UNIVERSIDADE FEDERAL DE OURO PRETO

ESCOLA DE MINAS

DEPARTAMENTO DE ENGENHARIA MECÂNICA

Sávio José Ferreira Melo

Aplicação de rede neural artificial no processamento de sinais de vibração para detecção de anomalias mecânicas

OURO PRETO - MG
2026

Sávio José Ferreira Melo

Aplicação de rede neural artificial no processamento de sinais de vibração para detecção de anomalias mecânicas

Monografia apresentada ao Curso de Graduação em Engenharia Mecânica da Universidade Federal de Ouro Preto como requisito para a obtenção do título de Engenheiro Mecânico.

Orientador: Dr. Jonathas Haniel Castro Silva.

Coorientador: Dr. Gustavo Paulinelli Guimarães.

OURO PRETO – MG
2026

SISBIN - SISTEMA DE BIBLIOTECAS E INFORMAÇÃO

M528a Melo, Savio Jose Ferreira.
Aplicação de rede neural artificial no processamento de sinais de vibração para detecção de anomalias mecânicas. [manuscrito] / Savio Jose Ferreira Melo. - 2026.
99 f.: il.: color., gráf., tab..

Orientador: Prof. Dr. Jonathas Haniel Silva.
Coorientador: Prof. Dr. Gustavo Paulinelli Guimarães.
Monografia (Bacharelado). Universidade Federal de Ouro Preto. Escola de Minas. Graduação em Engenharia Mecânica .

1. mANUTENÇÃO - Manutenção preditiva. 2. Vibração - Análise. 3. Redes neurais (Computação) - Redes neurais LSTM. 4. Internet das coisas - Monitoramento em Tempo Real (Real-time Monitoring). 5. Inteligência artificial. I. Silva, Jonathas Haniel. II. Guimarães, Gustavo Paulinelli. III. Universidade Federal de Ouro Preto. IV. Título.

CDU 621

Bibliotecário(a) Responsável: Maristela Sanches Lima Mesquita - CRB-1716

MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DE OURO PRETO
REITORIA
ESCOLA DE MINAS
DEPARTAMENTO DE ENGENHARIA MECANICA

FOLHA DE APROVAÇÃO

Sávio José Ferreira Melo

Aplicação de rede neural artificial no processamento de sinais de vibração para detecção de anomalias mecânicas

Monografia apresentada ao Curso de Engenharia Mecânica da Universidade Federal
de Ouro Preto como requisito parcial para obtenção do título de Engenheiro Mecânico

Aprovada em 09 de fevereiro de 2026

Membros da banca

Prof. Dr. - Jonathas Haniel Castro Silva - Orientador (UFOP)
Prof. Dr. - Gustavo Paulinelli Guimarães - Coorientador (UFOP)
Prof. Dr. - André Luiz Carvalho Ottoni - (UFOP)
Prof. Dr. - Diogo Antônio de Sousa - (UFOP)

Jonathas Haniel Castro Silva, orientador do trabalho, aprovou a versão final e autorizou seu depósito na Biblioteca Digital de Trabalhos de Conclusão de Curso da UFOP em 09/02/2026

Documento assinado eletronicamente por **Jonathas Haniel Castro Silva**, **PROFESSOR DE MAGISTERIO SUPERIOR**, em 13/02/2026, às 15:49, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

A autenticidade deste documento pode ser conferida no site http://sei.ufop.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0 , informando o código verificador **1055309** e o código CRC **62DB12B5**.

R. Diogo de Vasconcelos, 122, - Bairro Pilar Ouro Preto/MG, CEP 35402-163
Telefone: (31)3559-1533 - www.ufop.br

A Deus, dedico mais esta etapa vencida. Aos meus pais, dedico este trabalho pelo apoio incondicional, amor e por terem sido a base de tudo o que sou.

AGRADECIMENTO

Agradeço primeiramente a Deus, por me dar forças, saúde e perseverança para superar os obstáculos ao longo desta graduação.

