

UNIVERSIDADE ESTADUAL PAULISTA "JÚLIO DE MESQUITA FILHO"

FACULDADE DE CIÊNCIAS - CAMPUS BAURU

DEPARTAMENTO DE COMPUTAÇÃO

BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

MANUELE SABATINI CHRISTÓFALO

**ANÁLISE COMPARATIVA DE ARDUINOS NA IMPLEMENTAÇÃO
DE SISTEMAS EMBARCADOS PARA MONITORAMENTO DE
TREMORES NA DOENÇA DE PARKINSON**

BAURU

Novembro/2025

MANUELE SABATINI CHRISTÓFALO

**ANÁLISE COMPARATIVA DE ARDUINOS NA IMPLEMENTAÇÃO
DE SISTEMAS EMBARCADOS PARA MONITORAMENTO DE
TREMORES NA DOENÇA DE PARKINSON**

Trabalho de Conclusão de Curso do Curso
de Ciência da Computação da Universidade
Estadual Paulista “Júlio de Mesquita Filho”,
Faculdade de Ciências, Campus Bauru.
Orientador: Prof. Dr. Clayton Reginaldo Pereira

BAURU
Novembro/2025

C556a

Christófalo, Manuele Sabatini

Análise comparativa de Arduinos na implementação de sistemas embarcados para monitoramento de tremores na doença de Parkinson / Manuele Sabatini Christófalo. -- Bauru, 2025

72 p.

Trabalho de conclusão de curso (Bacharelado - Ciência da Computação) - Universidade Estadual Paulista (UNESP), Faculdade de Ciências, Bauru

Orientador: Clayton Reginaldo Pereira

1. Doença de Parkinson. 2. Monitoramento de tremores. 3. Sistemas embarcados. 4. Arduino. 5. Redes Neurais Recorrentes. I. Título.

Manuele Sabatini Christófaló

Análise comparativa de Arduinos na implementação de sistemas embarcados para monitoramento de tremores na doença de Parkinson

Trabalho de Conclusão de Curso do Curso de Ciência da Computação da Universidade Estadual Paulista "Júlio de Mesquita Filho", Faculdade de Ciências, Campus Bauru.

Banca Examinadora

Prof. Dr. Clayton Reginaldo Pereira

Orientador

Universidade Estadual Paulista "Júlio de Mesquita Filho"

Faculdade de Ciências

Departamento de Ciência da Computação

Prof. Dra. Simone das Graças Domingues Prado

Universidade Estadual Paulista "Júlio de Mesquita Filho"

Faculdade de Ciências

Departamento de Ciência da Computação

Prof. Dr. Marco Aurélio Rocha

Universidade Estadual Paulista "Júlio de Mesquita Filho"

Faculdade de Ciências

Departamento de Ciência da Computação

Bauru, 12 de Novembro de 2025.

Ao meu pai, meu primeiro parceiro de pesquisa e de experimentos malucos.

Agradecimentos

Agradeço, primeiramente, aos meus pais e ao meu irmão, por todo apoio fornecido durante a minha vida. Agradeço também aos meus amigos de infância, Brenda e Edgar, que, mesmo longe, me acompanham em quem eu sou e serei; e aos meus amigos de faculdade, Paulo e Dante, com quem compartilhei 4 anos de trabalhos, desafios e, principalmente, boas risadas. Por fim, agradeço aos meus docentes - em especial, ao meu orientador, Prof. Dr. Clayton, por me apresentar a carreira acadêmica e consolidar minha aspiração em me tornar professora.

*"Não há nada estático, nada quieto neste universo. Tudo é ondulação, tudo é dança; tudo é
rapidez e triunfo"*

- *As Ondas*, Virgínia Woolf

Resumo

A Doença de Parkinson (DP) é uma afecção neurodegenerativa crônica e progressiva do sistema nervoso central, que impacta significativamente a função motora. O monitoramento de seus sintomas, em especial os tremores, é fundamental para o manejo clínico, porém frequentemente se baseia em avaliações subjetivas. Este trabalho propõe uma revisão bibliográfica de soluções quantitativas por meio do desenvolvimento e análise comparativa de dois sistemas embarcados vestíveis, projetados visando a coleta e análise de dados de tremores. O estudo confronta duas abordagens de *hardware*: um bracelete utilizando a placa Arduino Nano 33 BLE Sense Rev 2, que possui uma unidade de medição inercial (IMU) integrada, e um segundo protótipo baseado no Arduino UNO, complementado por sensores externos. As plataformas são avaliadas sob múltiplos critérios, incluindo precisão dos sensores, eficiência computacional, consumo energético, usabilidade, escalabilidade e custo-benefício. Ainda no estudo, desenvolve-se uma arquitetura *Long Short-Term Memory* (LSTM) otimizada para a identificação de padrões nas séries temporais possivelmente coletadas pelos Arduinos. A análise final visa determinar a configuração de *hardware* mais adequada para o contexto biomédico da DP, contribuindo para o avanço de tecnologias assistivas de baixo custo e alta precisão no acompanhamento de doenças neurodegenerativas.

Palavras-chave: Doença de Parkinson; monitoramento de tremores; sistemas embarcados; Arduino; redes neurais recorrentes.

Abstract

Parkinson's Disease (PD) is a chronic and progressive neurodegenerative disorder of the central nervous system that significantly impacts motor function. Monitoring its symptoms, particularly tremors, is fundamental for clinical management, yet it often relies on subjective assessments. This work proposes a literature review of quantitative solutions through the development and comparative analysis of two wearable embedded systems, designed for the collection and analysis of tremor data. The study compares two hardware approaches: a bracelet using the Arduino Nano 33 BLE Sense Rev 2 board, which has an integrated inertial measurement unit (IMU), and a second prototype based on the Arduino UNO, supplemented by external sensors. The platforms are evaluated under multiple criteria, including sensor accuracy, computational efficiency, power consumption, usability, scalability, and cost-effectiveness. Furthermore, the study develops an optimized Long Short-Term Memory (LSTM) architecture for identifying patterns in the time-series data potentially collected by the Arduinos. The final analysis aims to determine the most suitable hardware configuration for the biomedical context of Parkinson's disease, contributing to the advancement of low-cost, high-precision assistive technologies for monitoring neurodegenerative diseases.

Keywords: Parkinson's Disease; tremor monitoring; embedded systems; Arduino; recurrent neural networks.

Lista de figuras

Figura 1 – Arduino UNO	23
Figura 2 – Arduino Nano	24
Figura 3 – Acelerômetro capacitor MEMS	25
Figura 4 – Giroscópio MEMS	26
Figura 5 – Esquematização do Efeito Hall	26
Figura 6 – Funcionamento LSTM	34
Figura 7 – Esquemático do pUNO_v2	40
Figura 8 – Esquemático do pNANO_v2	40
Figura 9 – Arduino Nano no Bracelete (pós-solda)	42
Figura 10 – Ilustração do funcionamento da janela deslizante	52
Figura 11 – Resultados do Ensaio A para 2Hz	61
Figura 12 – Resultados do Ensaio B para 30° de Amplitude	61

Lista de quadros

Quadro 1 – Detalhamento de componentes e custos dos protótipos	57
Quadro 2 – Resultado dos testes de duração de bateria	59

Lista de tabelas

Tabela 1 – Resultados do Ensaio A: Análise de Precisão em Frequência	60
Tabela 2 – Resultados do Ensaio B: Análise de Fidelidade da Forma de Onda	61
Tabela 3 – Resultados do Ensaio C: Quantificação do Ruído Intrínseco (Desvio Padrão)	62
Tabela 4 – Resultados do Ensaio C: Quantificação da Deriva (<i>Drift</i>)	62

Lista de abreviaturas e siglas

AC	Componente AC
ADC	Conversor Analógico-Digital
API	API Keras
BLE	<i>Bluetooth Low Energy</i>
CSV	Formato CSV (implícito)
DC	Componente DC
DFT	Transformada Discreta de Fourier
DP	Doença de Parkinson
FFT	Transformada Rápida de Fourier
FSR	Fundo de Escala (<i>Full-Scale Range</i>)
GPU	Unidades de Processamento Gráfico
GT	<i>Ground Truth</i> (Verdade Fundamental)
HY	Escala Hoehn e Yahn
IA	Inteligência Artificial
IMU	<i>Inertial Measurement Unit</i> (Unidade de Medição Inercial)
IoT	Internet das Coisas
L-Dopa	Levodopa
LSB	<i>Least Significant Bits</i>
LSTM	<i>Long Short-Term Memory</i>
MEMS	<i>Micro-Electro-Mechanical-Systems</i> (Sistemas Micro-eleto-mecânicos)
ML	<i>Machine Learning</i> (Aprendizado de Máquina)
MPU	<i>Motion Processing Unit</i> (Unidade de processamento motora)
PDM	Microfone PDM (implícito)

PWM	<i>Pulse Width Modulation</i> (Pulso por Modulação de Largura)
RMSE	<i>Root Mean Square Error</i> (Erro Quadrático Médio)
RNN	<i>Recurrent Neural Network</i> (Rede Neural Recorrente))
SoC	<i>System-on-Chip</i>
SPI	Protocolo SPI (implícito)
SVM	Máquinas de Vetores de Suporte
TFLite	<i>TensorFlow Lite</i>
TinyML	<i>Tiny Machine Learning</i>
UPDRS	<i>Unified Parkinson's Disease Rating Scale</i> (Escala Unificada de Avaliação da Doença de Parkinson)

Sumário

1	INTRODUÇÃO	16
1.1	Problemática	18
1.2	Justificativa	18
1.3	Objetivos	19
1.3.1	Objetivo Geral	19
1.3.2	Objetivos Específicos	19
1.4	Organização do Trabalho	20
2	FUNDAMENTAÇÃO TEÓRICA	21
2.1	<i>Wearables</i> aplicados na saúde	21
2.2	Arduino	21
2.2.1	Arduino UNO R3	22
2.2.2	Arduino Nano 33 BLE Sense Rev 2	23
2.2.3	Síntese Comparativa	24
2.3	Funcionamento dos Sensores	24
2.4	Cálculo do Movimento e Filtro de Kalman	26
2.4.1	Processamento e Obtenção de Grandezas Físicas dos Sensores Inerciais	26
2.4.1.1	Acelerômetro: Cálculo da Aceleração Própria	27
2.4.1.2	Giroscópio: Cálculo da Velocidade Angular e Integração da Orientação	28
2.4.1.3	Magnetômetro: Cálculo do Vetor do Campo Magnético	29
2.4.2	Fusão de Sensores e o Filtro de Kalman	29
2.5	Aprendizado de Máquina	31
2.6	Redes Neurais Recorrentes	33
2.6.1	<i>Long Short-Term Memory</i>	33
2.7	Trabalhos Relacionados	35
3	MATERIAIS E MÉTODOS	36
3.1	Validação dos Sensores e Componentes	36
3.2	Funcionamento dos Sistemas	40
3.2.1	Duração da Bateria	41
3.2.2	Soldagem	41
3.3	Protocolo de Testes Mecânicos Comparativos	42
3.3.1	Montagem da Plataforma de Testes	43
3.3.2	Ensaio A: Análise de Precisão em Frequência	43
3.3.3	Ensaio B: Análise de Fidelidade da Forma de Onda	44
3.3.4	Ensaio C: Análise de Estabilidade Estática (Ruído e Deriva)	45

3.4	Análise Comparativa dos Sinais Inerciais	46
3.4.1	Análise no Domínio da Frequência (FFT)	46
3.4.2	Análise no Domínio do Tempo (Fidelidade)	47
3.4.3	Análise de Estabilidade Estática (Ruído e Deriva)	48
3.4.3.1	Quantificação do Ruído Intrínseco	48
3.4.3.2	Quantificação da Deriva (Drift)	48
3.5	Análise de Viabilidade do Modelo LSTM	49
3.5.1	Pré-Processamento dos Dados	49
3.5.2	Arquitetura de Referência (LSTM)	52
3.5.3	Metodologia de Treinamento e Avaliação	54
3.5.4	Aplicação do <i>TinyML</i>	55
3.5.4.1	<i>Pipeline</i> de Conversão para o Ambiente Embarcado	55
4	RESULTADOS EXPERIMENTAIS	57
4.1	Comparativo de <i>Hardware</i> e Custo	57
4.2	Comparativo de Consumo Energético	58
4.3	Comparativo de Captura	60
4.4	Análise de Viabilidade da Implementação LSTM	63
4.4.1	Validação da Viabilidade do <i>TinyML</i>	64
4.5	Análise da Escalabilidade e Facilidade de Implementação de cada Sistema	64
4.6	Comparação Final	65
5	CONSIDERAÇÕES FINAIS	67
	REFERÊNCIAS	68

1 Introdução

Descrita pela primeira vez em 1817 pelo cirurgião britânico James Parkinson (JAMES, 1817), a Doença de Parkinson (DP) é ocasionada pela degradação dos neurônios dopaminérgicos. Esse efeito é associado, principalmente, ao acúmulo de Corpos de Lewy na região dos núcleos da base do mesencéfalo - em específico, na substância negra, estrutura responsável por modular a via cortical, impactando diretamente o controle de movimento (HOLMES et al., 2024). Isso ocorre porque os Corpos de Lewy possuem agregados proteicos de α -sinucleína que, quando mal-dobrados ou mutantes, induzem os neurônios ao estresse oxidativo, o que implica na morte irreversível e contínua dessas células (KALIA; LANG, 2015). Por essas razões, a DP é classificada como uma doença neurodegenerativa, crônica e progressiva.

A baixa produção do neurotransmissor dopaminérgico provocada pela morte de seus neurônios geradores acarreta na relutância cinética, já que a dopamina é responsável pela modulação da motricidade voluntária exercida pelos neurônios da substância negra. Consequentemente, surgem os sintomas característicos da DP: a bradicinesia, tônus muscular e tremores; além de sintomas não motores como disfunções cognitivas e a apatia (KLEIN et al., 2019).

Devido ao perfil da doença e aos múltiplos fatores que a impactam (e.g. idade, nível de atividade física, condições pré-existentes), a classificação de sua gravidade muitas vezes se baseia na observação visual do médico e em sua experiência. Assim, os métodos mais utilizados para quantificar a eficácia do tratamento e o avanço da doença trabalham a partir da manifestação dos sintomas, como é o caso da escala de Hoehn e Yahr (HY) (HOEHN; YAHR, 1967) ou através da Escala Unificada de Avaliação da Doença de Parkinson (*Unified Parkinson's Disease Rating Scale* - UPDRS) (Movement Disorder Society Task Force on Rating Scales for Parkinson's Disease, 2003). Quantificadores desse tipo são de extrema importância para o entendimento comportamental do indivíduo e da patologia (PAHAPILL; LOZANO, 2000). Capturar os níveis de tremor e realizar suas análises, com ou sem o efeito do medicamento Levodopa (L-Dopa), pode contribuir significativamente para a compreensão da fisiopatogênese e do funcionamento cerebral do paciente.

Neste contexto, constata-se o uso emergente de tecnologias assistivas para diagnóstico e acompanhamento de doenças e outras condições médicas. Em especial, destacam-se os dispositivos compostos por sistemas embarcados vestíveis - os chamados *Wearables*. A proposta desses mecanismos é de atuarem como um quantificador objetivo e concreto para a análise patológica, eliminando possíveis subjetividades e imprecisões advindas de uma classificação baseada majoritariamente em observações humanas.

Especificamente para a DP, como constatado por Rovini e outros (2017), sensores como giroscópios e acelerômetros em *wearables* possuem grande potencial no contexto da

doença, já que permitem a captura de movimentações e suas intensidades, o que condiz com os métodos requeridos pelas principais escalas avaliativas.

Baseado nessas informações, o presente trabalho aborda o desenvolvimento e comparação de dois *wearables* braceletes diferentes, os quais contêm sensores de medição inercial (como giroscópio, acelerômetro e magnetômetro) para captura dos tremores na região do punho e antebraço. A intenção é preparar, validar e comparar os dois dispositivos antes de serem aplicados clinicamente para captura do comportamento motor dos pacientes, auxiliando na definição do melhor protótipo para essa situação. Nesse sentido, enquanto um dos braceletes é equipado com uma placa de última geração do tipo Arduino Nano 33 BLE Sense Rev 2 (referenciada ao longo do trabalho simplesmente como "Arduino Nano"), a qual já possui todos os sensores necessários integrados, mas a um custo relativamente alto, o outro bracelete usufrui de uma placa mais simples, mas extremamente acessível - o Arduino UNO -, a qual exige a integração dos sensores de forma adicional. Vale ressaltar que, apesar dos dispositivos finais já, teoricamente, serem capazes de realizar as coletas de movimentos em pacientes, o presente trabalho se dedica, exclusivamente, ao desenvolvimento dos protótipos e comparação teórica dessas ferramentas, de forma que nenhum teste executado envolve dados e participação humanos. Além disso, em termos de construção, para não contrapor a proposta de custo reduzido atrelada ao Arduino UNO, a unidade de sensores adicionada ao respectivo sistema utiliza apenas acelerômetro e giroscópio - a adição de um magnetômetro elevaria o custo do protótipo e invalidaria o caráter custo-benefício desejado.

Ainda assim, a presente pesquisa também apresenta técnicas e recursos atrelados ao Aprendizado de Máquina (*Machine Learning* - ML), que utilizam Redes Neurais Recorrentes (*Recurrent Neural Networks* - RNN)(SCHMIDT, 2019), mais especificamente a *Long Short-Term Memory* (LSTM)(STAUDEMEYER; MORRIS, 2019), uma arquitetura especial de RNN desenvolvida com o intuito de trabalhar com sequências de dados de comprimento variável, como as séries temporais. Como expandido a seguir, o objetivo desse desenvolvimento é preparar e sugerir um método para análise e identificação dos episódios de tremor compatível com os protótipos. Ademais, um dos recursos do Arduino Nano escolhido é o suporte ao *TinyML*, o que permitiria que o ML fosse embutido diretamente no dispositivo.

Vale ressaltar, ainda, que apesar de já existirem alguns dispositivos *wearables* para monitoramento de tremores em pacientes com Parkinson, a pesquisa valida sistemas capazes de integrar técnicas avançadas de ML a um design de *hardware* otimizado para custo e conforto, abordando a acessibilidade e a precisão na detecção de tremores. Sendo assim, o desempenho de cada plataforma é avaliado com base na precisão dos sensores, eficiência computacional, consumo energético, conforto e acessibilidade. Qualquer imprecisão na coleta de dados pode comprometer a análise realizada pelo modelo de ML, tornando essencial a escolha da melhor configuração para garantir resultados confiáveis. Além disso, a facilidade de implementação, a durabilidade em condições adversas e o custo total do sistema, incluindo *hardware* e sensores,

são considerados na comparação. A análise desses fatores possibilita uma escolha informada sobre a plataforma mais adequada, equilibrando precisão, eficiência, custo e usabilidade. Outro ponto importante é avaliar a escalabilidade da solução para aplicações em larga escala, como monitoramento remoto de pacientes ou estudos clínicos sobre a eficácia do tratamento.

1.1 Problemática

Considerando que a DP é uma patologia incurável, o acompanhamento terapêutico e monitoramento desses sintomas é imprescindível para a manutenção da saúde e qualidade de vida dos indivíduos diagnosticados. Por se tratar de uma doença neurológica, impactada por múltiplos fatores como idade, prática de atividades físicas e condições pré-existent, a gravidade e evolução dos sintomas são completamente individuais, de forma que as suas determinações acabam representando um desafio aos médicos (ARMSTRONG et al., 2020).

Dessa forma, urge um acompanhamento sólido, capaz de quantificar objetivamente o nível da doença e resposta do paciente ao tratamento. A questão é definir qual o melhor método para construir um dispositivo especializado em promover esse acompanhamento, visando precisão, com baixo consumo energético e não invasivo àqueles que passam pela terapia. Sendo assim, pondera-se qual das plataformas analisadas — Arduino Nano 33 BLE Sense Rev 2 ou Arduino UNO com sensores externos — apresenta melhor adequação na detecção de tremores do Parkinson e permite um monitoramento mais acurado do tratamento.

Essa questão engloba múltiplos aspectos técnicos e clínicos, tais como a precisão na captação dos tremores, a eficiência computacional do sistema, a viabilidade energética e a facilidade de implementação das plataformas. Além disso, envolve a hipótese de que a escolha do *hardware* pode influenciar diretamente a confiabilidade dos dados coletados e, conseqüentemente, a tomada de decisão clínica. Sendo assim, antes de uma aplicação clínica real, é crucial que um método e sistema otimizado seja definido.

