/EdgeECG
[3] https://biosignalinfhir.if4health.com.br/api-docs

6 CONCLUSÕES

Este trabalho abordou a compressão de sinais ECG utilizando transformadas, com uma explicação detalhada sobre transformadas wavelet, técnicas de quantização e codificação de entropia. Com base nesses fundamentos, foi desenvolvido o sistema EdgeECG, projetado para a compressão de sinais ECG. O desempenho do sistema foi analisado de forma comparativa, utilizando as principais métricas abordadas na literatura sobre o tema. Os resultados destacam a eficácia e o potencial do EdgeECG para aplicações práticas, especialmente na transmissão eficiente dos dados processados.

Ao analisar a literatura, observa-se que poucos trabalhos aplicam, de forma prática, a teoria de compressão de sinais em sistemas embarcados, demonstrando simultaneamente a eficiência de transmissão e armazenamento, como realizado neste estudo. Este trabalho se diferencia ao integrar etapas do processo de compressão em um dispositivo IoT, com validação tanto no ambiente de desenvolvimento quanto no hardware embarcado.

Para viabilizar os testes, as etapas foram implementadas em diferentes ambientes e linguagens de programação. O MATLAB foi utilizado para a aplicação das transformadas e para os testes iniciais de eficiência, devido à sua robustez e ampla biblioteca de ferramentas específicas para processamento de sinais e transformadas. Essa escolha garantiu maior confiabilidade na validação do método, uma vez que as transformadas implementadas no microprocessador puderam ser comparadas a funções consolidadas e amplamente testadas no MATLAB.

Os resultados apresentados demonstram um desempenho relevante em termos de compressão, aliado a um elevado nível de preservação do sinal original. Um aspecto fundamental observado na reconstrução do sinal, após as etapas de quantização e limiarização, foi o aproveitamento das características específicas da decomposição por transformada wavelet (WT) que permitem a separação hierárquica dos coeficientes do sinal em componentes de detalhe e aproximação. As etapas de compressão foram aplicadas exclusivamente aos coeficientes de detalhe, que carregam informações de alta frequência menos críticas para o exame de ECG. Por outro lado, os coeficientes de aproximação, responsáveis por encapsular as características globais e essenciais do sinal, foram preservados sem alterações. Essa estratégia assegura a integridade das informações fundamentais para diagnóstico médico, mantendo a fidelidade necessária para avaliações clínicas, mesmo após o processo de compressão.

Além disso, o código das transformadas foi adaptado para implementação no microcontrolador, e a codificação Huffman foi inicialmente validada em Python, com posterior migração para C. A escolha do Python para a codificação considerou a necessidade de suporte para decodificação futura do sinal em uma nuvem de armazenamento. A implementação em C assegurou a compatibilidade com dispositivos embarcados.

Os resultados obtidos nas diferentes etapas confirmam que o sistema atinge um grau significativo de compressão, mantendo a precisão e eficiência na reconstrução e decodificação do sinal. Assim, o método desenvolvido cumpre o objetivo proposto de comprimir e armazenar

dados de ECG processados em dispositivos IoT, destacando-se como uma solução prática e eficiente para aplicações em telemedicina e monitoramento remoto.

A integração de dispositivos IoT com soluções baseadas no padrão FHIR representa um marco importante na evolução da saúde digital, permitindo uma abordagem mais personalizada e eficiente no cuidado ao paciente. Este estudo busca contribuir para essa transformação, explorando as possibilidades de compressão eficiente de sinais ECG em ambientes interoperáveis.

6.1 Trabalhos Futuros

Um ponto relevante ao analisar o armazenamento de sinais codificados em nuvem é o acesso ao sinal após a aplicação inversa das etapas do sistema de compressão. Embora o EdgeECG tenha comprovado eficiência na reconstrução e decodificação dos dados, oferecendo os meios necessários para que essas operações possam ser realizadas em um ambiente de armazenamento em nuvem, testes práticos dessas etapas inversas, incluindo a exibição do sinal reconstruído após o ciclo completo de compressão, ainda não foram realizados.