1.2 Justificativa

A detecção e análise dos tremores associados à DP são fundamentais para aprimorar abordagens terapêuticas e ampliar o conhecimento sobre a patologia. O avanço na tecnologia de sensores e sistemas embarcados possibilita um monitoramento contínuo desses tremores, fornecendo dados detalhados que auxiliam na adaptação do tratamento às necessidades individuais de cada paciente.

Do ponto de vista técnico, a pesquisa explora uma revisão comparativa entre o Arduino Nano 33 BLE Sense Rev 2 e o Arduino UNO (com sensores adicionais) no contexto da precisão na captação de dados, facilidade de implementação, consumo energético e compatibilidade com métodos computacionais avançados. Essa investigação não apenas fortalece a aplicação

da eletrônica e da computação na área da saúde, mas também fornece diretrizes importantes para a escolha de plataformas adequadas em projetos futuros de monitoramento biomédico.

O uso particular de placas Arduino se dá pela sua característica *open-source*, que garante o baixo custo, a expansividade e a democratização das ferramentas, já que tanto sua programação quanto seu *hardware* são completamente públicos, e seu estudo, modificação e distribuição são liberados; além disso, o Arduino apresenta facilidade na integração com outros componentes (os quais são igualmente acessíveis), visto que processos como soldagem muitas vezes não são necessários. Isso implica em uma integração com módulos e sensores que não só são capazes de constituir um projeto avançado e preciso, mas também capazes de fazer isso de forma democrática e com valores reduzidos. A escolha dos modelos específicos foi baseada no contraste entre as placas: o Arduino Nano 33 BLE Sense Rev 2 propõe superioridade tecnológica com tamanho reduzido, sendo considerado uma das placas mais avançadas de Arduino atualmente, o que implica em um custo maior; enquanto o Arduino UNO representa uma plataforma canônica extremamente democratizada para a prototipagem de sistemas de complexidade fundamental e para a introdução à eletrônica embarcada.

Além do impacto clínico e tecnológico, a pesquisa apresenta soluções acessíveis e viáveis para o monitoramento dos tremores, considerando o custo-benefício das diferentes abordagens disponíveis. O crescimento do uso de dispositivos portáteis para fins médicos reforça a importância de avaliar opções que equilibrem desempenho e acessibilidade, tornando a tecnologia mais inclusiva e eficaz no contexto do acompanhamento de doenças neurológicas. Assim, o estudo contribui para a inovação na área da engenharia biomédica, promovendo avanços tanto na qualidade do tratamento quanto na viabilidade econômica de sua aplicação.

1.3 Objetivos

1.3.1 Objetivo Geral

Delimitar uma análise comparativa entre o Arduino Nano 33 BLE Sense Rev 2 e o Arduino UNO (munido de sensores externos), a fim de concluir qual das plataformas se mostra mais eficaz, eficiente e viável no contexto de sistemas assistivos para detecção de tremores da DP.

1.3.2 Objetivos Específicos

A partir da implementação e revisão de dois sistemas embarcados capazes de capturar, processar e analisar dados inerciais, busca-se compreender as vantagens e limitações de cada abordagem, bem como sua aplicabilidade em diferentes cenários clínicos e de pesquisa. Dessa forma, a investigação visa fornecer subsídios técnicos e científicos para futuras aplicações de dispositivos portáteis no monitoramento do sintomas motores de pacientes com Parkinson.

Dessa forma, para alcançar o objetivo geral proposto, este trabalho se desdobra nos seguintes objetivos específicos:

1. Desenvolver os protótipos de *hardware* e *software* dos dois sistemas embarcados para a aquisição de dados inerciais, sendo o primeiro baseado no Arduino UNO com sensores externos (acelerômetro e giroscópio) e o segundo utilizando o Arduino Nano 33 BLE Sense Rev 2 com seus sensores integrados.
2. Através de um protocolo de testes mecânicos controlados para simular os tremores, comparar a precisão e confiabilidade dos dados coletados utilizando o Arduino UNO com os sensores externos e o Arduino Nano 33 BLE Sense Rev 2 com seus sensores integrados, avaliando a capacidade de cada plataforma em captar tremores com fidelidade e consistência. Essa comparação considera aspectos como a sensibilidade dos sensores, a frequência amostral e a qualidade do sinal coletado, permitindo identificar a solução mais adequada para esse tipo de monitoramento.
3. Avaliar a viabilidade do uso dos sistemas embarcados para o monitoramento contínuo da doença, considerando fatores como o consumo energético, a portabilidade, a facilidade de integração com outras tecnologias e a capacidade de armazenamento e transmissão dos dados coletados. A análise desses aspectos permite determinar a adequação das plataformas para um possível uso clínico ou domiciliar, facilitando o acompanhamento remoto da progressão da doença.
4. Investigar arquiteturas de RNN (no caso, a LSTM) aplicadas à classificação de séries temporais de movimento humano e, a partir desta análise, propor um modelo de referência otimizado para a detecção de tremores e para os dispositivos propostos. Adicionalmente, analisar a viabilidade técnica da implementação deste modelo diretamente no Arduino Nano 33 BLE Sense Rev 2, avaliando o impacto das restrições de memória (RAM e *Flash*) e capacidade de processamento do microcontrolador, e explorando as metodologias de otimização (como quantização e poda de rede) oferecidas pelo ecossistema *TinyML* para uma futura classificação *on-device*.

1.4 Organização do Trabalho

O restante do documento está organizado da seguinte forma: a seção 2 apresenta conceitos sobre os sensores utilizados e a otimização de suas capturas, além de apresentar definições importantes sobre o método de ML aplicado. Toda a metodologia de desenvolvimento, construção dos protótipos e obtenção e tratamento dos dados são expostos no capítulo 3. Por fim, os resultados obtidos com as operações e as considerações finais são apresentados nas seções 4 e 5.

2 Fundamentação Teórica

2.1 *Wearables* aplicados na saúde

O surgimento dos dispositivos *wearables*, como *smartwatches*, pulseiras de *fitness* e dispositivos de monitoramento de saúde, tem transformado significativamente a maneira como as doenças são diagnosticadas e acompanhadas. Esses instrumentos permitem o monitoramento contínuo dos sinais vitais e do comportamento do usuário em tempo real, oferecendo informações valiosas tanto para profissionais de saúde quanto para pacientes.

No estudo publicado por Zambotti e outros (2023), foram utilizadas técnicas de classificação de padrões para rastreamento do sono na busca de avaliar os dispositivos atuais aplicados a essa finalidade, respondendo questionamentos como: Que dados os dispositivos vestíveis de rastreamento do sono podem fornecer? Quão precisos são esses dados? O que deve ser levado em conta ao incorporar dispositivos vestíveis de rastreamento do sono em pesquisas? Já no acompanhamento da DP, Ferreira e outros (2015) utilizaram sensores diurnos e noturnos durante 7 dias e um aplicativo denominado *SENSE-PARK* instalado em um celular para acompanhar os pacientes em busca de encontrar padrões de comportamento entre os 22 indivíduos que fizeram parte do estudo.

Warmerdam e outros (2020) realizaram um estudo semelhante, porém com uma amostra maior de participantes (24) e uma variedade expandida de sensores (5, os quais não foram especificados). O estudo teve como objetivo avaliar a viabilidade e a usabilidade de um sistema de monitoramento de movimento baseado em sensores vestíveis em pacientes com Parkinson. O sistema foi desenvolvido para fornecer uma avaliação quantitativa dos sintomas motores da doença, abrangendo tremores, bradicinesia e instabilidade postural.

Apesar da quantidade de trabalhos utilizando algum tipo de *wearable*, pouco fora encontrado voltado ao acompanhamento do indivíduo, principalmente utilizando Arduino e técnicas de ML. Outro ponto importante a ser destacado é que, em grande parte dos dispositivos de coleta de dados, sua administração parte do paciente, o que em determinadas situações é impossível dependendo da idade e do nível da doença, sem contar que dispositivos como *smartwatches* não possuem, na maioria das vezes, valores acessíveis para qualquer pessoa.

2.2 Arduino

Desenvolvido em 2005 por um grupo de acadêmicos no *Interaction Design Institute Ivrea*, a proposta do Arduino era atuar como uma ferramenta de baixo custo e de fácil uso para estudantes, artistas, e pessoas sem ou com baixa instrução em programação e *hardware*.

Para isso, fora desenvolvida uma placa serial, cujos componentes eram organizados e alinhados de forma a priorizar a educação, e não a viabilidade eletrônica (Arduino, 2021). Conforme sua implementação foi se tornando mais célebre, a placa Arduino passou a se moldar conforme as necessidades do público, lançando novos modelos com recursos ou características que permitiam sua diferenciação em relação às placas mais simples, o que impulsionou seu amplo uso em uma diversa gama de aplicações, tais como *wearables*, ferramentas para *IoT*, e sistemas embarcados (Arduino, 2018).

Hoje, a Arduino já possui diversos modelos de placas e uma comunidade global de usuários. Sua característica *open-source* - isto é, seu código fonte e a sua configuração de *hardware* são passíveis de estudo, modificação e distribuição - permite um crescimento rápido, acessível e amplo das possibilidades através da ferramenta.

Essa filosofia de *hardware* aberto resultou em uma vasta diversificação de placas, projetadas para atender a nichos de aplicação específicos. No contexto deste trabalho, que visa comparar soluções de prototipagem para sistemas embarcados com foco em eficiência energética e capacidade de processamento para inferência de dados, foram selecionados dois modelos representativos de distintas gerações e propósitos da plataforma: o Arduino UNO R3 e o Arduino Nano 33 BLE Sense Rev 2.

2.2.1 Arduino UNO R3

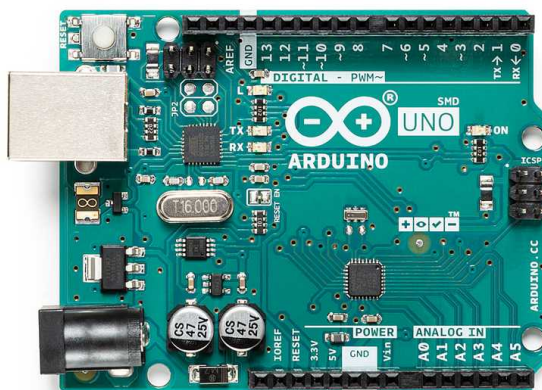
O Arduino UNO, especificamente em sua Revisão 3 (R3), é frequentemente considerado o modelo canônico da plataforma. Sua robustez, simplicidade e o vasto ecossistema de documentação e periféricos (denominados *shields*) consolidaram-no como o *benchmark* para prototipagem e ensino (Arduino, 2025b).

Este modelo é fundamentado no microcontrolador Atmel ATmega328P, uma arquitetura AVR de 8 bits operando a uma frequência de *clock* de 16 MHz. Dispõe de 32 KB de memória *Flash* para armazenamento do programa, 2 KB de SRAM para variáveis de tempo de execução e 1 KB de EEPROM. Embora essas especificações sejam modestas em comparação com microcontroladores modernos, são suficientes para uma vasta gama de aplicações que não exigem processamento intensivo de dados ou conectividade complexa.

Do ponto de vista energético, o perfil do UNO R3 reflete sua concepção como uma plataforma de desenvolvimento de bancada. A placa opera com uma lógica de 5V e, quando alimentada por uma fonte externa (via conector P4 ou pino VIN), utiliza um regulador de tensão linear (como o NCP1117) para reduzir a tensão de entrada para 5V. Reguladores lineares, embora simples, são inerentemente ineficientes, dissipando a diferença de tensão na forma de calor. O próprio ATmega328P não é otimizado para modos de baixíssimo consumo (comparado a arquiteturas mais recentes), tornando o UNO uma escolha menos viável para aplicações *wearables* ou que dependam estritamente de alimentação por bateria de longa duração.

A visão superior da placa está representada na figura 1:

Figura 1 – Arduino UNO



Fonte: *Arduino Documentation* (2025)

2.2.2 Arduino Nano 33 BLE Sense Rev 2

Em contrapartida, o Arduino Nano 33 BLE Sense Rev 2 representa a evolução da plataforma em direção aos domínios da IoT e da Inteligência Artificial embarcada (*TinyML*). Seu formato compacto (*nano form factor*) é apenas o primeiro indicador de sua vocação para sistemas embarcados e *wearables*.

O diferencial central desta placa reside em seu microcontrolador, o u-blox NINA-B306, que é baseado no potente SoC (*System-on-Chip* - "Sistema em Chip", em tradução livre) Nordic nRF52840 (Arduino, 2025a). Este componente utiliza uma arquitetura ARM Cortex-M4 de 32 bits, operando a 64 MHz, e oferece recursos de processamento e memória vastamente superiores aos do UNO: 1 MB de memória *Flash* e 256 KB de SRAM. Essa capacidade robusta é fundamental para a execução de modelos de ML diretamente no dispositivo (*on the edge*).

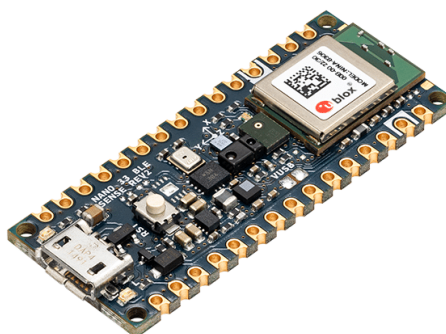
O sufixo "Sense" indica sua principal característica em destaque no presente projeto: a integração de um conjunto de sensores de alta precisão. Destaca-se a Unidade de Medição Inercial (IMU) de 9 eixos (LSM9DS1 ou similar), que combina um acelerômetro, um giroscópio e um magnetômetro, permitindo a captura complexa de dados cinéticos e de orientação. Além da IMU, a placa inclui sensores de temperatura, pressão, umidade, luz ambiente, cor, proximidade e um microfone PDM.

Criticamente para este estudo, o perfil energético do Nano 33 BLE Sense é otimizado para baixo consumo. Primeiramente, a placa opera com uma lógica de 3.3V, um padrão mais moderno e eficiente. Em segundo lugar, o SoC nRF52840 foi especificamente projetado para eficiência, oferecendo modos de sono profundo (*deep sleep*) avançados e gerenciamento de energia granular. Por fim, a presença nativa do *Bluetooth Low Energy* (BLE) 5.0 permite a comunicação sem fio com um dispêndio energético significativamente menor do que os

protocolos de comunicação tradicionais, como o Wi-Fi ou o *Bluetooth* Clássico. Vale ressaltar, contudo, que tanto o *deep sleep* quanto o BLE não foram utilizados nesta pesquisa comparativa.

A visão superior dessa placa está representada na figura 2:

Figura 2 – Arduino Nano



Fonte: *Arduino Documentation* (2025)

2.2.3 Síntese Comparativa

A seleção desses dois modelos permite um contraste direto entre uma abordagem de prototipagem clássica, robusta, mas energeticamente ineficiente (UNO), e uma abordagem moderna, altamente integrada e otimizada para aplicações *wearables* e de baixo consumo (Nano 33 BLE Sense). A análise subsequente investigará como essas diferenças arquitetônicas fundamentais impactam o consumo de energia e a viabilidade de implementação de algoritmos de ML em cada protótipo.

2.3 Funcionamento dos Sensores

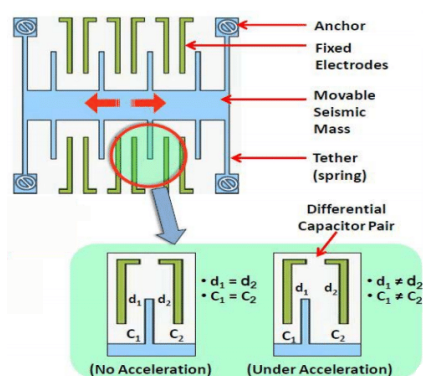
No contexto do projeto, para a quantificação de deslocamento destacam-se a Unidade de Medição Inercial (*Inertial Measurement Unit* - IMU), embarcada no Arduino Nano, que dispõe de uma junção de acelerômetro, magnetômetro e giroscópio, e a Unidade de Processamento Motora (*Motion Processing Unit* - MPU), anexada ao Arduino UNO. Os três (ou dois, no caso da MPU) sensores combinados permitem aferir a direção, o ângulo e a aceleração dos tremores, garantindo a completude e fidelidade dos registros.

Os acelerômetros são amplamente utilizados em aplicações governamentais, tecnológicas, industriais e de localização; em especial, em dispositivos móveis como celulares, relógios e GPS. Como o nome propõe, a função do acelerômetro é mensurar a mudança na velocidade, ou seja, a aceleração. Isso garante a medição de tremores, mudança de direção, frenagem ou precipitação do movimento, entre outros.

No caso, os módulos acelerômetro/giroscópio BMI270 (no caso do Arduino Nano) e MPU-6050 (utilizado em conjunto com o Arduino UNO) são do tipo capacitivo MEMS (*Micro-Electro-Mechanical-Systems*, em português Sistemas Micro-eleto-mecânicos) (BOSCH SENSORTEC, 2019).

Sobre o acelerômetro, a determinação do movimento reside na mudança da capacitância elétrica em resposta à aceleração. Para isso, são utilizados capacitores de placas paralelas, em que uma das placas é fixa e a outra é atrelada a uma massa com movimento livre, de forma que, com a aceleração, a distância entre essas placas aumenta ou diminui, suscitando a mudança de capacitância (VENKATANARAYANAN; SPAIN, 2014). Tal comportamento é descrito visualmente através da figura 3:

Figura 3 – Acelerômetro capacitor MEMS

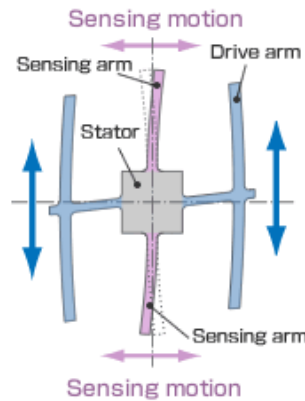


Fonte: AMERINI *et al.* (2016).

O giroscópio, também pertencente às unidades de medição de cada caso, opera com base nos princípios da inércia e do efeito Coriolis. O efeito Coriolis surge quando uma massa se desloca em uma direção específica com uma determinada velocidade e um movimento angular externo é aplicado a ela. Isso resulta na geração da chamada força de Coriolis, que induz um deslocamento perpendicular da massa (FRAUTSCHI; OLENICK; APOSTOL, 2007). No giroscópio, um conjunto de quatro massas de prova é mantido livre. Quando um movimento angular é aplicado, o efeito Coriolis causa uma alteração na capacitância entre as massas, dependendo do eixo do movimento angular, como demonstra a figura 4. Essa mudança na capacitância é detectada e, em seguida, convertida em uma leitura final (PASSARO *et al.*, 2017).

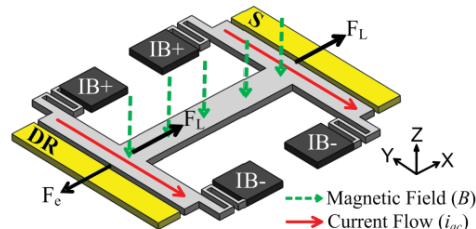
Além disso, o magnetômetro do tipo BMI150 (aplicado exclusivamente no caso do Arduino Nano), utiliza o efeito Hall (HALL, 1879) para suas operações. O efeito Hall é um fenômeno físico que ocorre quando uma corrente elétrica flui por meio de um condutor sob a influência de um campo magnético externo, resultando em uma diferença de potencial perpendicular tanto à direção da corrente quanto ao campo magnético aplicado, como visível na f 5.

Figura 4 – Giroscópio MEMS



Fonte: MADHAVA *et al.* (2023).

Figura 5 – Esquematização do Efeito Hall



Fonte: SONER SONMEZOGLU *et al.* (2017).

2.4 Cálculo do Movimento e Filtro de Kalman

2.4.1 Processamento e Obtenção de Grandezas Físicas dos Sensores Inerciais

A funcionalidade de uma IMU, como a presente no Arduino Nano 33 BLE Sense Rev 2, depende da capacidade de seus sensores de quantificar o movimento e a orientação. Os sensores baseados em MEMS, como acelerômetros, giroscópios e magnetômetros, não fornecem diretamente unidades físicas como m/s^2 ou graus. Em vez disso, eles geram sinais analógicos (normalmente variações de tensão ou capacitância) que são processados internamente e convertidos em um valor digital bruto por um Conversor Analógico-Digital (ADC) integrado, geralmente com uma resolução de 16 bits.