Para implementar essa validação, seria necessária uma requisição adicional ao sistema, que incluiria a construção da árvore de Huffman, ou seja, a relação dos códigos binários com o conjunto total de dados codificados. Após essa etapa, seria possível executar o código de decodificação em Python (já desenvolvido e disponível em [ref]) e aplicar a função `wavedec` no mesmo ambiente para validar a reconstrução do sinal. Além disso, a etapa de quantização/limiarização, que foi desenvolvida no MATLAB, permanece desconectada do fluxo de processamento na placa embarcada. Assim, um trabalho futuro promissor seria integrar essa etapa diretamente ao processamento no microcontrolador, elevando o desempenho geral do sistema.

Outra área de investigação relevante seria a exploração de diferentes famílias de wavelets, que poderiam oferecer maior eficiência em termos de compressão e preservação de características críticas do sinal. Além disso, seria interessante validar o método utilizando sinais com ruídos típicos de exames realizados em campo, assegurando que o sistema mantém seu desempenho em condições reais de aplicação.

Por fim, uma importante melhoria futura seria implementar o processamento em buffers do sinal, utilizando DMA (Direct Memory Access), para possibilitar o manejo eficiente de sinais ECG extensos sem comprometer a capacidade de processamento do microcontrolador. Isso permitiria que o sistema processasse grandes volumes de dados de forma mais eficiente, expandindo seu uso para aplicações mais robustas em telemedicina e monitoramento remoto.

REFERÊNCIAS

ABENSTEIN, M.; TOMPKINS, W. CORTES: a hybrid algorithm for ecg signal compression. **IEEE Transactions on Medical Imaging**, [S.l.], v.11, p.123–130, 1982.

ABREU MOREIRA, M. de; SOUZA, A. de. Estudo comparativo entre algoritmos das transformadas discretas de Fourier e Wavelet. **Revista Brasileira de Computação Aplicada**, [S.l.], v.7, n.1, p.11–22, 2015.

ANGOMED. **Anatomia e Electrofisiologia Básica**, 2024. Acessado em: 25 de dezembro de 2024, Disponível em: `https://angomed.com/anatomia-e-electrofisiologia-basica/`.

AYAZ, M. et al. The Fast Health Interoperability Resources (FHIR) standard: systematic literature review of implementations, applications, challenges and opportunities. **JMIR Medical Informatics**, [S.l.], v.9, n.7, p.e21929, 2021.

BARBONI, M. A. A. **Compressão de sinais de ECG**: análise de métodos e aplicações, 1992. Disponível em: `http://cris.uefs.br/pdfs/barboni_1992`. Acesso em: 30 dez. 2024.

BENDER, D.; SARTIPI, K. HL7 FHIR: an agile and restful approach to healthcare information exchange. In: IEEE INTERNATIONAL SYMPOSIUM ON COMPUTER-BASED MEDICAL SYSTEMS, 26., 2013. **Proceedings...** [S.l.: s.n.], 2013. p.326–331.

CAMPITELLI, M. A. **Compressão de sinais ECG utilizando DWT com quantização não-linear e por sub-bandas**. 2015. Dissertação (Mestrado em Ciência da Computação) — Universidade de Brasília, Brasília, Brasil.

CDR, S. **HAPI FHIR - The Open Source FHIR API for Java.**, 2019. Acessado em 11/12/2024, Disponível em: `https://hapifhir.io/`.

CHANDRA, S.; SHARMA, A.; SINGH, G. K. A comparative analysis of performance of several wavelet based ECG data compression methodologies. **Irbm**, [S.l.], v.42, n.4, p.227–244, 2021.

CHOUAKRI, S. A.; DJAAFRI, O.; TALEB-AHMED, A. Wavelet transform and Huffman coding based electrocardiogram compression algorithm: application to telecardiology. In: JOURNAL OF PHYSICS: CONFERENCE SERIES, 2013. **Anais...** [S.l.: s.n.], 2013. v.454, n.1, p.012086.

CODING, R. **Huffman Coding**, 2024. Acesso em: 25 de dezembro de 2024, Disponível em: `https://rosettacode.org/wiki/Huffman_coding`.

COVER, T. M.; THOMAS, J. A. **Elements of Information Theory**. 2nd.ed. [S.l.]: John Wiley & Sons, 2006.

COX, W.; COLS, J. AZETEC: a method for time-amplitude and interval encoding. **IEEE Transactions on Communications**, [S.l.], v.16, p.34–45, 1968.

COX, W.; COLS, J. Time-Compression Algorithms for Signal Transmission. **IEEE Transactions on Information Theory**, [S.l.], v.18, p.112–120, 1972.

DAUBECHIES, I. **Ten Lectures on Wavelets**. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics (SIAM), 1992. Clássico sobre teoria e aplicações das wavelets.

DOMINGUES, M. et al. Explorando a transformada wavelet contínua. **Revista Brasileira de Ensino de Física**, [S.l.], v.38, 09 2016.

ENFERMAGEM, A. da. **O que é Eletrocardiograma?**, 2024. 2024.

ENGENHARIA, A. **O que é**: limiarização em processamento de imagem, 2023. Acessado em: dezembro de 2024.

FERREIRA, R. **REST**: princípios e boas práticas, 2017. Acessado em: dezembro de 2024.

GERSHO, A.; GRAY, R. M. **Vector Quantization and Signal Compression**. [S.l.]: Springer, 1992. (The Springer International Series in Engineering and Computer Science, v.159).

GOLDBERGER, A. L. et al. PhysioBank, PhysioToolkit, and PhysioNet: components of a new research resource for complex physiologic signals. **Circulation**, [S.l.], v.101, n.23, p.e215–e220, 2000.

GUYTON, A. C.; HALL, J. E. **Tratado de Fisiologia Médica**. 13ª.ed. Rio de Janeiro: Elsevier, 2017.

HL7. **FHIR Release 4**, 2019. Acessado em 11/12/2024, Disponível em: `https://www.hl7.org/fhir/`.

HUFFMAN, D. A. A Method for the Construction of Minimum-Redundancy Codes. **Proceedings of the IRE**, [S.l.], v.40, n.9, p.1098–1101, 1952.

Kaplan Berkaya, S. et al. A survey on ECG analysis. **Biomedical Signal Processing and Control**, [S.l.], v.43, p.216–235, 2018.

KOZAKEVICIUS, A.; BAYER, F. FILTRAGEM DE SINAIS VIA LIMIARIZAÇÃO DE COEfiCIENTES WAVELET. **Ciência e Natura**, [S.l.], v.26, p.37–51, 10 2014.

KUMAR, R.; KUMAR, A.; PANDEY, R. **ECG Signal Compression Using Different Techniques**. [S.l.: s.n.], 2010. v.125, p.231–241.

LI, X.; ZHANG, Y.; WANG, J. Application of Continuous Wavelet Transform in ECG Signal Analysis: detection of qrs complex, p and t waves. **Sensors**, [S.l.], v.22, n.12, p.4343, 2022.

MALEKI, A. Wavelet Transform in ECG Signal Classification: a comprehensive study. **arXiv preprint arXiv:2404.09393**, [S.l.], 2024.

MALLAT, S. **A Wavelet Tour of Signal Processing**. [S.l.]: Academic Press, 1999.

MANZAN, M. F. **Utilização de Transformadas Wavelet na Análise e Classificação de Sinais Biomédicos**. 2006. Tese (Doutorado em Ciência da Computação) — Universidade Federal de Uberlândia (UFU), Uberlândia, Brasil. Acesso em: 25 de dezembro de 2024.

MITRA, S. K. **Digital Signal Processing**: a computer-based approach. 3rd.ed. New York: McGraw-Hill, 2006.

MUELLER, A. Turning Point Algorithm for Signal Compression. **Journal of Signal Processing**, [S.l.], v.12, p.101–112, 1978.

My EKG. **Derivações do ECG**, n.d. Acesso em: 19 dez. 2024.

OLIVEIRA, B. R. d. Transformada Wavelet: teoria e aplicações na análise de eletrocardiograma. **Academia.edu**, [S.l.], 2018.

OLIVEIRA, C. H.; SANTOS, L. F. V.; SOUZA, P. S. T. **Técnica de Compressão de Sinais de Oscilografia Baseada na Variação do Conteúdo Espectral**, 2017. Acessado em 22 de dezembro de 2024.