O "cálculo" para obter a variação em uma grandeza física é, portanto, um processo de conversão que transforma esse número digital adimensional em uma unidade métrica significativa. Este processo é fundamentado em dois parâmetros-chave: a sensibilidade (ou fator de escala) do sensor e o viés (ou *bias* / *offset*).

A sensibilidade (S) define quantos "pontos" (ou LSBs - *Least Significant Bits*) da saída digital correspondem a uma unidade física. Este valor é determinado pela configuração do Fundo de Escala (FSR - *Full-Scale Range*) selecionada pelo usuário. Por exemplo, um ADC de 16 bits possui um intervalo total de $2^{16} = 65536$ valores (de -32768 a +32767). Se o FSR for

configurado para $\pm 2g$, a sensibilidade (S_A) é dada pela equação 2.1.

$$S_A = \frac{65536 \text{ LSB}}{(2g - (-2g))} = \frac{65536 \text{ LSB}}{4g} = 16384 \text{ LSB/g} \quad (2.1)$$

O viés (B) é um deslocamento de zero; idealmente, um sensor em repouso e sem influências externas deveria registrar zero. Na prática, imperfeições de fabricação resultam em um valor não nulo ($Valor_{\text{raw,repouso}}$). A calibração é o processo de determinar este viés para subtraí-lo das leituras subsequentes.

A fórmula geral para a conversão de um valor bruto ($Valor_{\text{raw}}$) de qualquer eixo para sua unidade física ($Valor_{\text{físico}}$) é dada pela equação 2.2.

$$Valor_{\text{físico}} = \frac{Valor_{\text{raw}} - B}{S} \quad (2.2)$$

A seguir, detalha-se a aplicação deste princípio para cada sensor.

2.4.1.1 Acelerômetro: Cálculo da Aceleração Própria

O acelerômetro mede a aceleração própria (ou gravito-inercial) em seus três eixos ortogonais (X, Y, Z). Esta medição inclui tanto a aceleração linear do dispositivo quanto o vetor da gravidade terrestre ($\vec{G}$) (PETER, 2025b).

Conforme a equação geral, o valor da aceleração em unidades de g (onde $1g \approx 9.81 \text{ m/s}^2$) para um eixo específico é obtido pela equação 2.3.

$$A_g = \frac{Valor_{\text{raw,A}} - B_A}{S_A} \quad (2.3)$$

Onde:

- A_g é a aceleração calculada em g .
- $Valor_{\text{raw,A}}$ é a leitura digital bruta do ADC do acelerômetro.
- B_A é o viés (offset) do acelerômetro para aquele eixo, determinado durante a calibração (idealmente, o valor lido quando o eixo está perfeitamente perpendicular ao vetor gravitacional).
- S_A é a sensibilidade (ex: 16384 LSB/g para $\pm 2g$).

Para obter a aceleração em unidades do Sistema Internacional (m/s^2), o valor é multiplicado pela constante gravitacional (g), como indicado na equação 2.4.

$$A_{\text{m/s}^2} = A_g \times 9.81 \quad (2.4)$$

É crucial notar que, quando o dispositivo está estacionário, o vetor de aceleração $\vec{A} = [A_x, A_y, A_z]$ não será nulo. Ele irá registrar a reação ao vetor da gravidade. Por exemplo, em repouso sobre uma superfície plana, um dispositivo perfeitamente alinhado registraria $\vec{A} = [0, 0, 1]g$. A variação na leitura deste sensor, portanto, reflete mudanças na orientação do dispositivo em relação à gravidade ou mudanças em sua velocidade linear.

2.4.1.2 Giroscópio: Cálculo da Velocidade Angular e Integração da Orientação

O giroscópio mede a velocidade angular (ω), ou a taxa de rotação, em torno dos três eixos (conhecidos como *roll*, *pitch* e *yaw*). A unidade padrão é graus por segundo (dps ou $^\circ/s$) ou radianos por segundo (rad/s) (PETER, 2025a).

Similarmente ao acelerômetro, o cálculo da velocidade angular baseia-se na sensibilidade (S_G) definida pelo FSR (ex: ± 250 dps, ± 500 dps, ± 2000 dps). Para um FSR de ± 250 dps e um ADC de 16 bits, a sensibilidade é expressa pela equação 2.5.

$$S_G = \frac{65536 \text{ LSB}}{500 \text{ dps}} \approx 131.072 \text{ LSB/dps} \quad (2.5)$$

O cálculo da velocidade angular (ω) para um eixo é definido pela equação 2.6.

$$\omega_{\text{dps}} = \frac{\text{Valor}_{\text{raw,G}} - B_G}{S_G} \quad (2.6)$$

Onde B_G é o viés do giroscópio, determinado pela média das leituras do sensor enquanto este permanece em completo repouso por um período.

Diferente do acelerômetro, o giroscópio mede uma *taxa de variação*. Para obter a *variação angular* (isto é, a mudança na orientação ou o ângulo θ), é necessário realizar uma integração numérica da velocidade angular ao longo do tempo. Utilizando a soma de Riemann (a forma mais simples de integração discreta), o novo ângulo (θ_k) é calculado a partir do ângulo anterior (θ_{k-1}) e da leitura atual da velocidade angular (ω_k) multiplicada pelo intervalo de tempo entre as amostras (Δt), como indicado na equação 2.7.

$$\theta_k = \theta_{k-1} + (\omega_k \times \Delta t) \quad (2.7)$$

Este cálculo é fundamental, mas também é a principal fonte de erro do giroscópio. Pequenos erros no viés (B_G) são acumulados a cada passo da integração, resultando em um fenômeno conhecido como *drift* (deriva), onde a orientação calculada diverge progressivamente da orientação real ao longo do tempo.

2.4.1.3 Magnetômetro: Cálculo do Vetor do Campo Magnético

O magnetômetro mede a intensidade e a direção do campo magnético ambiente local, incluindo o campo magnético da Terra, nos três eixos. A unidade de medida mais comum é o microtesla (μT) ou o miligauss (mG) (PETER, 2025c).

O processo de cálculo inicial segue o mesmo padrão, como mostra a equação 2.8.

$$M_{\mu T} = \frac{Valor_{\text{raw},M} - B_M}{S_M} \quad (2.8)$$

Onde $M_{\mu T}$ é a intensidade do campo em microteslas, $Valor_{\text{raw},M}$ é a leitura bruta, B_M é o viés do sensor, e S_M é a sensibilidade (ex: 6.842 LSB/ μT para um FSR de $\pm 4800\mu\text{T}$).

2.4.2 Fusão de Sensores e o Filtro de Kalman

A obtenção de dados de orientação (atitude) precisos e robustos a partir de uma IMU não é um processo trivial, pois está sujeita às limitações físicas de cada sensor embarcado. A solução para este desafio reside na fusão de sensores, uma técnica que combina dados de múltiplas fontes para produzir uma estimativa mais precisa do que seria possível com qualquer um dos sensores isoladamente (FEIGIN et al., 2020). No contexto de sistemas embarcados e rastreamento de movimento, o Filtro de Kalman é, historicamente e atualmente, um dos algoritmos de estimação estocástica mais proeminentes e eficazes para essa finalidade.

Desenvolvido por Rudolf E. Kálmán, o Filtro de Kalman é um estimador recursivo ótimo, projetado para inferir os parâmetros internos de um sistema dinâmico linear quando este é perturbado por ruído Gaussiano. Sua aplicabilidade neste projeto é mitigar a dicotomia fundamental entre os sensores inerciais (giroscópio) e os sensores de referência absoluta (acelerômetro e magnetômetro).

No caso, os sensores da IMU apresentam características de erro complementares, o que os torna ideais para a fusão:

1. **Giroscópio (Medição de Curto Prazo):** O giroscópio mede a velocidade angular (ω). Para obter a orientação (ângulo θ), é necessária a integração numérica de suas leituras ao longo do tempo (conforme Equação 2.7). Este processo é altamente preciso para rastrear mudanças rápidas de orientação, sendo imune a acelerações lineares externas. Contudo, qualquer pequeno erro de viés (*offset* ou B_G), mesmo após a calibração, é acumulado a cada passo da integração. Este *drift*, faz com que a estimativa do ângulo divirja ilimitadamente da realidade ao longo do tempo.
2. **Acelerômetro (Referência Absoluta para *Roll* e *Pitch*):** Em um estado estático (ou de velocidade constante), a única aceleração medida pelo acelerômetro é o vetor da gravidade terrestre ($\vec{G}$). A orientação deste vetor em relação aos eixos do sensor

pode ser usada, através de funções trigonométricas (como atan2), para calcular uma estimativa absoluta e livre de deriva dos ângulos de inclinação: *roll* (eixo X) e *pitch* (eixo Y). O problema é que o acelerômetro não consegue distinguir a aceleração gravitacional de acelerações externas (lineares), como as induzidas pelo próprio tremor que se deseja medir. Durante o movimento, essas acelerações corrompem a medição, tornando-a uma fonte de ruído de alta frequência e de picos espúrios para o cálculo da orientação.

3. **Magnetômetro (Referência Absoluta para Yaw):** De forma análoga, o magnetômetro mede o campo magnético terrestre, fornecendo um vetor de referência (similar a uma bússola) para o ângulo de guinada (*yaw*, eixo Z). Embora estável a longo prazo, é extremamente suscetível a distorções magnéticas (interferências *hard* e *soft-iron*) do ambiente circundante, tornando suas leituras ruidosas e, por vezes, inconfiáveis.

Conclui-se que o giroscópio é confiável em curto prazo, mas deriva em longo prazo. O acelerômetro e o magnetômetro são estáveis em longo prazo, mas ruidosos e inconfiáveis em curto prazo (ou durante o movimento). O Filtro de Kalman é o mecanismo matemático que soluciona esta dualidade.

Dessa forma, o Filtro de Kalman opera em um ciclo recursivo de duas fases para estimar o estado do sistema (neste caso, os ângulos de orientação).

1. **Fase de Predição (Time Update):** Nesta fase, o filtro utiliza o modelo matemático do sistema e a última estimativa de estado confiável para *predizer* qual será o próximo estado. No contexto da IMU, o modelo do sistema é a integração do giroscópio. O filtro "confia" na leitura do giroscópio (ω_k) para projetar o ângulo atual ($\hat{\theta}_{k-1}$) para o próximo instante de tempo ($\hat{\theta}_k^-$), conforme a equação 2.9.

$$\hat{\theta}_k^- = \hat{\theta}_{k-1} + (\omega_k - B_G)\Delta t \quad (2.9)$$

(Onde $\hat{\theta}_k^-$ é a estimativa *a priori* e B_G é o viés estimado do giroscópio). Simultaneamente, o filtro atualiza sua própria incerteza (a covariância do erro, P_k^-), aumentando-a. Essencialmente, o filtro reconhece que, ao confiar no giroscópio, ele também está acumulando a incerteza associada à deriva (representada pelo ruído do processo, Q).

2. **Fase de Correção (Measurement Update):** Nesta fase, o filtro incorpora uma nova medição de referência absoluta (z_k) — o ângulo calculado a partir do acelerômetro ou magnetômetro. O filtro então compara essa medição ruidosa com sua predição interna. A diferença entre eles é chamada de inovação ou residual.

O "núcleo" do algoritmo é o Ganho de Kalman (K_k), um valor recalculado a cada iteração que funciona como um peso de fusão. Este ganho é determinado pela razão entre

a incerteza da predição do filtro (P_k^-) e a incerteza da medição do sensor (R , o ruído da medição).

A estimativa final (chamada *a posteriori*, $\hat{\theta}_k$) é então calculada como uma média ponderada, em acordo com a equação 2.10.

$$\hat{\theta}_k = \hat{\theta}_k^- + K_k(z_k - \hat{\theta}_k^-) \quad (2.10)$$

A lógica do Ganho de Kalman é a seguinte:

- Se a incerteza da medição R (ruído do acelerômetro/magnetômetro) for muito alta em relação à incerteza da predição P_k^- (o filtro confia em sua predição baseada no giroscópio), o Ganho de Kalman (K_k) será próximo de zero. O filtro irá, efetivamente, ignorar a medição e manter sua predição.
- Se a incerteza da predição P_k^- (incerteza acumulada da deriva) for alta em relação à incerteza da medição R (o acelerômetro está estável), o Ganho de Kalman (K_k) será próximo de um. O filtro irá confiar mais na medição, corrigindo sua predição e anulando a deriva.

O resultado é uma estimativa final que combina a responsividade de curto prazo do giroscópio com a estabilidade de longo prazo do acelerômetro e magnetômetro, filtrando o ruído de alta frequência de um e a deriva de baixa frequência do outro.

A eficácia do filtro depende diretamente da correta modelagem dos ruídos do sistema, representados pelas matrizes de covariância Q (ruído do processo, associado à deriva do giroscópio) e R (ruído da medição, associado ao acelerômetro e magnetômetro).

Neste trabalho, as variâncias que compõem essas matrizes foram calculadas a partir das especificações de precisão (ruído) documentadas nos *datasheets* dos sensores. Assumindo que a precisão x documentada (ex: 0.122 mg) representa a meia-amplitude ($\pm x$) de uma distribuição de ruído uniforme e contínua, a variância (σ^2) para essa distribuição é obtida pela equação 2.11.

$$\sigma^2 = \frac{((x) - (-x))^2}{12} = \frac{(2x)^2}{12} = \frac{4x^2}{12} = \frac{x^2}{3} \quad (2.11)$$

Os valores de precisão (x) utilizados foram: 0.122 mg para o acelerômetro; 70 mdps (miligráus por segundo) para o giroscópio; e 0.014 μT para o magnetômetro. A relação entre estas variâncias sintoniza o filtro, determinando o equilíbrio de confiança entre os sensores para otimizar a estimativa de orientação.

2.5 Aprendizado de Máquina

O ML consolida-se como um subcampo proeminente da Inteligência Artificial (IA), dedicado ao desenvolvimento de algoritmos e modelos computacionais que permitem a um

sistema aprender e aprimorar seu desempenho a partir da análise de dados, sem a necessidade de ser explicitamente programado para cada tarefa. A essência do ML reside na sua capacidade de identificar padrões, extrair insights e realizar previsões ou tomar decisões de forma autônoma, transformando dados brutos em conhecimento acionável (MITCHELL, 1997).

Historicamente, as raízes do ML remontam aos primórdios da computação. Alan Turing, em 1950, com seu "Teste de Turing", já questionava se as máquinas poderiam exibir comportamento inteligente, lançando as bases conceituais para a área (TURING, 1950). Pouco depois, em 1952, Arthur Samuel desenvolveu um programa de jogo de damas capaz de aprender com a experiência, sendo um dos primeiros exemplos práticos de ML (SAMUEL, 1959). O desenvolvimento do Perceptron por Frank Rosenblatt em 1957 marcou outro avanço crucial, estabelecendo um modelo matemático para uma rede neural de camada única (ROSENBLATT, 1958). Contudo, a área enfrentou períodos de estagnação, conhecidos como "invernos da IA", devido às limitações computacionais da época e a complexidades teóricas ainda não resolvidas.

O renascimento e a subsequente ascensão exponencial do ML nas últimas décadas foram impulsionados por dois fatores convergentes: a disponibilidade massiva de dados (Big Data), gerada pela digitalização global, e o avanço significativo no poder de processamento computacional, notadamente com o desenvolvimento de Unidades de Processamento Gráfico (GPUs). Essa confluência permitiu que algoritmos complexos, como as redes neurais profundas (*Deep Learning*), fossem treinados em vastos conjuntos de dados, alcançando níveis de performance antes inatingíveis em domínios como reconhecimento de imagem, processamento de linguagem natural e análise preditiva (GOODFELLOW; BENGIO; COURVILLE, 2016).

No cerne do ML estão conceitos fundamentais que definem sua operação. O processo inicia-se com um conjunto de dados, denominado conjunto de treinamento, que serve como base para o aprendizado do modelo. As variáveis ou características desses dados, conhecidas como *features*, são os atributos utilizados pelo algoritmo para identificar os padrões subjacentes. O objetivo é que, após a fase de treinamento, o modelo seja capaz de generalizar o conhecimento adquirido para tomar decisões ou fazer previsões precisas sobre dados novos e nunca antes vistos.

Inicialmente, o campo foi dominado por algoritmos clássicos, como Máquinas de Vetores de Suporte (SVM), Árvores de Decisão e modelos de regressão linear. Embora eficazes, esses métodos frequentemente dependiam de um processo manual e intensivo de engenharia de características (*feature engineering*), no qual especialistas humanos precisavam extrair e selecionar os atributos mais relevantes dos dados brutos para alimentar os modelos (BISHOP, 2006).

O ponto de inflexão na trajetória do ML ocorreu com a ascensão das redes neurais e, subsequentemente, do Aprendizado Profundo (*Deep Learning*). Inspiradas na estrutura neural do cérebro humano, as redes neurais profundas introduziram a capacidade de aprender representações hierárquicas dos dados de forma automática. Essa habilidade de extrair

características complexas diretamente dos dados brutos — como *pixels* de uma imagem ou palavras de um texto — eliminou grande parte da necessidade de engenharia de características manual e permitiu um salto de performance em tarefas de alta complexidade (GOODFELLOW; BENGIO; COURVILLE, 2016).

Essa transição de modelos que necessitavam de características pré-processadas para modelos que aprendem as próprias características representa uma mudança de paradigma fundamental. Hoje, o ML não apenas automatiza a criação de modelos preditivos, mas também a própria descoberta do que é relevante nos dados, tornando-se uma ferramenta indispensável e transformadora na ciência e na indústria contemporânea.

2.6 Redes Neurais Recorrentes

As RNNs representam uma classe de redes neurais artificiais projetada especificamente para lidar com dados sequenciais ou de séries temporais. Diferentemente das redes neurais *feedforward*, cuja informação flui em uma única direção, da entrada para a saída, as RNNs possuem conexões cíclicas, ou "*loops*", que lhes permitem manter um estado interno, ou memória, das informações processadas anteriormente (GOODFELLOW; BENGIO; COURVILLE, 2016).

Essa arquitetura recorrente é o que confere às RNNs sua capacidade de modelar dependências temporais e contextos em sequências. Em cada passo de tempo (*timestep*), a rede processa uma nova entrada da sequência e combina essa informação com o estado oculto do passo anterior. Esse estado oculto funciona como uma memória cumulativa que resume os eventos passados, influenciando a saída atual e o estado do próximo passo. Essa característica as torna particularmente adequadas para tarefas como processamento de linguagem natural (tradução automática, modelagem de linguagem), reconhecimento de fala e análise de séries temporais.

Contudo, as RNNs simples enfrentam um desafio significativo conhecido como o problema do gradiente que desaparece ou explode (*vanishing/exploding gradient*). Em sequências longas, torna-se difícil para a rede aprender e propagar dependências de longo prazo, pois os gradientes utilizados no treinamento podem diminuir exponencialmente até se tornarem insignificantes (desaparecimento) ou crescer descontroladamente (explosão), comprometendo a eficácia do aprendizado (BENGIO; SIMARD; FRASCONI, 1994). Para superar essa limitação, foram desenvolvidas arquiteturas mais sofisticadas, como as redes LSTM (HOCHREITER; SCHMIDHUBER, 1997).

2.6.1 *Long Short-Term Memory*

Introduzida por *Sepp Hochreiter* e *Jürgen Schmidhuber* em 1997 (HOCHREITER; SCHMIDHUBER, 1997), esse tipo de rede foi projetada para resolver o problema do gradiente

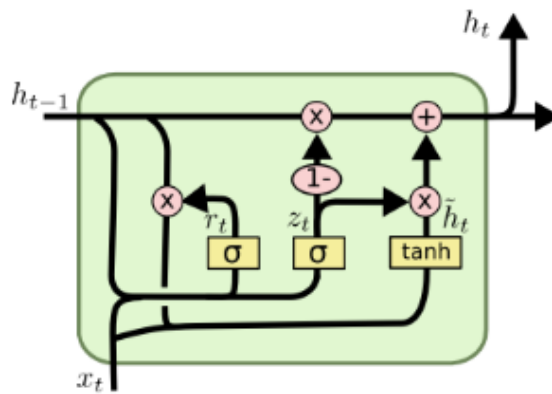
que desaparece (*vanishing gradient problem*) quando são treinadas em sequências longas.

A arquitetura da LSTM consiste em unidades de células de memória que são capazes de manter e acessar informações ao longo do tempo, tornando-as adequadas para modelar padrões complexos em séries temporais, justamente a necessidade do presente trabalho. A equação 2.12 representa o cálculo da célula LSTM.

$$\begin{aligned}
 i_t &= \sigma(W_{xi}x_t + W_{hi}h_{t-1} + W_{ci}c_{t-1} + b_i) \\
 f_t &= \sigma(W_{xf}x_t + W_{hf}h_{t-1} + W_{cf}c_{t-1} + b_f) \\
 g_t &= \tanh(W_{xg}x_t + W_{hg}h_{t-1} + b_g) \\
 o_t &= \sigma(W_{xo}x_t + W_{ho}h_{t-1} + W_{co}c_{t-1} + b_o) \\
 c_t &= f_t \odot c_{t-1} + i_t \odot g_t \\
 h_t &= o_t \odot \tanh(c_t)
 \end{aligned} \tag{2.12}$$

onde, i_t , f_t e g_t representam as portas de entrada, esquecimento, atualização e saída da célula LSTM, respectivamente. c_t é o vetor de célula de memória atualizado e h_t é o vetor de saída atualizado. Os símbolos x_t e h_{t-1} representam a entrada na etapa de tempo atual e a saída da célula LSTM na etapa de tempo anterior, respectivamente. W representa os pesos da rede e b representa os vieses. σ denota a função de ativação sigmoide e $\odot$ denota a multiplicação elemento a elemento. A figura 6 apresenta o comportamento descrito na equação 2.12.

Figura 6 – Funcionamento LSTM



Fonte: Understanding LSTMs (OLAH, 2015)

Esses mecanismos de controle de fluxo de informação, permitem que as LSTMs aprendam e mantenham dependências de longo prazo em dados sequenciais, tornando-as mais eficazes do que as RNNs tradicionais para muitas tarefas de ML, principalmente as tarefas envolvendo análise de séries temporais.

2.7 Trabalhos Relacionados

A principal fonte de trabalhos com proposta semelhante - isto é, *wearables* capazes de medir os tremores da DP - são os dispositivos comerciais. Entre eles, destacam-se o *CueBand* (WALKER et al., 2024), um monitor-alerta que evita o ptialismo ao lembrar os pacientes de engolir; o *Kinesia 360* (Great Lakes NeuroTechnologies, 2024), cujo foco é, assim como a atual proposta, a detecção dos tremores da DP; e o *Cala Trio* (Cala Health, 2024), que busca comedir esses tremores. Entre esses, ao analisarmos a função primária, o modo de construção, e os sensores utilizados, aquele que mais se aproxima da presente pesquisa é o *Kinesia 360*.

O *Kinesia 360* consiste em dois sensores - um utilizado no tornozelo e outro no punho - e, assim como o projeto, ele busca aplicar ML para classificação dos movimentos em tremores, além de permitir uma análise em tempo real, tanto por parte dos médicos quanto dos pacientes, já que possui um aplicativo *mobile* próprio. Além disso, pesquisas científicas que também buscam estudar o comportamento da L-Dopa e sua atuação na bradicinesia e nos tremores já utilizaram o *Kinesia 360* como instrumento medidor, como é o caso da publicada por Pulliam e outros (2018).

Contudo, assim como outros dispositivos focados no sintoma cinético, o *Kinesia 360* e o *Cala Trio* possuem um altíssimo custo e um acesso restrito e controlado. Dessa forma, os *wearables* propostos visam superar essas limitações dos instrumentos comerciais e contribuições existentes na literatura através de protótipos de baixo custo, mas ainda com aferições concretas, que possam ser aplicados como monitores dos tremores de forma plenamente acessível.

3 Materiais e Métodos

Esta seção descreve em detalhes todo o processo envolvendo a construção dos braceletes, as etapas de teste e a arquitetura da rede neural, bem como os métodos utilizados para avaliação e validação dos resultados, e os parâmetros comparativos adotados. O perfil da pesquisa, por sua vez, é quantitativo-exploratório, visto que ao mesmo tempo que utiliza métricas como sensibilidade de sensores, duração energética e frequência de amostragem, também há uma análise comparativa baseada em critérios técnicos e de desempenho, visando determinar a viabilidade dos sistemas.

3.1 Validação dos Sensores e Componentes

A primeira etapa do estudo envolveu a análise e a validação dos componentes eletrônicos propostos e sua adequação aos sistemas embarcados. A arquitetura de *hardware* foi concebida para ser compacta, de baixo consumo energético e com a capacidade de processamento e sensoriamento necessária para a aplicação. Os componentes centrais dos braceletes são os microcontroladores Arduino Nano 33 BLE Sense Rev2 e Arduino UNO, cujas características funcionais e energéticas individuais exigem montagens e componentes diferentes compondo cada sistema. A evolução de cada versionamento (n) dos protótipos serão denominadas 'pUNO_vn', para os protótipos que utilizam o Arduino UNO, e 'pNANO_vn', para os protótipos que utilizam o Arduino NANO 33 BLE Sense Rev 2.

A principal distinção entre as placas é a presença de sensores já integrados ao Arduino Nano 33 BLE Sense Rev 2, enquanto o Arduino UNO é munido apenas com o microcontrolador, de forma que o acelerômetro e o giroscópio são anexados à parte. Outro ponto importante trata sobre as dimensões de cada placa, já que o Arduino UNO possui cerca do quádruplo da área do Arduino Nano em questão. Por fim, ao modo em que o Arduino Nano 33 BLE Sense Rev2 exige uma voltagem de entrada de cerca de 3.3V, o pino de voltagem de entrada (Vin) do Arduino UNO aceita tensões entre 7-12V. Tais características são cruciais nas escolhas dos sensores, módulos e fontes de cada sistema, visto que a compatibilidade, custo e tamanho são decisivos no trabalho.

Inicialmente, os *wearables* seriam compostos, além dos Arduinos, por:

1. **Bracelete:** Devido sua adequação com as necessidades do projeto, fora utilizado um equipamento esportivo, chamado *Wrist Coach*, feito de tecidos ajustáveis e respiráveis (mistura de poliamida, poliéster e elastano), ocupando um terço do antebraço, e com abas plásticas e resistentes que permitem o alojamento do dispositivo de forma segura.

2. **Baterias CR2032 no suporte:** como fonte de alimentação para as placas usaria-se duas baterias CR2032, alinhadas em série.
3. **Módulo Micro SD:** responsável pelo armazenamento e transferência dos dados através de um cartão Micro SD.
4. **Módulo MPU6050 (exclusivo para o sistema do Arduino UNO):** módulo composto por acelerômetro e giroscópio, sendo então o componente responsável pela captura da movimentação no respectivo sistema.
5. **Step-Down AMS1117 (exclusivo para o sistema do Arduino NANO):** como as duas baterias CR2032 em série implicam em uma fonte maior que 3.3V, para alimentar o Arduino NANO seria necessário filtrar essa tensão. Tal componente é responsável por fazer esse filtro de forma eficiente, mesmo possuindo dimensões reduzidas.

Essas primeiras versões, postas como hipótese de construção, foram denominadas pUNO_v1 e pNANO_1.

A seleção da solução de alimentação, baseada em baterias do tipo CR2032, foi precedida por uma análise detalhada do orçamento de potência (*power budget*) do sistema. O objetivo era garantir que a fonte de energia escolhida pudesse, teoricamente, suprir a demanda de corrente do circuito durante a operação de coleta de dados. Entretanto, vale ressaltar, que essa fonte de alimentação teve de ser descontinuada logo nas primeiras etapas da pesquisa.

O componente de maior consumo em ambos os circuitos é o conjunto formado pelo microcontrolador e seus periféricos integrados. O consumo apenas do Arduino Nano 33 BLE Sense Rev 2, por exemplo, foi estimado entre 8 mA e 13 mA. Somam-se a este valor os sensores embarcados: BMI270 (acelerômetro e giroscópio) e o magnetômetro BMM150. Notavelmente, estes sensores são altamente eficientes, apresentando um consumo agregado marginal, inferior a 1.2 mA (menos de 0.7 mA para o BMI270 em alta taxa de amostragem e aproximadamente 0.5 mA para o BMM150 em modo normal de operação).

O segundo contribuinte mais significativo para o consumo energético é o módulo MicroSD, particularmente durante os ciclos de escrita. A operação de escrita de dados no cartão via protocolo SPI pode gerar picos de corrente transientes, estimados entre 30 mA e 50 mA. Contudo, é fundamental destacar que tais picos são de curtíssima duração, ocorrendo apenas no instante da gravação do bloco de dados, de forma que a corrente de pico, quando amortizada ao longo do tempo, traduz-se em um acréscimo médio de apenas 1 mA a 5 mA ao consumo total.

Utilizando duas baterias CR2032 conectadas em série para fornecer uma tensão nominal de 6V (necessária a regulação via AMS1117, no caso do Arduino Nano), a capacidade total de carga do conjunto permanece a de uma única célula, tipicamente em torno de 225 mAh. Com base no consumo médio estimado entre 15 mA e 25 mA, a autonomia do dispositivo pode ser

calculada. Em um cenário pessimista (25 mA), essa duração seria de 9 horas (225 mAh / 25 mA), de forma que, as baterias CR2032 seriam, na teoria, capazes de prover energeticamente o sistema.

Dessa forma, com a fonte energética já definida, iniciou-se a etapa de teste de componentes. A primeira seção dessa etapa compreendeu verificações de tensão, operação e funcionalidade para cada módulo do circuito. Por serem testes preliminares, cujo objetivo é a confirmação do pleno funcionamento dos componentes, os testes foram realizados com os Arduinos conectados ao computador, facilitando o retorno pela comunicação serial e a garantia de estabilidade energética.

O primeiro procedimento experimental consistiu na validação do subsistema de alimentação. Utilizando um multímetro de precisão, foram aferidas a tensão e a corrente de saída do soquete de baterias - no caso do pNANO_v1, tanto a montante quanto a jusante do regulador de tensão AMS1117. O objetivo primordial deste ensaio era confirmar a eficácia do regulador em reduzir a tensão nominal de 6V, proveniente da associação em série das duas células CR2032, para um nível estável e compatível com o pino de entrada de alimentação (Vin) do microcontrolador, especificado em 3.3V. A medição confirmou que o componente operava conforme o esperado, fornecendo uma tensão regulada e segura para o circuito.

Subsequentemente, a verificação funcional voltou-se para as unidades motoras (IMU e MPU). Para este fim, foi empregada a biblioteca de *software* `Arduino_BMI270_BMM150.h` para o Arduino Nano, e a biblioteca `MPU6050.h` desenvolvida pelo autor Jeff Rowberg para o Arduino UNO (2022). Executou-se códigos-exemplos, fornecidos pelas próprias bibliotecas, projetados para inicializar os sensores e transmitir suas leituras brutas via comunicação serial para o monitor do computador. O sucesso deste teste foi determinado pela recepção de um fluxo de dados contínuo e coerente de todos os eixos do acelerômetro, giroscópio (e magnetômetro), atestando seu pleno funcionamento.

De forma análoga, a funcionalidade dos módulos de armazenamento de dados foi avaliada. Recorreu-se a códigos de exemplo da biblioteca nativa do Arduino para cartões SD, que foram executados para realizar operações fundamentais. Os testes abrangeram a inicialização dos módulos, a criação de um arquivo de texto, a escrita de uma cadeia de caracteres neste arquivo e, por fim, a sua leitura para verificação da integridade dos dados. A conclusão bem-sucedida de todas as operações validou a capacidade dos módulos em realizar o armazenamento não-volátil de informações. Vale ressaltar, também, que apesar de possuírem o mesmo funcionamento, os módulos de cartão Micro SD utilizados para os dois Arduinos são, fisicamente, diferentes. O módulo utilizado em conjunto com o Arduino Nano é do tipo "mini", e opera em 3.3 V; enquanto o módulo do Arduino UNO possui cerca do dobro de tamanho, mas opera com uma voltagem compatível à respectiva placa (5V).

Uma vez que os componentes foram validados individualmente, o passo seguinte foi a integração dos subsistemas de sensoramento e armazenamento, simulando a operação central

do dispositivo. Para tanto, foi desenvolvido um programa específico cuja lógica consistia em adquirir dados da IMU/MPU a uma frequência de 5 leituras por segundo (5 Hz) durante um intervalo de 10 segundos. O conjunto de 50 amostras de dados resultante foi então formatado e gravado em um arquivo no cartão MicroSD. Essa programação serviu como um programa de teste padrão para todas as etapas subsequentes de desenvolvimento e depuração do circuito, sendo iterativamente refinada até a consolidação do *firmware* final.

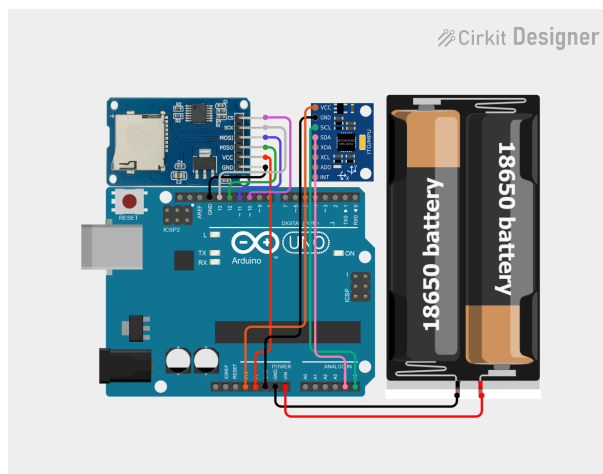
Com a validação individual e integrada dos componentes em ambiente controlado, a segunda seção de experimentação visou avaliar o desempenho do sistema em seu modo de operação autônomo. Nesta fase, o circuito foi desconectado do computador e, pela primeira vez, a alimentação foi provida exclusivamente pelo soquete com as duas baterias CR2032.

Apesar da análise teórica do orçamento de potência corroborar a adequação da fonte de energia selecionada, o resultado prático contradisse as estimativas. Ao ser ativado, o circuito integrado falhou em inicializar, indicando que as baterias do tipo moeda não foram capazes de suprir a demanda energética dos sistemas - nem para o Arduino UNO, nem para o NANO. Essa divergência entre o cálculo teórico e o comportamento empírico sugeriu que o problema não residia no consumo médio contínuo, mas sim na incapacidade da fonte em fornecer a corrente de pico (*inrush current*) necessária durante a partida simultânea do microcontrolador e dos periféricos.

A conclusão foi que a limitação era intrínseca à capacidade de descarga das células CR2032. A solução, portanto, exigia a substituição da fonte de alimentação. A alternativa escolhida foi a bateria de íon-lítio do tipo 18650, notória por sua alta densidade energética e por operar numa faixa de tensão entre 3.7V e 4.2V. Essa voltagem é diretamente compatível com o regulador interno presente na placa do Arduino, o que permitiu a remoção do regulador step-down AMS1117 do projeto, simplificando a arquitetura do circuito. Para o caso do Arduino UNO, bastou que duas baterias dessa fossem ligadas em série. O primeiro modelo 18650 testado, com capacidade nominal de 3500mAh, representou um avanço parcial. Verificou-se que esta bateria conseguia alimentar isoladamente o Arduino ou o módulo Micro SD, mas falhava ao tentar suprir a demanda de ambos simultaneamente. Este resultado reforçou a conclusão de que a corrente de partida combinada dos dois módulos era o fator limitante.

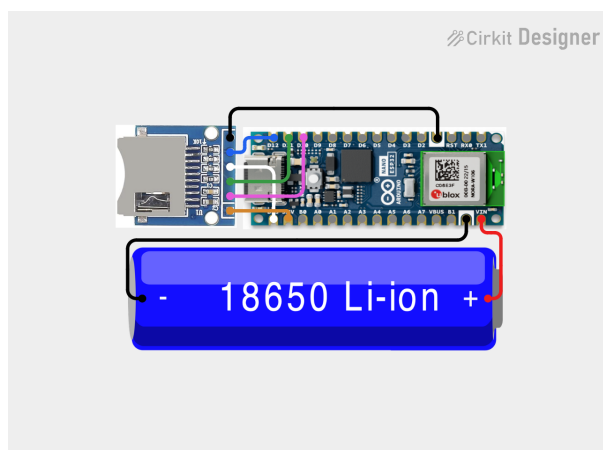
Dessa forma, a estratégia final foi adotar uma bateria 18650 de capacidade ainda maior, de 9800mAh. Com esta nova fonte, o sistema completo finalmente operou conforme o esperado, com o microcontrolador e o módulo de armazenamento funcionando de forma estável e conjunta, de maneira autônoma. Embora a autonomia total do dispositivo com esta configuração não tenha sido objeto de teste nesta etapa, o sucesso em ligar e manter o circuito funcional representou a superação de um obstáculo crítico e a definição da arquitetura de alimentação viável para a nova versão dos protótipos (pUNO_v2 e pNANO_v2). Os esquemáticos desses protótipos seguem na figura 7 e figura 8, respectivamente.

Figura 7 – Esquemático do pUNO_v2



Fonte: elaborada pelo autor.

Figura 8 – Esquemático do pNANO_v2



Fonte: elaborada pelo autor.

3.2 Funcionamento dos Sistemas

Com a arquitetura de *hardware* e a fonte de alimentação definitivamente estabelecidas, a etapa subsequente do projeto concentrou-se no desenvolvimento dos *firmwares* finais, responsáveis por orquestrar a aquisição, o processamento e o armazenamento das movimentações do antebraço dos usuários. Os códigos para cada Arduino, apresentados nos arquivos Final_Nano.ino e Final_UNO.ino (disponíveis na aba "Firmwares" no repositório:

<https://github.com/ManuChristofalo/Arduino_Parkinson_Comparison>), foram estruturados para garantir a robustez e a eficiência do sistema embarcado. Vale ressaltar que os códigos seguem exatamente a mesma estrutura (obviamente, com as adequações necessárias), mas com a inclusão da leitura do magnetômetro no caso do Arduino Nano (e, conseqüentemente, seu uso no Filtro de Kalman).

A lógica de armazenamento de dados foi otimizada para garantir a integridade dos registros e a eficiência do uso do cartão MicroSD. Primeiramente, para evitar a sobrescrita de

coletas anteriores, uma função é executada durante a inicialização para verificar os arquivos existentes no cartão e gerar um novo nome de arquivo sequencial, como COLETA1.csv, COLETA2.csv, e assim por diante. Em segundo lugar, para minimizar a latência e o desgaste do cartão associados a operações de escrita frequentes, foi implementado um sistema de *buffer*. As leituras processadas de orientação angular (*roll*, *pitch*, *yaw*) e o tempo decorrido são concatenados e armazenados temporariamente em uma variável do tipo String. A escrita no arquivo CSV só ocorre em lotes, a cada 10 amostras coletadas, o que corresponde a aproximadamente um segundo de dados.

3.2.1 Duração da Bateria

Com os *firmwares* finais consolidados e os *hardwares* definidos, a etapa subsequente da pesquisa consistiu na análise do orçamento de potência e na determinação da autonomia real do dispositivo. Para a análise teórica, adotou-se uma estimativa de consumo de corrente pessimista para os circuitos, da ordem de 100 mA. Considerando a fonte de alimentação selecionada -baterias do tipo 18650 com capacidade nominal de 9800 mAh (a corrente se mantém mesmo na configuração em série) - o cálculo de autonomia (resultante da divisão da capacidade da bateria pela corrente de consumo) previa uma duração de aproximadamente 98 horas, o que equivaleria a vários dias de operação contínua. Mesmo ao considerar capacidades menores, como 7500mAh, e um consumo de corrente significativamente maior, como 250mA (explicado através da energia dissipada através dos reguladores de tensão, por exemplo), a bateria ainda duraria mais de um dia.

Entretanto, para validar esta projeção, foram realizados três ensaios empíricos de duração. Nestes testes, os protótipos, executando o código final de coleta de dados, foram deixados em operação autônoma até a exaustão completa de diferentes unidades de bateria.