OLIVEIRA, H. N. de. **Arquivototal - Repositório UFPB**, 2024. Acesso em: 25 de dezembro de 2024, Disponível em: `https://repositorio.ufpb.br/jspui/bitstream/123456789/16919/1/Arquivototal.pdf`.

OPPENHEIM, A. V.; WILLSKY, A. S. **Sinais e Sistemas**. [S.l.]: Prentice Hall, 1997.

PEREIRA, C. R. et al. FASS-ECG: a fhir cloud api to enable streaming and storage of continuous 12-leads ecgs. In: IEEE 36TH INTERNATIONAL SYMPOSIUM ON COMPUTER-BASED MEDICAL SYSTEMS (CBMS), 2023., 2023. **Anais...** [S.l.: s.n.], 2023. p.898–903.

PROAKIS, J. G.; MANOLAKIS, D. G. **Digital Signal Processing**: principles, algorithms, and applications. 4th.ed. [S.l.]: Pearson, 2006. ISBN: 978-0131873746.

RANJEET, K.; KUMAR, A.; PANDEY, R. K. ECG Signal Compression Using Different Techniques. In: ADVANCES IN COMPUTING, COMMUNICATION AND CONTROL, 2011, Berlin, Heidelberg. **Anais...** Springer Berlin Heidelberg, 2011. p.231–241.

REBOLLO-NEIRA, L. Effective high compression of ECG signals at low level distortion. **Scientific Reports**, [S.l.], v.9, n.1, p.1–9, 2019.

RNDS. **Ecossistema FHIR**, 2022. 2022.

RUTTIMANN, U.; PIPBERGER, H. Linear Prediction and Entropy Encoding for ECG Data Compression. **Biomedical Engineering**, [S.l.], v.26, p.1012–1017, 1979.

SALOMON, D. **Data compression - The Complete Reference, 4th Edition.** [S.l.: s.n.], 2007.

SANTOS, M. **Algoritmos de Compressão de Dados**: huffman coding e lzw, 2024. Acessado em: 30 de dezembro de 2024.

SARIPALLE, R. K. Fast Health Interoperability Resources (FHIR): current status in the healthcare system. **International Journal of E-Health and Medical Communications (IJEHMC)**, [S.l.], v.10, n.1, p.76–93, 2019.

SAYOOD, K. **Introduction to Data Compression**. 5th.ed. [S.l.]: Morgan Kaufmann, 2017.

SAúDE, S. **Manual de ECG - Trecho**, 2024. Acesso em: 25 de dezembro de 2024, Disponível em: `https://s3.sanar.online/images/p/Manual%20de%20ECG_trecho.pdf`.

SHANNON, C. E. Communication in the Presence of Noise. **Proceedings of the IRE**, [S.l.], v.37, n.1, p.10–21, 1949.

SIDHWA, H. **Huffman Decoding**, 2023. Acesso em: 25 de dezembro de 2024, Disponível em: https://www.geeksforgeeks.org/huffman-decoding/.

SÖRNMO LEIF E LAGUNA, P. Processamento de sinal de eletrocardiograma (ECG). **Enciclopédia Wiley de engenharia biomédica**, [S.l.], 2006.

STEWART, S.; COLS, J. Delta Encoding for ECG Signal Compression. **IEEE Transactions on Biomedical Engineering**, [S.l.], v.20, p.234–241, 1973.

STMICROELECTRONICS. **STM32CubeIDE - Integrated Development Environment for STM32**. [S.l.: s.n.], 2024. Acesso em: 19 dez. 2024.

TONIN, M. V. P. **Técnicas de quantização e compressão aplicadas a sinais biomédicos**, 2021. Acessado em: 20 dez. 2024, Disponível em: http://icts.unb.br/jspui/bitstream/10482/42501/1/2021_MarcosViniciusPrescendoTonin.pdf.

WALLACE, G. K. The JPEG Still Picture Compression Standard. **Communications of the ACM**, [S.l.], v.34, n.4, p.30–44, 1991.

WEEKS, M. **Digital Signal Processing using MATLAB and Wavelets**. [S.l.]: Infinity Science Press, 2007. Acesso em: 19 dez. 2024.