3.2.2 Soldagem

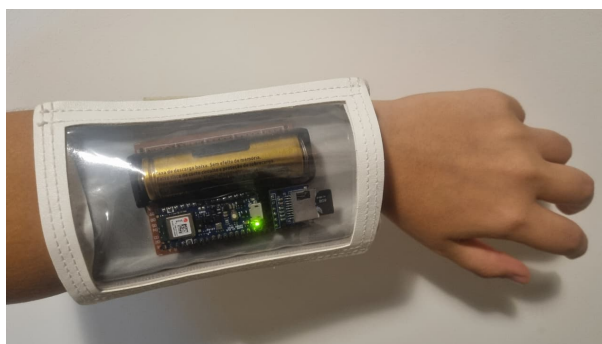
Com os componentes de *hardware* e os *firmwares* finalizados e validados, o projeto avançou para a etapa de testes simulados. Esta fase visava verificar o funcionamento do dispositivo em seu contexto de aplicação final: realizando coletas de dados de forma autônoma enquanto acondicionado no bracelete. Para garantir a robustez necessária para um dispositivo vestível, o primeiro passo foi transferir o circuito de uma *protoboard* para uma solução permanente, soldando as conexões em uma placa de prototipagem perfurada. Vale ressaltar que, para evitar o risco de danos térmicos e facilitar futuras manutenções, os componentes principais - os Arduinos, módulos Micro SD e MPU - não foram soldados diretamente à placa. Em vez disso, utilizou-se soquetes de pinos, que permitem a conexão e remoção prática e segura dos módulos.

Finalmente, a montagem realizada pelos autores foi efetiva, e todos os componentes

operaram conforme o esperado, concluindo a etapa de montagem do protótipo final. Para validar a robustez e a funcionalidade do sistema integrado, os primeiros testes estruturais foram conduzidos pelos próprios pesquisadores portando os dispositivos no bracelete e simulando movimentos diversos para exemplificar que o *wearable* realmente poderia ser utilizado e que o dispositivo estava bem acoplado. Para garantir o funcionamento dos circuitos agora soldados, o *firmware* foi executado; após a constatação da ativação dos Arduinos e da gravação bem-sucedida, os sistemas estavam prontos para coletas inerciais.

O exemplo de uso do sistema embarcado final do Arduino NANO, já soldado e no bracelete, segue na figura 9.

Figura 9 – Arduino Nano no Bracelete (pós-solda)



Fonte: elaborada pelo autor.

3.3 Protocolo de Testes Mecânicos Comparativos

Uma vez concluída a montagem e a validação funcional dos sistemas pUNO_v2 e pNANO_v2, detalhados nas seções 3.1 e 3.2, a pesquisa avança para a etapa de validação metrológica. Esta fase é crucial para endereçar o segundo objetivo específico deste trabalho: "comparar a precisão e confiabilidade dos dados coletados [...] avaliando a capacidade de cada plataforma em captar tremores com fidelidade e consistência".

Para realizar uma comparação quantitativa robusta, é imperativo submeter ambos os protótipos a um estímulo de entrada que seja idêntico, controlável e repetível. Visto que a variabilidade inerente aos testes clínicos excede o escopo desta validação de engenharia, optou-se por um "protocolo de testes mecânicos controlados", conforme proposto nos objetivos.

Para este fim, foi desenvolvida uma plataforma de testes (referida como *rig*) atuada por um servo motor. Este sistema gera oscilações senoidais programadas, simulando as características de frequência (tipicamente de 3 a 5 Hz na DP, mas utilizou-se 2 a 4,5 Hz) e amplitude dos tremores parkinsonianos. O sinal de comando enviado ao atuador estabelece a "verdade fundamental" (*ground truth* - GT) do experimento, permitindo uma aferição direta do erro de medição de cada protótipo.

Esta seção, portanto, descreve a montagem da plataforma de testes e o delineamento dos três ensaios comparativos executados: (A) um ensaio de varredura de frequência, para quantificar a precisão da detecção no domínio da frequência; (B) um ensaio de fidelidade de sinal, para analisar a acurácia da forma de onda no domínio do tempo; e (C), um teste estático de longa duração, para avaliar o ruído intrínseco e a estabilidade da orientação (deriva).

O *firmware*, códigos e resultados obtidos nessa e na próxima etapa ("Análise Comparativa dos Sinais Inerciais") também estão disponíveis no repositório, na pasta "Teste_Mecanico".

3.3.1 Montagem da Plataforma de Testes

Para gerar o estímulo mecânico controlado, foi construída uma plataforma de testes, ou "*rig*", composta por três subsistemas principais: o atuador, a plataforma de movimento e o sistema de controle.

Como atuador, foi selecionado o micro servo motor MG90S. A escolha deste componente se justifica por suas engrenagens metálicas, que oferecem maior precisão de posicionamento e menor *backlash* (folga mecânica) em comparação com servos de custo similar dotados de engrenagens plásticas, como o SG90, garantindo um movimento mais consistente para o GT (ZAITCHIK, 2024).

A plataforma de movimento, sobre a qual os protótipos sob teste seriam fixados, foi fabricada com uma chapa leve e rígida de MDF. Esta chapa foi então firmemente aparafusada a um dos anexos de hélice (*horn*) do servo motor, assegurando que o movimento angular do atuador fosse diretamente transferido para os sensores.

O controle do atuador foi gerenciado por um terceiro microcontrolador Arduino, dedicado exclusivamente à função de gerador de sinal. Este controlador foi programado para enviar os comandos de Pulso por Modulação de Largura (*Pulse Width Modulation* - PWM) ao servo, ditando a forma de onda (senoidal), a frequência e a amplitude do movimento simulado.

Para garantir a validade da comparação, ambos os protótipos, pUNO_v2 e pNANO_v2, foram montados um sob o outro na plataforma de MDF, alinhados no mesmo eixo - o Y, para *Pitch*. A fixação foi realizada utilizando fita adesiva dupla-face de alta aderência, complementada por fita isolante para assegurar uma montagem monolítica, eliminando qualquer possibilidade de movimento relativo ou vibração espúria entre os protótipos e a plataforma oscilante.

3.3.2 Ensaio A: Análise de Precisão em Frequência

O primeiro ensaio experimental teve como objetivo primário quantificar a precisão de cada sistema em detectar a frequência fundamental de um movimento periódico. Para este fim, o *rig* foi programado para executar um movimento senoidal com amplitude angular constante (definida em 20 graus), enquanto a frequência da GT foi sistematicamente variada.

A seleção do *range* de frequências para este ensaio foi diretamente ditada por uma limitação fundamental da metodologia de aquisição: a taxa de amostragem (f_s) dos protótipos. Conforme estabelecido no desenvolvimento do *firmware* (detalhado na Seção 3.2), os sistemas pUNO_v2 e pNANO_v2 foram configurados para realizar uma coleta a cada 0.1 segundo, resultando em uma taxa de amostragem fixa de $f_s = 10$ Hz.

De acordo com o Teorema de Nyquist-Shannon (SHANNON, 1949), um pilar da teoria de processamento de sinais, para evitar distorções e reconstruir fidedignamente um sinal analógico no domínio digital, a frequência de amostragem deve ser estritamente superior ao dobro da frequência máxima contida nesse sinal ($f_{\max}$). Esta relação é expressa pela equação 3.1.

$$f_s > 2 \cdot f_{\max} \quad (3.1)$$

Isso estabelece um limite teórico intransponível, a Frequência de Nyquist (f_N), para a máxima frequência que pode ser medida, dada pela equação 3.2.

$$f_N = \frac{f_s}{2} \quad (3.2)$$

No presente estudo, o limite de Nyquist é de $f_N = 10 \text{ Hz}/2 = 5 \text{ Hz}$.

Qualquer tentativa de medir um sinal com frequência igual ou superior a 5 Hz resultaria no fenômeno de *aliasing* (subamostragem). O *aliasing* faria com que a energia do sinal fosse "dobrada" de volta no espectro de frequência, sendo erroneamente representada por uma frequência "fantasma" (f_{alias}) de valor inferior. Por exemplo, um sinal de 6 Hz, ao ser amostrado a 10 Hz, seria falsamente medido como 4 Hz, conforme a fórmula $f_{\text{alias}} = |f_s - f_{\text{sinal}}| = |10 - 6| = 4 \text{ Hz}$.

Portanto, para garantir a integridade científica dos resultados e operar estritamente dentro do limite de Nyquist, os testes de varredura foram deliberadamente restritos a frequências inferiores a 5 Hz. Embora a faixa de tremor parkinsoniano possa se estender até 6 Hz, a faixa de 2 a 4.5 Hz ainda cobre uma porção clinicamente significativa do fenômeno.

O protocolo executado, com duração de 60 segundos para cada ensaio, consistiu na varredura das seguintes frequências de GT: 2.0 Hz, 3.0 Hz, 4.0 Hz e 4.5 Hz. A análise de precisão, detalhada na Seção 3.4.3, foi realizada comparando o pico de frequência detectado pela Transformada Rápida de Fourier (FFT) de cada protótipo com a GT estabelecida pelo controlador do servo.

3.3.3 Ensaio B: Análise de Fidelidade da Forma de Onda

Enquanto o Ensaio A se concentrou no domínio da frequência, o segundo ensaio experimental foi desenhado para avaliar a fidelidade dos protótipos no domínio do tempo. O

objetivo foi verificar o quão precisamente a forma de onda capturada pelos sensores (pUNO_v2 e pNANO_v2) correspondia à forma de onda senoidal da GT.

Para isolar a resposta à amplitude, a frequência da GT foi mantida constante em 4.0 Hz, um valor cuja precisão foi validada no ensaio anterior e que se encontra dentro da faixa de interesse. O protocolo consistiu na execução de três testes de 60 segundos cada, variando-se sistematicamente a amplitude angular programada no servo motor, com os seguintes valores: 10 graus, 20 graus e 30 graus.

A análise subsequente, detalhada na Seção 3.4.4, foca na comparação direta entre o sinal medido por cada protótipo e um sinal senoidal de referência (gerado digitalmente). Esta comparação utiliza a Normalização *Z-score*, para equiparar as escalas e focar puramente na forma, e o Erro Quadrático Médio (*Root Mean Squared Error* - RMSE) como métrica quantitativa da fidelidade do sinal.

3.3.4 Ensaio C: Análise de Estabilidade Estática (Ruído e Deriva)

O terceiro e último ensaio experimental, de caráter estático, foi concebido para avaliar duas métricas passivas cruciais: o ruído intrínseco de cada sensor e a estabilidade da estimativa de orientação ao longo do tempo, (o *drift*).

Diferentemente dos ensaios dinâmicos A e B, o atuador (servo motor) foi mantido desligado, e a plataforma de testes permaneceu em repouso absoluto sobre uma superfície nivelada. Nesta condição, o sinal de GT para todos os eixos angulares é teoricamente zero.

Os protótipos pUNO_v2 e pNANO_v2 foram acionados simultaneamente e realizaram a coleta de dados de forma contínua por um período de 30 minutos. Este teste de longa duração é fundamental para validar a eficácia dos algoritmos de fusão de sensores (Filtro de Kalman) implementados em cada firmware, permitindo uma comparação direta da acumulação de erro no eixo *Yaw*. Este eixo, por não ser corrigido pelo vetor gravitacional do acelerômetro, depende da estabilidade do giroscópio e, no caso do pNANO_v2, da referência absoluta do magnetômetro para sua estabilização.

Ao término dos três ensaios (A, B e C), os arquivos de dados brutos em formato CSV, gerados por ambos os protótipos, foram transferidos dos cartões MicroSD, organizados e catalogados de acordo com o protocolo executado. Este conjunto de dados constitui a base para a seção subsequente, 3.4 Análise Comparativa dos Sinais Inerciais, na qual os métodos de processamento de sinal e as métricas quantitativas são aplicados para determinar o desempenho comparativo de cada plataforma.

3.4 Análise Comparativa dos Sinais Inerciais

Completada a fase de coleta de dados (Seção 3.3), esta seção detalha os métodos de processamento e análise computacional empregados para extrair métricas quantitativas e avaliar o desempenho comparativo dos sistemas pUNO_v2 e pNANO_v2. Todo o pós-processamento dos sinais foi conduzido em ambiente *Python* (versão 3.12), com o suporte de bibliotecas especializadas para manipulação de dados e computação científica, notadamente *Pandas*, *NumPy*, *SciPy* e *Matplotlib*.

A metodologia de análise foi estruturada para responder diretamente aos objetivos de cada um dos três ensaios experimentais. As subseções seguintes descrevem, portanto: (1) a metodologia da FFT aplicada aos dados do Ensaio A para aferição da precisão em frequência; (2) o processo de cálculo do RMSE para os dados do Ensaio B, visando comparar a fidelidade da forma de onda no domínio do tempo; e (3) o método de quantificação do ruído e da deriva a partir dos dados do Ensaio C, utilizando o desvio padrão e a análise gráfica do eixo *Yaw*.

3.4.1 Análise no Domínio da Frequência (FFT)

Para avaliar os dados do Ensaio A, cujo objetivo era aferir a precisão na detecção da frequência, foi empregada a análise espectral. Os dados brutos coletados pelos sensores existem no domínio do tempo (i.e., amplitude angular em função do tempo), formato que não permite a extração direta da frequência fundamental com precisão. Tornou-se, portanto, necessária a conversão destes sinais para o domínio da frequência.

A ferramenta matemática fundamental para esta conversão é a Transformada de Fourier, nomeada em homenagem a Jean-Baptiste Joseph Fourier, que postulou que qualquer sinal periódico complexo pode ser decomposto em uma soma de ondas senoidais e cossenoidais simples (PROAKIS; MANOLAKIS, 1996). Para sinais discretos e finitos, como os gerados pelos protótipos, a ferramenta computacional apropriada é a Transformada Discreta de Fourier (DFT).

Neste trabalho, foi utilizada a Transformada Rápida de Fourier (FFT), um algoritmo otimizado para o cálculo da DFT, notavelmente popularizado por Cooley e Tukey (COOLEY; TUKEY, 1965), que reduz drasticamente a complexidade computacional. A FFT analisa a série temporal de entrada e retorna o espectro de frequência do sinal, quantificando a magnitude (ou energia) presente em cada "bin" de frequência.

A metodologia de aplicação da FFT consistiu nos seguintes passos:

1. **Seleção do Sinal de Análise:** Embora o movimento ocorresse primariamente em um eixo, optou-se por utilizar um sinal agregado que representasse a energia total do movimento, independente da orientação. Para isso, foi calculada a magnitude vetorial

(M) a partir dos três eixos de rotação (*Roll*, *Pitch* e *Yaw*) para cada amostra de tempo, conforme a Equação 3.3.

$$M = \sqrt{\text{Roll}^2 + \text{Pitch}^2 + \text{Yaw}^2} \quad (3.3)$$

2. **Remoção do Componente DC:** O sinal de magnitude resultante possui um valor médio (componente DC) que não representa o tremor (um componente AC). Este valor médio geraria um pico indesejado em 0 Hz no espectro. Portanto, o sinal foi normalizado subtraindo-se sua própria média, centrando-o em zero.
3. **Execução da FFT:** A função `fft` da biblioteca *SciPy* foi aplicada ao sinal de magnitude processado.
4. **Cálculo do Espectro de Magnitude:** O resultado da FFT é um vetor de números complexos. A magnitude em cada frequência foi obtida calculando-se o valor absoluto (`numpy.abs`) da saída da FFT e normalizando-a pelo número de amostras. Apenas a metade positiva do espectro (de 0 Hz até a Frequência de Nyquist de 5 Hz) foi utilizada para a análise.

A métrica de avaliação final foi o pico de frequência. A frequência correspondente à maior magnitude no espectro calculado foi identificada como a frequência medida (f_{medida}). Este valor foi então diretamente comparado ao GT ($f_{\text{ground_truth}}$) programado no controlador do servo, permitindo o cálculo do erro absoluto e relativo para cada protótipo em cada teste.

3.4.2 Análise no Domínio do Tempo (Fidelidade)

Para os dados do Ensaio B, o foco da análise foi transferido do domínio da frequência para o domínio do tempo. O objetivo foi quantificar a fidelidade da forma de onda, ou seja, o quão precisamente o sinal medido por cada protótipo reproduziu a forma senoidal da GT.

Diferentemente da análise de FFT, onde a magnitude agregada foi usada, esta metodologia exigiu o isolamento do eixo principal de movimento (no caso, o *Pitch*). Isso foi necessário pois o cálculo da magnitude (Equação 1) é uma operação não-linear que retifica o sinal (tornando-o sempre positivo), o que distorceria a forma de onda senoidal original e invalidaria a comparação de fidelidade.

A métrica quantitativa escolhida para esta comparação foi o RMSE. O RMSE é uma medida padrão da diferença entre valores preditos por um modelo (ou medidos por um sensor) e os valores reais observados (CHAI; DRAXLER, 2014). Ele é definido pela raiz quadrada da média dos quadrados das diferenças (os resíduos), conforme a equação 3.4.

$$RMSE = \sqrt{\frac{1}{N} \sum_{i=1}^N (\hat{y}_i - y_i)^2} \quad (3.4)$$

onde N é o número de amostras, $\hat{y}_i$ é o valor medido pelo sensor no instante i , e y_i é o valor da GT no mesmo instante. Um valor de RMSE menor indica um erro menor e, portanto, uma maior fidelidade do sinal.

Para garantir que a comparação de RMSE fosse justa e focada puramente na *forma* da onda, em vez de sua amplitude absoluta (que pode ser afetada por pequenas variações na calibração do sensor), foi aplicada uma etapa de pré-processamento crucial: a Padronização (ou Normalização Z-score). Ambos os sinais, o medido ($\hat{y}$) e a GT digital (y), foram submetidos a esta transformação, que ajusta os dados para terem uma média de zero e um desvio padrão de um. A análise de RMSE foi então realizada sobre estes sinais padronizados.

3.4.3 Análise de Estabilidade Estática (Ruído e Deriva)

Finalmente, para a análise dos dados do Ensaio C (teste estático de 30 minutos), a metodologia foi dividida em duas partes para avaliar as métricas de estabilidade passiva.

3.4.3.1 Quantificação do Ruído Intrínseco

O primeiro objetivo foi medir o nível de ruído de cada sistema em repouso. Idealmente, um sensor estático deveria registrar um valor constante, resultando em um sinal perfeitamente plano. Na prática, flutuações térmicas e imperfeições eletrônicas introduzem ruído no sinal. Uma medida estatística robusta para quantificar a magnitude dessa flutuação é o Desvio Padrão (σ).

Portanto, o desvio padrão foi calculado para os sinais de *Roll* e *Pitch* de cada protótipo ao longo dos 30 minutos de coleta. Um valor de σ menor indica menos flutuação, caracterizando um sinal mais "limpo" e com menor ruído intrínseco.

3.4.3.2 Quantificação da Deriva (Drift)

O segundo objetivo, e o mais crítico deste ensaio, foi avaliar a estabilidade da orientação a longo prazo, especificamente o *drift* no eixo de guinada *Yaw*. Conforme detalhado na Fundamentação Teórica (Seção 2.4.2), o *Yaw* não pode ser corrigido pelo acelerômetro (vetor gravitacional), dependendo puramente da estabilidade do giroscópio.

Esta análise testa diretamente a hipótese de que o pNANO_v2, ao empregar um magnetômetro em seu Filtro de Kalman, teria uma estabilidade de *Yaw* superior, enquanto o pUNO_v2 (sem magnetômetro) acumularia erro de integração do giroscópio, resultando em deriva.

A metodologia consistiu em uma análise gráfica do sinal de Yaw ao longo do tempo. Para uma comparação visual direta, ambos os sinais (pUNO e pNANO) foram normalizados para iniciar em 0 grau, subtraindo-se o valor da primeira amostra de toda a série. A métrica quantitativa, denominada Deriva Acumulada Total, foi definida como o valor final (em graus) do sinal de Yaw ao término dos 30 minutos, onde um valor mais próximo de zero indica maior estabilidade.

3.5 Análise de Viabilidade do Modelo LSTM

Essa etapa metodológica foi concentrada no desenvolvimento de um modelo de ML adaptado para o tipo de dado produzido pelos sensores. Ressalta-se, novamente, que o intuito da presente pesquisa é exclusivamente propor e verificar a viabilidade de um sistema embarcado no contexto da identificação dos tremores da DP, mas não contempla qualquer tipo de coleta com pacientes. Sendo assim, os métodos de ML a seguir foram minuciosamente construídos considerando futuras pesquisas e possíveis aplicações clínica dos protótipos desenvolvidos, através de uma densa revisão bibliográfica. Toda a sua modelagem foi pensada para estar de acordo com os dados provenientes dos Arduinos. Os códigos e algoritmos mencionados nesta seção podem ser encontrados no diretório "LSTM" no repositório: <https://github.com/ManuChristofalo/Arduino_Parkinson_Comparison>.

O desenvolvimento desse modelo não só é importante como fundamento ou métrica para futuros trabalhos e estudos, mas para a própria pesquisa em si, já que a LSTM desenvolvida foi também embarcada no Arduino Nano 33 BLE Sense Rev 2, através do recurso *TinyML*. A análise da viabilidade dessa possibilidade é crucial para a comparação entre os sistemas e, portanto, constituiu uma parte vital da metodologia.

Para o desenvolvimento da rede, considera-se hipotéticas coletas monitoradas e cronometradas com 1 hora de duração. Essa diretriz é de extrema importância, já que, para o treinamento do modelo, é explicitamente necessário saber SE e QUANDO houve um tremor dentro desse período de coleta.

3.5.1 Pré-Processamento dos Dados

A etapa de pré-processamento é um pilar fundamental no desenvolvimento de modelos de *deep learning*, especialmente ao lidar com dados brutos de sensores, que são inerentemente ruidosos e não estruturados para entrada direta em uma rede neural. O objetivo deste processo foi transformar dados coletados em um formato limpo, normalizado e sequencial, adequado para o treinamento de uma RNN do tipo LSTM.

Para a preparação dos dados, considera-se que os dados dos arquivos COLETA gerados pelo Arduino sejam agrupados em um só arquivo - no caso do código, um arquivo exemplo foi criado (*exemplo_artificial.csv* - disponível na pasta "data" do repositório citado anteriormente),

com dados puramente artificiais, cujo único objetivo foi exemplificar a estrutura e formatação necessária para o processamento (os valores contidos nesse arquivo não devem ser considerados para qualquer fim a não ser o de exemplificação). No caso, esse arquivo utilizado pelo modelo deve conter: uma coluna indicando o número da coleta (1, 2, 3...); três colunas para os dados dos três eixos inerciais; uma coluna marcando o respectivo tempo desses dados; uma coluna final marcando se, naquele momento, o movimento era um tremor ou não. Nota-se que essa coluna final só é necessária para as coletas de treinamento: com um modelo já treinado, ela pode ser suprimida, indicando uma possível coleta não monitorada.

O *pipeline* de pré-processamento foi dividido em quatro etapas principais: divisão estratégica dos dados, engenharia de *features*, normalização e geração de sequências (janelamento).

A primeira e mais crítica etapa do pré-processamento foi a segmentação do *dataset* completo em conjuntos de treino e teste. Para garantir a validade e a generalização do modelo, esta divisão não foi realizada de forma aleatória (linha a linha), mas sim com base no identificador da coleta (ID_Coleta).

Oito coletas (IDs 1-8) foram designadas para o conjunto de treinamento e duas coletas (IDs 9-10) foram reservadas para o conjunto de teste. Esses valores podem ser adaptados para mais ou menos coletas através das variáveis `TRAIN_COLETAS` e `TEST_COLETAS`, localizadas no arquivo *main.py*. Esta abordagem é crucial para prevenir o vazamento de dados (*data leakage*), assegurando que o modelo seja avaliado em dados genuinamente inéditos, provenientes de sessões de coleta que ele nunca encontrou durante o treinamento. Todas as etapas subsequentes de pré-processamento, como a normalização, foram calibradas usando apenas os dados de treino.

Os dados das unidades inerciais pós Filtro de Kalman forneceram três eixos de rotação: *Roll* (x), *Pitch* (y) e *Yaw* (z). Embora esses três canais pudessem ser usados diretamente como *features* multivariadas, eles representam componentes direcionais do movimento. Com o objetivo de reduzir a dimensionalidade e, simultaneamente, criar uma *feature* mais robusta que represente a intensidade total do movimento angular, independente da orientação específica do sensor, uma nova *feature* denominada 'Magnitude' foi calculada.

A magnitude vetorial foi computada como a raiz quadrada da soma dos quadrados dos três componentes, conforme a equação 3.5.

$$Magnitude = \sqrt{Roll(x)^2 + Pitch(y)^2 + Yaw(z)^2} \quad (3.5)$$

Esta nova *feature*, juntamente com os três eixos originais, compôs o conjunto de *features* de entrada (*Roll*, *Pitch*, *Yaw*, *Magnitude*), fornecendo ao modelo tanto a informação direcional quanto a de intensidade agregada.

Contudo, as redes neurais, incluindo LSTMs, são sensíveis à escala das *features* de

entrada. Diferenças significativas nas faixas de valores entre as *features* podem prejudicar a convergência do modelo durante o treinamento. Para mitigar este problema, optou-se pela normalização *Z-score* (Padronização), implementada através do *StandardScaler* da biblioteca *Scikit-learn*.

Este método transforma os dados para que tenham uma média de zero ($\mu = 0$) e um desvio padrão de um ($\sigma = 1$). A importância desta etapa reside em garantir que todas as *features* contribuam de forma equitativa para o aprendizado do modelo, acelerando a convergência e melhorando a estabilidade do gradiente. Outros métodos, como o *MinMax*, por exemplo, sofre alta sensibilidade a *outliers* (ruído) e, no caso, impactariam negativamente o modelo (JAFARI, 2022).

É crucial notar que o *StandardScaler* foi "treinado" (método *fit*) exclusivamente com os dados do conjunto de treinamento (Coletas 1-8, no exemplo). Os parâmetros de média e desvio padrão calculados foram, então, aplicados (método *transform*) tanto ao conjunto de treino quanto ao conjunto de teste. Esta prática evita que qualquer informação estatística do conjunto de teste "vaze" para o processo de treinamento, mantendo a integridade da avaliação do modelo.

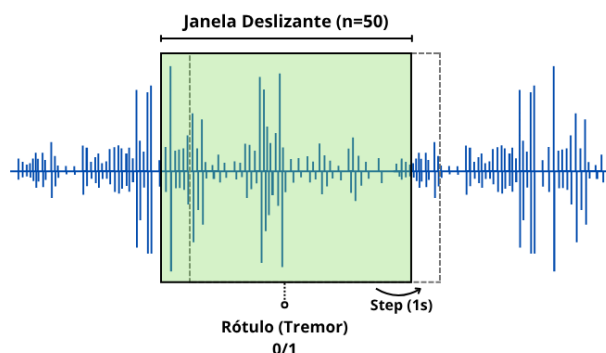
Como o modelo LSTM é projetado para aprender padrões a partir de sequências de dados, não de instâncias isoladas, foi necessário transformar a série temporal contínua em segmentos supervisionados, ou "janelas" deslizantes (*sliding windows*).

Para esta tarefa, foram definidos dois hiperparâmetros: o tamanho da janela (*window size*) e o passo (*step*). O *window size* define quantos passos de tempo (amostras) a LSTM analisará de uma só vez (ex: 50 amostras, equivalentes a 5 segundos de dados). O *step* define de quantos em quantos passos de tempo uma nova janela será criada (ex: 10 amostras, ou 1 segundo).

Este processo desliza sobre os dados de cada coleta. Para cada janela de N amostras (o *window size*), o rótulo de saída (*target*) correspondente foi determinado. Como o objetivo é classificar a janela inteira como "Tremor"(1) ou "Não-Tremor"(0), utilizou-se a moda estatística da coluna Tremor dentro daquela janela. Ou seja, se a maioria dos pontos de dados dentro da janela fosse marcada como '1', a janela inteira receberia o rótulo '1', e vice-versa. Esse processo está devidamente ilustrado na através da figura 10.

Ao final desta etapa, o conjunto de dados foi transformado de uma série temporal 2D ([linhas, features]) para um tensor 3D ([número_de_janelas, tamanho_da_janela, número_de_features]), que é o formato de entrada esperado pela camada LSTM no *Keras*/*TensorFlow*. O *target* (y) foi transformado em um vetor 1D ([número_de_janelas]) contendo os rótulos de classificação para cada sequência.

Figura 10 – Ilustração do funcionamento da janela deslizante



Fonte: elaborada pelo autor.

3.5.2 Arquitetura de Referência (LSTM)

Para endereçar o objetivo central dos sistemas - identificar a presença de tremor parkinsoniano em segmentos de dados de movimento - foi proposta uma abordagem de classificação de séries temporais utilizando *deep learning*.

O objetivo metodológico difere fundamentalmente de uma análise de regressão (onde se busca prever o próximo valor de um sinal), já que, no caso, visa a classificação: o modelo deve analisar um segmento de tempo (uma janela) e atribuir-lhe um rótulo binário (0 para Não-Tremor, 1 para Tremor). Esta abordagem permite ao modelo aprender os padrões temporais complexos e as assinaturas de frequência que distinguem um tremor patológico de um movimento voluntário ou repouso, mesmo que ambos apresentem variabilidade nos sinais de *Roll*, *Pitch* e *Yaw*.

A implementação foi realizada em *Python*, utilizando a biblioteca *TensorFlow* com a *API Keras*, e organizada de forma modular para garantir reprodutibilidade e manutenibilidade.

A escolha da LSTM se justifica por sua capacidade intrínseca de aprender dependências de longo prazo em dados sequenciais, superando o problema do desaparecimento do gradiente (*vanishing gradient*) RNNs simples (HOCHREITER; SCHMIDHUBER, 1997). A arquitetura foi projetada para receber o tensor 3D ([janelas, window_size, features]) gerado na etapa de pré-processamento e produzir uma saída binária (0 ou 1).

A arquitetura do modelo é composta pelas seguintes camadas:

1. **Camada de Entrada (*Input Layer*):** Define o formato de entrada esperado, que corresponde a uma janela de tamanho WINDOW_SIZE (ex: 50 amostras) e N_FEATURES (ex: 4 features).
2. **Camada LSTM (*LSTM Layer*):** A primeira camada LSTM (com 64 unidades) processa a sequência de entrada. O parâmetro `return_sequences=True` é utilizado para que a camada retorne a sequência completa de estados ocultos, permitindo que a próxima camada recorrente também receba uma sequência.

3. **Camada de Regularização (*Dropout Layer*):** Uma camada Dropout (com taxa de 0.2) é inserida para reduzir o sobreajuste (*overfitting*), desativando aleatoriamente uma fração dos neurônios durante o treinamento.
4. **Segunda Camada LSTM (*LSTM Layer*):** Uma segunda camada LSTM (com 32 unidades) recebe a sequência da camada anterior. Com o parâmetro `return_sequences=False`, ela retorna apenas o último estado oculto, que serve como um vetor de características (*feature vector*) resumindo toda a sequência de entrada.
5. **Camada Densa (*Dense Layer*):** Uma camada totalmente conectada (com 16 neurônios e ativação ReLU) atua como um classificador, aprendendo combinações não-lineares do vetor de características gerado pela LSTM.
6. **Camada de Saída (*Output Layer*):** A camada final é uma única unidade Densa com ativação sigmoide. Esta função comprime a saída em um valor entre 0 e 1, que é interpretado como a probabilidade da sequência de entrada pertencer à classe Tremor (1).

Para a compilação do modelo, utilizou-se o otimizador Adam (*Adaptive Moment Estimation*) e a função de perda `binary_crossentropy`. A escolha desta função de perda é padrão e matematicamente adequada para problemas de classificação binária.

Visando as boas práticas de engenharia de *software* e a clareza metodológica, o *pipeline* de processamento e modelagem foi implementado de forma modular. Cada arquivo *Python* encapsula uma responsabilidade específica dentro do fluxo de trabalho:

- `data_loader.py`: Responsável por carregar o arquivo CSV, realizar a engenharia de *features* (cálculo da Magnitude) e executar a divisão estratégica dos dados em treino e teste com base no ID_Coleta.
- `preprocessing.py`: Contém as funções críticas de pré-processamento. Inclui a lógica para treinar o `StandardScaler` (normalização) apenas nos dados de treino e a função `create_sequences` para realizar o janelamento deslizante e a atribuição de rótulos (moda estatística).
- `model.py`: Define a arquitetura da rede neural. As funções `build_model` e `compile_model` constroem e compilam a estrutura LSTM descrita na seção anterior, permitindo fácil reconfiguração e experimentação.
- `plotting.py`: Módulo dedicado à visualização dos resultados. Contém funções para gerar os gráficos de dados normalizados, histórico de treinamento (perda e acurácia) e a comparação entre as previsões do modelo e os rótulos reais.

- `main.py`: O orquestrador do projeto. Este *script* principal importa os demais módulos e executa o *pipeline* completo na ordem correta: carregar dados, pré-processar, construir o modelo, treinar e, por fim, avaliar e plotar os resultados das coletas de teste.

3.5.3 Metodologia de Treinamento e Avaliação

O processo de treinamento do modelo (`model.fit`) foi configurado não apenas para ajustar os pesos da rede, mas também para lidar com desafios comuns em dados biomédicos, como o desbalanceamento de classes e o sobreajuste.

Monitoramento e Combate ao Sobreajuste: Durante o treinamento, 20% dos dados de treino exemplo (Coletas 1-8) foram separados internamente como um conjunto de validação (`validation_split=0.2`). A cada época, o modelo é avaliado neste subconjunto. Isso permite monitorar a perda (*loss*) e a acurácia de validação em paralelo com as métricas de treino. O desempenho neste conjunto de validação é o indicador principal para diagnosticar o sobreajuste (*overfitting*) – o ponto em que o modelo começa a memorizar os dados de treino em vez de generalizar os padrões.

Tratamento de Classes Desbalanceadas: É esperado que os períodos de Não-Tremor (classe 0) sejam significativamente mais frequentes que os de Tremor (classe 1). Se não tratado, o modelo pode desenvolver um viés para a classe majoritária. Para mitigar isso, foi utilizado o parâmetro `class_weight='balanced'`. Esta técnica ajusta automaticamente a função de perda para penalizar mais severamente os erros de classificação na classe minoritária (Tremor), forçando o modelo a dedicar igual importância à detecção correta de ambas as classes.

Metodologia de Avaliação Final: A performance final do modelo será quantificada exclusivamente no conjunto de teste (Coletas 9 e 10, no exemplo), que não participou de nenhuma fase do treinamento ou ajuste de parâmetros. Esta avaliação utilizará um conjunto de métricas padrão para classificação:

1. **Acurácia:** A proporção de classificações corretas (Tremor e Não-Tremor).
2. **Matriz de Confusão:** Uma tabela que detalha os Verdadeiros Positivos, Falsos Positivos, Verdadeiros Negativos e Falsos Negativos.
3. **Precisão (*Precision*):** Das janelas classificadas como Tremor, quantas eram *realmente* Tremor.
4. **Revocação (*Recall*):** Das janelas que eram *realmente* Tremor, quantas o modelo conseguiu identificar.
5. **F1-Score:** A média harmônica entre Precisão e Revocação.

A análise conjunta destas métricas, especialmente Precisão e Revocação, fornece uma visão robusta da capacidade do modelo em identificar corretamente os episódios de tremor, que é o objetivo primário da classificação.

3.5.4 Aplicação do *TinyML*

A arquitetura de referência LSTM, conforme descrita na Seção 3.5.2, representa um modelo computacional treinado em um ambiente de alta capacidade (*desktop/Python*). No entanto, um dos objetivos centrais desta pesquisa é avaliar a viabilidade de executar esta inteligência diretamente no dispositivo embarcado, um paradigma conhecido como *Edge Computing* ou, mais especificamente, *Tiny Machine Learning (TinyML)*. Esta abordagem oferece vantagens críticas para um dispositivo vestível de monitoramento de saúde, como a redução drástica do consumo energético (ao evitar a transmissão de dados via *Bluetooth*), a garantia de privacidade (os dados do paciente não saem do dispositivo) e a capacidade de inferência em tempo real.

O pNANO_v2, equipado com o processador ARM Cortex-M4 e 256KB de SRAM, foi especificamente projetado para esta finalidade (Arduino, 2025a). Esta seção detalha, portanto, a metodologia empregada para converter o modelo LSTM treinado e avaliar sua viabilidade de implementação no pNANO_v2.

3.5.4.1 *Pipeline* de Conversão para o Ambiente Embarcado

O processo de migração do modelo Keras (.h5) para um formato executável em microcontrolador seguiu um *pipeline* de conversão padrão do *TensorFlow Lite* (ABADI et al., 2015).

1. **Conversão do Modelo (Float32):** O primeiro passo consistiu em utilizar o TFLite-Converter do *TensorFlow* para converter o modelo Keras treinado em um arquivo de modelo *TensorFlow Lite* (.tflite). Este modelo inicial ainda opera com aritmética de ponto flutuante de 32 bits (Float32).
2. **Otimização por Quantização (Int8):** Modelos Float32 são computacionalmente custosos e lentos em microcontroladores. Para otimização, foi aplicada a técnica de quantização pós-treinamento (*post-training quantization*). Este processo analisa os pesos e ativações do modelo e os converte de valores Float32 para inteiros de 8 bits (Int8). A quantização resulta em uma redução de aproximadamente 4 vezes no tamanho do modelo e um aumento expressivo na velocidade de inferência em *hardware* compatível (como o ARM Cortex-M4), com impacto mínimo ou nulo na acurácia do modelo para muitas tarefas (ABADI et al., 2015).

3. **Geração do Array em C:** Microcontroladores não leem arquivos de sistema. Portanto, o modelo .tflite otimizado (Int8) foi convertido em um arquivo de *header* C (.h) usando a ferramenta de linha de comando *xxd*. Este processo transforma o modelo binário em um grande `unsigned char` array, que é compilado diretamente no *firmware* do Arduino e armazenado em sua memória de programa (*Flash*).
4. **Desenvolvimento do *Firmware* de Inferência:** Um *firmware* específico foi desenvolvido para o pNANO_v2 utilizando a biblioteca `Arduino_TensorFlowLite`. Este código é responsável por:
 - Incluir o *header* do modelo (.h).
 - Alocar um bloco de memória SRAM contíguo, conhecido como Arena de Tensores (*Tensor Arena*), para as operações do modelo.
 - Instanciar o interpretador TFLite.
 - No *loop* principal, realizar o mesmo pré-processamento dos dados em tempo real (coletar 50 amostras, calcular a magnitude e aplicar a normalização *Z-score* com os parâmetros de média e desvio padrão obtidos no treinamento).
 - Preencher o tensor de entrada do modelo com a janela de dados processada.
 - Executar a inferência chamando "interpreter->Invoke()".
 - Ler o tensor de saída, que contém a probabilidade (0.0 a 1.0) da janela ser classificada como "Tremor".

4 Resultados Experimentais

4.1 Comparativo de *Hardware* e Custo

A primeira análise comparativa foca nos atributos físicos e financeiros de cada protótipo, fatores determinantes para a viabilidade e usabilidade de um dispositivo vestível. O quadro 1 detalha os componentes de *hardware* utilizados em cada sistema (excluindo-se conectores e *jumpers*), seus custos individuais no varejo nacional (sem frete) e o custo total de montagem de cada protótipo.

Quadro 1 – Detalhamento de componentes e custos dos protótipos

Protótipo pUNO_v2	
Componente	Preço (R\$)
Arduino UNO R3	37,00
MPU6050 (Giroscópio/Acelerômetro)	35,90
Módulo Micro SD 5V	18,90
Suporte 2x 18650 (em série)	15,90
Baterias 18650 9800 mAh (2x)	29,80
Cartão Micro SD 8GB	29,70
Placa Perfurada 8x12cm	8,30
CUSTO TOTAL (pUNO_v2)	175,50
Protótipo pNANO_v2	
Componente	Preço (R\$)
Arduino Nano 33 BLE Sense Rev2	415,00
Módulo Micro SD Mini 3.3V	19,00
Suporte 18650 (uma única)	14,90
Bateria 18650 9800 mAh (1x)	14,90
Cartão Micro SD 8GB	29,70
Placa Perfurada 5x7cm	4,50
CUSTO TOTAL (pNANO_v2)	498,00

Fonte: Elaborado pelo autor.

Além dos custos, os atributos físicos finais dos circuitos montados e soldados, incluindo as baterias, foram mensurados:

- **pUNO_v2:** Dimensões de 8cm x 12cm x 2cm e peso total de 123,9g.
- **pNANO_v2:** Dimensões de 5cm x 7cm x 2cm e peso total de 54,7g.

A análise dos dados revela disparidades significativas. O custo do *pNANO_v2* é aproximadamente 2,8 vezes superior ao do *pUNO_v2*. Esta diferença é quase inteiramente atribuível ao microcontrolador: o Arduino Nano 33 BLE Sense Rev 2 (R\$ 415,00) é um

componente de alta tecnologia, cujo valor é justificado pelo seu processador ARM Cortex-M4 de 32 bits e, crucialmente, pelo conjunto de sensores de alta precisão já integrados à placa, como a IMU de 9 eixos. Em contrapartida, o *pUNO_v2* baseia-se na plataforma canônica e de baixo custo do Arduino UNO, cujos sensores (MPU6050) e periféricos são módulos externos que, embora funcionais, são adquiridos separadamente a um custo reduzido.

Essa distinção arquitetônica - sensores integrados versus externos - impacta diretamente os atributos físicos. O *pNANO_v2* é substancialmente mais compacto e mais leve, pesando menos da metade. Isso se deve não apenas ao *form factor* reduzido da placa Nano, mas também à eliminação da necessidade de módulos adicionais e ao uso de apenas uma bateria 18650, conforme estabelecido na metodologia de alimentação (Seção 3.1).

Em termos de viabilidade e aplicação, cada protótipo atende a um cenário distinto. O *pUNO_v2*, com seu custo notavelmente baixo e uso de componentes amplamente democratizados, consolida-se como uma plataforma ideal para a prototipagem inicial, validação de conceitos (proof-of-concept), e aplicações em contextos com severas restrições orçamentárias ou, simplesmente, como ferramenta pedagógica. Embora suas dimensões e peso o tornem menos confortável para um monitoramento contínuo, ele é perfeitamente viável para coletas de dados controladas e de curta duração, além de possuir grande valor para fins educacionais e de introdução à pesquisa em engenharia biomédica.

Por outro lado, o *pNANO_v2*, apesar do custo elevado, representa uma solução tecnologicamente superior. Suas dimensões compactas e peso reduzido são características essenciais para um dispositivo vestível, priorizando o conforto e a usabilidade do paciente. O *hardware* avançado não só permite a captura de dados mais complexos (como o uso do magnetômetro), mas é o único dos dois sistemas com capacidade computacional para futuras implementações de inteligência artificial embarcada (*TinyML*), um dos objetivos centrais da pesquisa. Portanto, o *pNANO_v2* é a plataforma indicada para o desenvolvimento de um protótipo clinicamente robusto, onde a precisão, o conforto e a capacidade de processamento *on-device* são mais críticos que o custo de fabricação.

4.2 Comparativo de Consumo Energético

A análise teórica do orçamento de potência, detalhada na Seção 3.2.1, projetou uma autonomia de bateria superior a 98 horas, com base na capacidade nominal de 9800 mAh das baterias 18650 selecionadas. Para validar esta projeção e avaliar o consumo energético real de cada protótipo, foram conduzidos três ensaios empíricos de duração. Nestes testes, os sistemas *pUNO_v2* e *pNANO_v2*, executando o *firmware* final de coleta de dados, operaram de forma autônoma até a exaustão completa da fonte de alimentação.

Os resultados, apresentados no quadro 2, revelam o tempo de operação contínua de cada protótipo em cada ensaio.

Quadro 2 – Resultado dos testes de duração de bateria

Nº Teste	Duração Nano	Duração UNO
1	4 horas e 31 minutos	7 horas e 3 minutos
2	4 horas e 23 minutos	6 horas e 45 minutos
3	5 horas e 36 minutos	5 horas e 57 minutos

Fonte: Elaborado pelo autor.

A primeira observação é a notável discrepância entre a autonomia teórica de 98 horas e a autonomia empírica, que se situa consistentemente na faixa de 4 a 7 horas. Esta diferença, em que a autonomia real representa menos de 10% do valor teórico, corrobora a hipótese levantada durante a validação dos componentes (Seção 3.1): a capacidade efetiva da maioria das baterias comerciais 18650 utilizadas é significativamente inferior à capacidade nominal indicada em seus rótulos. Esse comportamento é condizente com os testes preliminares, nos quais uma bateria com capacidade nominal de 3500mAh mostrou-se insuficiente para alimentar o circuito integrado, sugerindo que a capacidade real das unidades de 9800mAh, embora funcional, é drasticamente menor que a anunciada.

A segunda observação relevante é que o protótipo *pUNO_v2* apresentou, em todos os testes, uma autonomia superior à do *pNANO_v2*. Este resultado, à primeira vista, contradiz a fundamentação teórica estabelecida na Seção 2.2. A plataforma Arduino UNO é reconhecida por sua arquitetura energeticamente ineficiente, operando com lógica de 5V e utilizando reguladores de tensão lineares que dissipam o excesso de tensão como calor, um processo inerentemente ineficiente. Em contrapartida, o Arduino Nano 33 BLE Sense Rev 2 foi especificamente projetado para baixo consumo, utilizando uma arquitetura moderna ARM Cortex-M4, lógica de 3.3V e um SoC otimizado para eficiência.

A explicação para esta aparente contradição não reside na eficiência de consumo dos microcontroladores, mas sim na arquitetura de alimentação de cada protótipo, conforme definido na metodologia (Seção 3.1). O *pUNO_v2*, para satisfazer sua exigência de tensão de entrada (7-12V), foi alimentado por *duas* baterias 18650 em série. O *pNANO_v2*, por sua vez, operou com apenas *uma* bateria 18650.

Portanto, embora o *pUNO_v2* tenha registrado maior duração, seu sistema dispunha do dobro do reservatório de energia total (em Watt-hora) em comparação com o *pNANO_v2*. Os dados, na verdade, reforçam a tese de ineficiência do UNO: mesmo com o dobro da energia disponível, o protótipo não alcançou o dobro da autonomia (especialmente no Teste 3, com 5h 57m contra 5h 36m do Nano). Isso indica que o sistema *pUNO_v2* drena sua fonte de energia (maior) a uma taxa consideravelmente mais elevada, confirmando sua menor eficiência energética, conforme previsto na fundamentação teórica. Conclui-se que o *pNANO_v2*, apesar de apresentar menor autonomia nesta configuração específica, é comprovadamente a plataforma mais eficiente, alcançando um tempo de operação robusto com metade do reservatório de

energia.

4.3 Comparativo de Captura

A avaliação da qualidade de captura de cada protótipo foi realizada através dos três ensaios mecânicos controlados, detalhados na metodologia (Seção 3.3). Os resultados destes testes permitem uma comparação quantitativa direta da precisão na detecção de frequência (Ensaio A), da fidelidade na reprodução da forma de onda (Ensaio B) e da estabilidade estática do sinal (Ensaio C).

O Ensaio A (Análise de Precisão em Frequência) submeteu os protótipos a oscilações senoidais em frequências controladas. Os dados, analisados via FFT, estão sumarizados na tabela 1.

Tabela 1 – Resultados do Ensaio A: Análise de Precisão em Frequência

Ground Truth (Hz)	<i>pUNO_v2</i>		<i>pNANO_v2</i>	
	Freq. Medida (Hz)	Erro (%)	Freq. Medida (Hz)	Erro (%)
2.0	2.01	0.56	1.99	-0.32
3.0	2.18	-27.40	2.99	-0.32
4.0	0.02	-99.58	3.99	-0.32
4.5	3.67	-18.54	0.22	-95.20

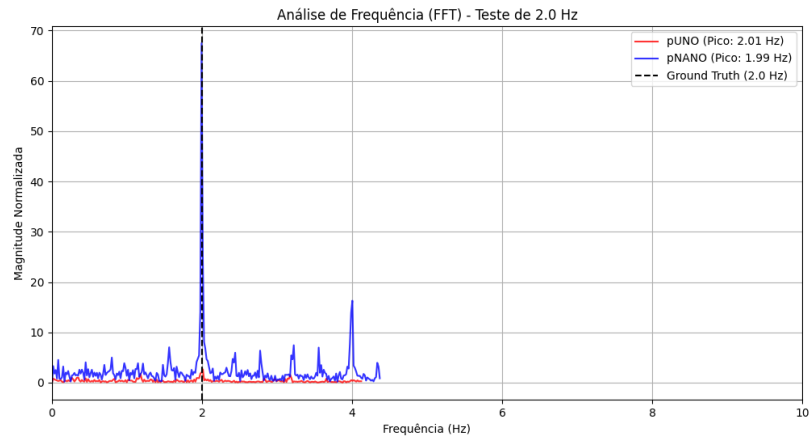
Fonte: Elaborada pelo autor.

Os resultados do Ensaio A demonstram uma clara superioridade do *pNANO_v2* na detecção de frequências clinicamente relevantes. Na faixa de 2 Hz a 4 Hz, que engloba a maior parte dos tremores parkinsonianos, o *pNANO_v2* manteve uma precisão notável, com um erro percentual máximo de -0.32%. Em contrapartida, o *pUNO_v2* mostrou-se confiável apenas em 2 Hz (figura 11). A partir de 3 Hz, seu desempenho degradou-se severamente, culminando em uma falha total em 4 Hz (Erro de -99.58%), onde foi incapaz de detectar o movimento. Este comportamento sugere que o sensor MPU6050 ou a capacidade de processamento da plataforma 8-bit do *pUNO_v2* são insuficientes para capturar de forma confiável movimentos nesta faixa de frequência.

É crucial notar que ambos os sistemas falharam no teste de 4.5 Hz. Esta falha mútua não deve ser atribuída aos sensores, mas sim a uma limitação da metodologia de aquisição: a taxa de amostragem de 10 Hz (definida no *firmware*). Estando a 4.5 Hz muito próxima da frequência de Nyquist (5.0 Hz), ocorre uma atenuação severa do sinal, tornando a detecção pela FFT impraticável.

O Ensaio B (Análise de Fidelidade da Forma de Onda) avaliou a capacidade dos sensores em reproduzir a forma senoidal do movimento, utilizando o Erro Quadrático Médio RMSE como métrica. Os resultados estão dispostos na tabela 2.

Figura 11 – Resultados do Ensaio A para 2Hz



Fonte: Elaborada pelo autor.

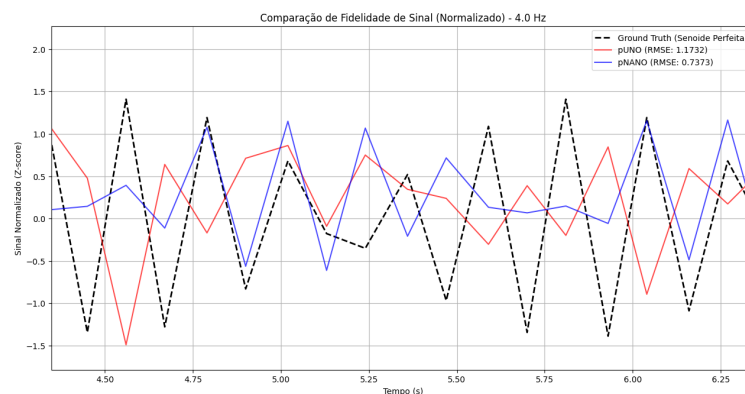
Tabela 2 – Resultados do Ensaio B: Análise de Fidelidade da Forma de Onda

Amplitude (Ground Truth)	RMSE <i>pUNO_v2</i>	RMSE <i>pNANO_v2</i>
10 graus	1.6520	2.1173
20 graus	1.3291	1.5577
30 graus	1.1732	0.7373

Fonte: Elaborada pelo autor.

Neste ensaio, os resultados foram mistos. Em amplitudes baixas e médias (10 e 20 graus), o *pUNO_v2* apresentou um RMSE menor, sugerindo uma fidelidade ligeiramente superior à forma de onda de movimentos menos intensos. Contudo, em alta amplitude (30 graus), o *pNANO_v2* demonstrou uma capacidade significativamente maior de rastrear o movimento, com um erro consideravelmente menor que o do *pUNO_v2*. Isso indica que o sensor BMI270 do *pNANO_v2* responde com maior precisão a dinâmicas de movimento mais enérgicas. A adequação dos sensores às ondas simuladas pode ser constatada na figura 12.

Figura 12 – Resultados do Ensaio B para 30° de Amplitude



Fonte: Elaborada pelo autor.

Finalmente, o Ensaio C (Análise de Estabilidade Estática) avaliou o ruído intrínseco e o *drift* dos sensores durante um período de 30 minutos em repouso absoluto. O ruído foi quantificado pelo desvio padrão (tabela 3) e a deriva pela variação total acumulada no eixo Yaw (tabela 4).

Tabela 3 – Resultados do Ensaio C: Quantificação do Ruído Intrínseco (Desvio Padrão)

Protótipo	Roll (σ)	Pitch (σ)	Yaw (σ)
<i>pUNO_v2</i>	0.1638	0.2714	0.1037
<i>pNANO_v2</i>	0.0960	0.0642	1.7586

Fonte: Elaborada pelo autor.

Tabela 4 – Resultados do Ensaio C: Quantificação da Deriva (*Drift*)

Protótipo	Deriva após 30 minutos
<i>pUNO_v2</i>	1.56
<i>pNANO_v2</i>	-0.18

Fonte: Elaborada pelo autor.

Na análise de ruído, o *pNANO_v2* apresentou um sinal "mais limpo" nos eixos de Roll e Pitch, que são estabilizados pelo acelerômetro. Seus valores de desvio padrão foram inferiores aos do *pUNO_v2*, indicando um sensor de maior qualidade. Paradoxalmente, o ruído no eixo Yaw foi drasticamente maior no *pNANO_v2* do que no *pUNO_v2*.

Este paradoxo é resolvido pela análise de deriva (*drift*). Ao término dos 30 minutos, o *pUNO_v2*, sem correção de referência absoluta, acumulou um erro de 1.56 graus no eixo Yaw. Em contraste, o *pNANO_v2* apresentou uma estabilidade quase perfeita, com uma variação de apenas -0.18 graus.

A justificativa para este comportamento reside na metodologia de fusão de sensores. O *pUNO_v2* depende exclusivamente do giroscópio para o Yaw, resultando em baixo ruído estático, mas em uma deriva inevitável a longo prazo devido à integração de erros. O *pNANO_v2*, por sua vez, utiliza o magnetômetro em seu Filtro de Kalman para corrigir ativamente a deriva do giroscópio, garantindo estabilidade de orientação a longo prazo. No entanto, o próprio magnetômetro é suscetível a flutuações do campo magnético ambiente, e é essa correção ativa que introduz o "ruído" (flutuação de curto prazo) observado no Yaw (1.7586 σ). Este resultado valida que o *pNANO_v2* sacrifica a limpeza estática do sinal de Yaw em troca de uma estabilidade de orientação a longo prazo, sendo esta última uma característica muito mais crucial para o monitoramento contínuo.

Vale ressaltar, também, que nessa coleta de longo prazo, o *pUNO_v2* apresentou instabilidade em alguns momentos de escrita: em quatro momentos, durante 3s aproximadamente, os dados foram sobrepostos, alterando até mesmo a formatação do arquivo CSV. Isso provavelmente é um reflexo da capacidade de processamento do Arduino Nano, que permite a

recepção e gravação de dados de forma segura e contínua. Ainda assim, devido ao anexo 'do magnetômetro e à quantidade maior de operações realizadas pelo *firmware* do *pNANO_v2*, e pelo método de gravação estar atrelado a um *delay* fixo, o tempo entre uma coleta e outra é levemente maior quando comparada ao *pUNO_v2*. Isso implica que, no caso das coletas serem delimitadas pelo *timestamp* (e.g. 60s), o arquivo gerado pelo *pUNO_v2* contém mais amostras que o gerado pelo Nano.

4.4 Análise de Viabilidade da Implementação LSTM

A análise de viabilidade do uso de RNNs do tipo LSTM é crucial, pois representa o método de análise de dados mais avançado e proposto por este trabalho. A LSTM é uma arquitetura de *deep learning* projetada especificamente para modelar e aprender dependências temporais em dados sequenciais, como as séries temporais geradas por sensores inerciais. Sua capacidade de reter informações de longo prazo, superando o problema do gradiente que desaparece (comum em RNNs simples), a torna ideal para a tarefa de classificar padrões complexos, como a distinção entre um tremor patológico e um movimento voluntário.

A viabilidade de sua aplicação pode ser dividida em dois cenários: análise *offline* (pós-coleta) e análise embarcada (*on-device*).

No cenário de análise *offline*, o protótipo *wearable* atua puramente como um dispositivo de aquisição e armazenamento (um *data logger*). Os dados são coletados e gravados em um cartão MicroSD. Posteriormente, esses arquivos são transferidos para um computador, onde um modelo LSTM, treinado em um ambiente de alta capacidade (como *Python* com *TensorFlow/Keras*, conforme a metodologia), realiza a classificação.

Neste contexto, ambos os protótipos, *pUNO_v2* e *pNANO_v2*, são tecnicamente viáveis, pois ambos cumprem a função de gerar os arquivos de dados. Contudo, a *eficácia* dessa viabilidade é drasticamente diferente. Como demonstrado na Seção 4.3 (Comparativo de Captura), a qualidade dos dados do *pUNO_v2* é comprometida: sua incapacidade de detectar com precisão frequências nas faixas de 3 Hz e 4 Hz (Tabela 1) significa que os dados de entrada para a LSTM seriam de baixa fidelidade, potencialmente corrompendo o processo de aprendizado e classificação. O *pNANO_v2*, ao contrário, forneceu dados de alta precisão e fidelidade nessas faixas críticas. Portanto, para a análise *offline*, embora ambos sejam funcionalmente viáveis, apenas o *pNANO_v2* fornece a qualidade de dados necessária para que um modelo LSTM produza resultados clinicamente relevantes e confiáveis.

O segundo cenário, a análise embarcada, refere-se à capacidade de executar o modelo LSTM diretamente no microcontrolador, através do *TinyML*. Esta abordagem é detalhada a seguir.

4.4.1 Validação da Viabilidade do *TinyML*

A implementação do *TinyML* permite que o *wearable* realize a classificação dos tremores em tempo real, diretamente no dispositivo. Isso oferece vantagens substanciais em termos de privacidade (dados brutos não saem do dispositivo), consumo de energia (a transmissão de dados é minimizada) e resposta imediata.

A viabilidade desta abordagem difere radicalmente entre os protótipos:

O *pUNO_v2* é completamente inviável para a implementação do *TinyML*. Seu micro-controlador, o Atmel ATmega328P, é uma arquitetura 8-bit com especificações de memória que são insuficientes para as demandas do *framework TensorFlow Lite*. A biblioteca de inferência, o modelo LSTM (mesmo que quantizado) e a "Arena de Tensores" (o bloco de RAM necessário para as operações do modelo) excedem em ordens de magnitude a capacidade de 2KB de SRAM disponível.

O *pNANO_v2*, por outro lado, foi especificamente escolhido para este fim. Ele é equipado com um processador ARM Cortex-M4 e, mais importante, 256KB de SRAM e 1MB de memória *Flash*. Esta arquitetura é robusta e totalmente compatível com a biblioteca *Arduino_TensorFlowLite*. A metodologia de conversão, detalhada na Seção 3.5.4.1, confirma o *pipeline* de implementação: o modelo Keras treinado é convertido para o formato *TensorFlow Lite*, otimizado através de quantização (Float32 para Int8) para reduzir seu tamanho em cerca de 4x e, então, transformado em um *array C* armazenado na memória *Flash*. Os 256KB de SRAM são mais do que suficientes para alocar a Arena de Tensores necessária para a execução da inferência.

Portanto, a implementação de um modelo LSTM para classificação em tempo real via *TinyML* é não apenas viável, mas uma das principais vantagens estratégicas do *pNANO_v2*, validando-o como uma plataforma de *hardware* adequada para o desenvolvimento de um sistema assistivo inteligente e autônomo.

4.5 Análise da Escalabilidade e Facilidade de Implementação de cada Sistema

A avaliação da facilidade de implementação e do potencial de escalabilidade de cada sistema revela distinções fundamentais que transcendem o custo e o desempenho energético, impactando diretamente a viabilidade de transição de um protótipo de bancada para um produto final robusto.

Do ponto de vista da facilidade de implementação de *hardware*, o *pNANO_v2* apresenta uma vantagem significativa. Sua arquitetura é altamente integrada, notavelmente pela inclusão da IMU (acelerômetro, giroscópio e magnetômetro) diretamente na placa. Esta integração elimina a necessidade de módulos de sensores externos, reduzindo drasticamente a complexidade

da montagem, a fiação e os pontos de falha associados à soldagem de periféricos. Em contrapartida, o *pUNO_v2* exige a conexão manual de módulos externos, como o MPU6050, o que aumenta o tempo de montagem, a suscetibilidade a erros de conexão e o volume final do dispositivo, conforme visto na Seção 4.1.

Contudo, é na escalabilidade funcional e de *software* que o *pNANO_v2* demonstra sua superioridade de forma decisiva. Esta escalabilidade é fundamentada em dois pilares tecnológicos ausentes no *pUNO_v2*: o poder computacional e *TinyML* e a possibilidade de conectar o BLE.

O *pUNO_v2* é baseado no microcontrolador ATmega328P, com especificações modestas suficientes para a coleta e armazenamento de dados, mas inteiramente inadequadas para a execução de algoritmos computacionalmente intensivos, como a LSTM proposta. Em contraste, o *pNANO_v2* é equipado com o SoC nRF52840, cuja capacidade de processamento e memória é robusta e foi especificamente projetada para suportar inferências de ML no dispositivo, um paradigma conhecido como *TinyML*. Conforme um dos objetivos específicos do trabalho, esta capacidade permite que o modelo LSTM de detecção de tremores seja embarcado diretamente no *wearable*. Isso possibilita a classificação em tempo real, garante a privacidade ao não necessitar transmitir dados brutos do paciente e reduz o consumo energético ao evitar a comunicação sem fio contínua.

Sobre conectividade, placa Nano 33 BLE Sense Rev 2 possui BLE 5.0 nativo. Esta funcionalidade, embora não explorada nos testes de consumo, é um vetor de escalabilidade crítico. Ela permite que o dispositivo se comunique de forma energeticamente eficiente com outros dispositivos, como *smartphones* ou concentradores de dados. Isso abre caminho para aplicações clínicas avançadas, como o monitoramento remoto de pacientes, o envio de alertas em tempo real ou a configuração do dispositivo sem a necessidade de conexões físicas.

Em síntese, o *pUNO_v2* demonstra ser um sistema de baixa escalabilidade, cuja facilidade de implementação é limitada pelo seu baixo custo de aquisição, mas dificultada pela necessidade de integração de módulos externos. Ele se qualifica como um "coletor de dados" (*data logger*) eficaz para prototipagem de baixo custo. O *pNANO_v2*, por sua vez, embora tenha uma barreira de custo inicial mais alta, oferece uma implementação de *hardware* mais simples (menos conexões) e uma escalabilidade funcional imensamente superior. Ele é uma plataforma de "computação vestível" (*wearable computing*), capaz de evoluir de um simples coletor para um dispositivo inteligente, autônomo e conectado, alinhado com os requisitos de tecnologias assistivas modernas para o monitoramento de saúde.

4.6 Comparação Final

Este estudo teve como objetivo central realizar uma análise comparativa aprofundada entre duas plataformas de prototipagem, o *pUNO_v2* e o *pNANO_v2*, visando determinar a solução de *hardware* mais eficaz, eficiente e viável para o desenvolvimento de um sistema

embarcado de monitoramento de tremores na DP. Os resultados experimentais, abrangendo análises de custo, atributos físicos, consumo energético, qualidade de captura e escalabilidade de *software*, convergem para uma conclusão clara.

O *pUNO_v2* demonstrou ser uma plataforma de custo notavelmente baixo, quase três vezes mais acessível que seu concorrente. Esta vantagem financeira o posiciona como uma ferramenta viável para a prototipagem inicial, validação de conceitos em estágio embrionário e aplicações pedagógicas. Contudo, este benefício é ofuscado por suas limitações técnicas: seu volume e peso são consideráveis, tornando-o inadequado para o uso contínuo como *wearable*; sua eficiência energética é comprovadamente baixa; e, mais criticamente, sua capacidade de captura de dados falhou em frequências clinicamente relevantes (acima de 2 Hz). Além disso, sua arquitetura de 8 bits o torna inviável para qualquer processamento avançado, limitando-o estritamente à função de um registrador de dados simples e de baixa fidelidade.

O *pNANO_v2*, por outro lado, embora represente um investimento inicial significativamente maior, justificou seu custo em todos os aspectos técnicos. É uma plataforma compacta, leve e energeticamente eficiente, atributos essenciais para a usabilidade e conforto do paciente. Sua arquitetura de sensores integrados simplifica a montagem e garante uma qualidade de sinal superior. Nos testes mecânicos, demonstrou alta precisão na detecção de frequências em toda a faixa de interesse (2-4 Hz) e estabilidade de orientação quase perfeita em longo prazo, graças ao uso do magnetômetro. Mais importante, sua arquitetura de *hardware* avançada o valida como uma plataforma preparada para o futuro, suportando nativamente a implementação de inteligência artificial embarcada via *TinyML* e conectividade sem fio via BLE.

Portanto, ao responder à problemática central deste trabalho, conclui-se que o *pUNO_v2* não se qualifica como uma solução viável para um sistema assistivo robusto, servindo apenas como uma ferramenta introdutória de baixo custo. A plataforma *pNANO_v2* emergiu como a escolha inequivocamente superior, demonstrando ser a base de *hardware* eficaz, eficiente e escalável necessária para o desenvolvimento de uma tecnologia assistiva precisa e inteligente para o monitoramento da DP.

5 Considerações Finais

O presente trabalho concluiu com êxito o desenvolvimento e a validação comparativa de dois protótipos de *wearables* para o monitoramento de tremores na DP, endereçando os objetivos propostos. Foi realizada uma análise multifacetada, comparando os sistemas *pUNO_v2* e *pNANO_v2* em termos de custo, atributos físicos, consumo energético, qualidade de captura e potencial de escalabilidade.

A pesquisa demonstrou que, embora a abordagem de baixo custo do *pUNO_v2* seja acessível, ela falha em fornecer a fidelidade de dados e a capacidade de processamento necessárias para uma aplicação clínica robusta. A plataforma *pNANO_v2*, apesar de seu custo elevado, provou ser tecnicamente superior em todos os critérios de desempenho relevantes: precisão na detecção de frequência, estabilidade de orientação, eficiência energética e adequação para uso como *wearable*. Além disso, sua compatibilidade com *TinyML* e BLE foi validada, estabelecendo-o como a plataforma viável para o desenvolvimento de um dispositivo inteligente.

Identificam-se, contudo, limitações no estudo que abrem caminho para trabalhos futuros. A principal limitação metodológica foi a taxa de amostragem de 10 Hz, que se mostrou insuficiente ao operar muito próxima da frequência de Nyquist, causando falhas na detecção em 4.5 Hz. Além disso, a capacidade nominal das baterias comerciais utilizadas mostrou-se pouco confiável, impactando a previsibilidade da autonomia. Finalmente, esta pesquisa se limitou a uma validação técnica e mecânica, não envolvendo a coleta de dados com pacientes.

Como sugestão para trabalhos futuros, recomenda-se a otimização do *firmware* para alcançar taxas de amostragem mais elevadas, como 20 Hz ou 50 Hz, garantindo a captura fidedigna de todo o espectro do tremor. Sugere-se também uma investigação sobre fornecedores de baterias com capacidade certificada. A etapa subsequente e mais crucial será a transição da validação mecânica para a validação clínica, utilizando o protótipo *pNANO_v2* para a coleta de dados reais em pacientes. Estes dados permitirão, por fim, o treinamento e a validação do modelo LSTM proposto, completando o ciclo de desenvolvimento e aproximando o sistema de uma aplicação assistiva funcional.

Referências

ABADI, M.; AGARWAL, A.; BARHAM, P.; BREVDO, E.; CHEN, Z.; CITRO, C.; CORRADO, G. S.; DAVIS, A.; DEAN, J.; DEVIN, M.; GHEMAWAT, S.; GOODFELLOW, I.; HARP, A.; IRVING, G.; ISARD, M.; JIA, Y.; JOZEFOWICZ, R.; KAISER, L.; KUDLUR, M.; LEVENBERG, J.; MANE, D.; MONGA, R.; MOORE, S.; MURRAY, D.; OLAH, C.; SCHUSTER, M.; SHLENS, J.; STEINER, B.; SUTSKEVER, I.; TALWAR, K.; TUCKER, P.; VANHOUCKE, V.; VASUDEVAN, V.; VIEGAS, F.; VINYALS, O.; WARDEN, P.; WATTENBERG, M.; WICKE, M.; YU, Y.; ZHENG, X. Tensorflow: large-scale machine learning on heterogeneous systems. 2015. Software available from tensorflow.org. Disponível em: <<https://www.tensorflow.org/>>. Acesso em: 08 out. 2025.

AMERINI, I.; BESTAGINI, P.; BONDI, L.; CALDELLI, R. Robust smartphone fingerprint by mixing device sensors features for mobile strong authentication. In: . [S.l.: s.n.], 2016. v. 28.

Arduino. *What is Arduino?* 2018. Acesso em: 30 mar. 2025. Disponível em: <<https://www.arduino.cc/en/Guide/Introduction/>>.

Arduino. *One board to rule them all: History of the Arduino UNO*. 2021. Acesso em: 30 mar. 2025. Disponível em: <<https://blog.arduino.cc/2021/12/09/one-board-to-rule-them-all-history-of-the-arduino-uno/?queryID=582dd1d4ea530a77d7988f12f7a9311d>>.

Arduino. *Arduino Nano 33 BLE Sense Rev2: Documentation*. 2025. <<https://docs.arduino.cc/hardware/nano-33-ble-sense-rev2>>. Acesso em: 21 out. 2025.

Arduino. *Arduino UNO R3: Documentation*. 2025. <<https://docs.arduino.cc/hardware/uno-rev3>>. Acesso em: 21 out. 2025.

ARMSTRONG; J., M.; OKUN; S., M. Diagnosis and treatment of parkinson disease: a review. *JAMA*, v. 323, n. 6, p. 548–560, fev. 2020.

BOSCH SENSORTEC. *BMI270 Datasheet*. [S.l.], 2019. Rev. 1.

Cala Health. *Cala Trio Therapy*. 2024. Acessado em: 06 ago. 2025. Disponível em: <<https://calahealth.com/support/help-center/about-cala/cala-trio-therapy/>>.

CHAI, T.; DRAXLER, R. R. Root mean square error (rmse) or mean absolute error (mae)? – arguments against avoiding rmse in the literature. *Geoscientific Model Development*, v. 7, n. 3, p. 1247–1250, 2014. Acesso em: 08 out. 2025. Disponível em: <<https://gmd.copernicus.org/articles/7/1247/2014/>>.

COOLEY, J. W.; TUKEY, J. W. An algorithm for the machine calculation of complex fourier series. *Mathematics of Computation*, v. 19, n. 90, p. 297–301, 1965.

ERIKA, R.; CARLO, M.; FILIPPO, C. How wearable sensors can support parkinson's disease diagnosis and treatment: a systematic review. *Frontiers in Neuroscience*, v. 11, 2017.

FEIGIN, V. L.; VOS, T.; NICHOLS, E.; OWOLABI, M. O.; CARROLL, W. M.; DICHGANS, M.; DEUSCHL, G.; PARMAR, P.; BRAININ, M.; MURRAY, C. The global burden of

neurological disorders: translating evidence into policy. *The Lancet Neurology*, v. 19, n. 3, p. 255–265, mar. 2020.

FERREIRA, J. J.; GODINHO, C.; SANTOS, A. T.; DOMINGOS, J.; ABREU, D.; LOBO, R.; GONÇALVES, N.; BARRA, M.; LARSEN, F.; FAGERBAKKE Øyvind; AKEREN, I.; WANGEN, H.; SERRANO, J. A.; WEBER, P.; THOMS, A.; MECKLER, S.; SOLLINGER, S.; UEM, J. van; HOBERT, M. A.; MAIER, K. S.; MATTHEW, H.; ISAACS, T.; DUFFEN, J.; GRAESSNER, H.; MAETZLER, W. Quantitative home-based assessment of parkinson's symptoms: the sense-park feasibility and usability study. *BMC Neurol.*, 2015.

FRAUTSCHI, S. C.; OLENICK, R. P.; APOSTOL, T. M. *The Mechanical Universe: mechanics and heat and Advanced Edition*. Cambridge: Cambridge University Press, 2007. ISBN 9780521715904. Disponível em: <<https://books.google.com.br/books?id=ZTnxQGJ1fHMC>>.

Great Lakes NeuroTechnologies. *Kinesia 360*. 2024. Acessado em: 06 ago. 2025. Disponível em: <<https://www.glneurotech.com/products/kinesia-360/>>.

HALL, E. H. On a new action of the magnet on electric currents. *American Journal of Mathematics*, v. 2, n. 4, p. 287–292, 1879.

HOCHREITER, S.; SCHMIDHUBER, J. Long short-term memory. *Neural Computation*, v. 9, n. 8, p. 1735–1780, dez. 1997.

HOEHN, M. M.; YAHR, M. D. Parkinsonism: onset and progression and mortality. *Neurology*, v. 17, 1967.

HOLMES, S. E.; HONHAR, P.; TINAZ, S.; NAGANAWA, M.; HILMER, A. T.; GALLEZOT, J.-D.; DIAS, M.; YANG, Y.; TOYONAGA, T.; ESTERLIS, I.; MECCA, A.; DYCK, C. V.; HENRY, S.; ROPCHAN, J.; NABULSI, N.; LOUIS, E. D.; COMLEY, R.; FINNEMA, S. J.; MATUSKEY, R. E. C. . D. Synaptic loss and its association with symptom severity in parkinson's disease. *npj Parkinson's Disease*, Springer Nature, v. 10, n. 42, 2024.

JAFARI, R. *Hands-On Data Preprocessing in Python: Learn how to effectively prepare data for successful data analytics*. [S.l.: s.n.], 2022.

JAMES, P. *An essay on the shaking palsy*. London: Whittingham and Rowland, 1817.

KALIA, L. V.; LANG, A. E. Parkinson's disease. *The Lancet*, v. 386, n. 9996, p. 896–912, 2015. Acesso em: 08 out. 2025. Disponível em: <<https://www.sciencedirect.com/science/article/pii/S0140673614613933>>.

KLEIN, M. O.; BATTAGELLO, D. S.; CARDOSO, A. R.; HAUSER, D. N.; BITTENCOURT, J. C.; CORREA, R. G. Dopamine: functions and signaling and and association with neurological diseases. *Cellular and Molecular Neurobiology*, v. 39, p. 31–59, 2019.

Movement Disorder Society Task Force on Rating Scales for Parkinson's Disease. The unified parkinson's disease rating scale: status and recommendations. *Movement Disorders*, v. 18, p. 738–750, jan. 2003.

OLAH, C. *Understanding LSTM networks*. 2015. Acesso em: 08 out. 2025. Disponível em: <<https://colah.github.io/posts/2015-08-Understanding-LSTMs/>>.

PAHAPILL, P. A.; LOZANO, A. M. The pedunculo pontine nucleus in parkinson's disease. *Brain*, v. 123, n. 9, 2000.

- PASSARO, V. M. N.; CUCCOVILLO, A.; VAIANI, L.; CARLO, M. D.; CAMPANELLA, C. E. Gyroscope technology and applications: a review in the industrial perspective. *Sensors*, v. 17, n. 10, 2017. Acesso em: 08 out. 2025. Disponível em: <<https://www.mdpi.com/1424-8220/17/10/2284>>.
- PETER, C. *How gyroscopes work*. Queensland University of Technology, 2025. Acesso em: 08 out. 2025. Disponível em: <<https://robotacademy.net.au/lesson/how-gyroscopes-work/>>.
- PETER, C. *Using accelerometers*. Queensland University of Technology, 2025. Acesso em: 08 out. 2025. Disponível em: <<https://robotacademy.net.au/lesson/using-accelerometers/>>.
- PETER, C. *Using magnetometers*. Queensland University of Technology, 2025. Acesso em: 08 out. 2025. Disponível em: <<https://robotacademy.net.au/lesson/using-magnetometers/>>.
- PROAKIS, J. G.; MANOLAKIS, D. G. *Digital signal processing: principles and algorithms and applications*. 3. ed.. ed. Upper Saddle River: Prentice-Hall and Inc., 1996. ISBN 0133737624.
- PULLIAM, C. L.; HELDMAN, D. A.; BROKAW, E. B.; MERA, T. O.; MARI, Z. K.; BURACK, M. A. Continuous assessment of levodopa response in parkinson's disease using wearable motion sensors. *IEEE Transactions on Biomedical Engineering*, v. 65, n. 1, p. 159–164, 2018.
- ROWBERG; JEFF. *MPU6050.h*. i2cdevlib, 2022. Acesso em: 08 out. 2025. Disponível em: <<https://www.i2cdevlib.com/>>.
- SCHMIDT, R. M. *Recurrent Neural Networks (RNNs): a gentle introduction and overview*. 2019.
- SHANNON, C. E. Communication in the presence of noise. *Proceedings of the IRE*, IEEE, v. 37, n. 1, p. 10–21, jan. 1949. Disponível em: <<https://doi.org/10.1109/jrproc.1949.232969>>.
- SONMEZOGLU, S.; FLADER, I. B.; CHEN, Y.; SHIN, D. D.; KENNY, T. W.; HORSLEY, D. A. Dual-resonator mems lorentz force magnetometer based on differential frequency modulation. *2017 IEEE International Symposium on Inertial Sensors and Systems (INERTIAL)*, p. 160–163, 2017. Disponível em: <<https://api.semanticscholar.org/CorpusID:24939870>>.
- STAUEMEYER, R. C.; MORRIS, E. R. *Understanding LSTM – a tutorial into Long Short-Term Memory Recurrent Neural Networks*. 2019.
- TEJA, V. M. S.; SAI, B.; VEERAPANDU, G.; SINGH, M. K. A comparative study of different iot sensors. In: YADAV, S.; HALEEM, A.; ARORA, P. K.; KUMAR, H. (Ed.). *Proceedings of Second International Conference in Mechanical and Energy Technology*. Singapore: Springer Nature Singapore, 2023. p. 53–61. ISBN 978-981-19-0108-9.
- VENKATANARAYANAN, A.; SPAIN, E. 13.03 - review of recent developments in sensing materials. In: HASHMI; SALEEM; BATALLA; FERREIRA, G.; TYNE, V.; J., C.; YILBAS; BEKIR (Ed.). *Comprehensive Materials Processing*. Oxford: Elsevier, 2014. p. 47–101. ISBN 978-0-08-096533-8. Acesso em: 08 out. 2025. Disponível em: <<https://www.sciencedirect.com/science/article/pii/B9780080965321013030>>.
- WALKER, R.; GUAN, Y.; JACKSON, D.; DISMORE, L.; MONTAGUE, K.; GUERREIRO, T.; CARVALHO, L.; WINDER, L.; NICOLAU, H.; MORRIS, R.; FLETCHER, K.; RODRIGUES, A. *Cue Band and Wearables for Parkinson's*. 2024. Acessado em: 02 ago. 2025. Disponível em: <<https://cue.band/>>.

WARMERDAM, E.; HAUSDORFF, J. M.; ATRSAEI, A.; ZHOU, Y.; MIRELMAN, A.; AMINIAN, K.; ESPAY, A. J.; HANSEN, C.; EVERS, L. J. W.; KELLER, A.; LAMOTH, C.; PILOTTO, A.; ROCHESTER, L.; SCHMIDT, G.; BLOEM, B. R.; MAETZLER, W. Long-term unsupervised mobility assessment in movement disorders. *E-Pub and Lancet Neurol.*, 2020.

ZAITCHIK, E. *Exploring SG90 and MG90S and and MG996R servos*. 2024. Acesso em: 08 out. 2025. Disponível em: <https://zaitronics.com.au/blogs/guides/exploring-sg90-mg90s-and-mg996r-servos?srltid=AfmBOopHKeRZrAyCEHNZ1t-_3DTp9C7m3J1s-ZPSIH-p7fLYdtWti>.

ZAMBOTTI, M. de; GOLDSTEIN, C.; COOK, J.; MENGHINI, L.; ALTINI, M.; CHENG, P.; ROBILLARD, R. State of the science and recommendations for using wearable technology in sleep and circadian research. *Sleep*, v. 47, n. 4, dez. 2023. ISSN 0161-8105.