Aos meus pais, **José Vitorino de Melo Júnior** e **Maria Izabel Ferreira Melo**, que sempre foram meus maiores incentivadores. Obrigado por todo o sacrifício, pelo apoio incondicional e por acreditarem no meu potencial mesmo nos momentos mais difíceis. Esta conquista também é de vocês. À minha irmã **Maria Flávia Ferreira Melo**, e a todos os meus familiares, pelo apoio.

Ao meu orientador, Prof. **Jonathas Haniel Castro Silva** e meu co-orientador **Gustavo Paulinelli Guimarães** pela paciência, confiança e pelas valiosas diretrizes acadêmicas. Agradeço por me guiar no complexo mundo da análise de vibrações e da inteligência artificial, incentivando-me a buscar uma solução robusta e inovadora.

Aos meus amigos e colegas de turma, em especial **Matheus Henrique**, pelo companheirismo, pelas horas de estudo em conjunto e pelo apoio mútuo durante os desafios do curso.

Agradeço também à minha namorada **Maria Fernanda**, pela compreensão nos momentos de ausência e pelo incentivo constante durante as longas noites de pesquisas e testes.

Por fim, agradeço a todos que, direta ou indiretamente, contribuíram para a realização deste trabalho e para o meu crescimento pessoal e profissional.

“O cientista descreve o que existe; o engenheiro cria o que nunca existiu. ”.

Theodore von Kármán

“Qualquer tecnologia suficientemente avançada é indistinguível de magia. ”.

Arthur C. Clarke

RESUMO

O presente trabalho investiga a aplicação de técnicas de Inteligência Artificial para o aprimoramento da manutenção preditiva em sistemas rotativos, abordando a necessidade da Indústria 4.0 por diagnósticos automatizados e precisos. O problema de pesquisa concentra-se na limitação dos métodos convencionais de monitoramento de vibração, que muitas vezes detectam a anomalia, mas falham em identificar sua causa raiz específica. O objetivo principal é desenvolver e validar um sistema integrado de baixo custo, capaz de classificar anomalias mecânicas em tempo real utilizando uma Rede Neural Recorrente do tipo *Long Short-Term Memory* (LSTM). A metodologia consistiu na instrumentação de uma bancada experimental do tipo viga engastada, equipada com acelerômetro e módulo de aquisição de dados, onde foram simuladas condições operacionais distintas, incluindo estado normal, desbalanceamentos por massa, folga mecânica e deslocamento estrutural. Desenvolveu-se um *software* proprietário em linguagem Python que unifica a coleta de dados e a inferência da rede neural, utilizando enjanelamento deslizante no domínio do tempo para processamento contínuo. Os resultados demonstraram que o modelo LSTM atingiu excelência na classificação das anomalias , superando a análise puramente estatística com uma acurácia de 100% em ambiente controlado. Adicionalmente, testes de robustez com injeção de ruído artificial (SNR de 15 dB) comprovaram a resiliência do algoritmo, que manteve uma acurácia global de 90%, sustentando a assertividade diagnóstica em anomalias críticas mesmo sob degradação severa do sinal. Conclui-se que a abordagem proposta oferece uma solução viável e eficiente para o monitoramento de condição, permitindo a identificação instantânea de anomalias incipientes e proporcionando maior confiabilidade à gestão de ativos industriais.

Palavras-chave: Manutenção preditiva. Análise de vibração. Redes neurais LSTM. Monitoramento em tempo real. Inteligência artificial.

ABSTRACT

This work investigates the application of Artificial Intelligence techniques for the enhancement of predictive maintenance in rotating systems, addressing the Industry 4.0 demand for automated and precise diagnostics. The research problem focuses on the limitations of conventional vibration monitoring methods, which often detect the anomaly but fail to identify its specific root cause. The main objective is to develop and validate a low-cost integrated system capable of classifying mechanical anomalies in real-time using a Recurrent Neural Network of the Long Short-Term Memory (LSTM) type. The methodology consisted of instrumenting a cantilever beam experimental bench equipped with an accelerometer and a data acquisition module, where distinct operating conditions were simulated, including normal state, mass unbalance, mechanical looseness, and structural displacement. Custom software was developed in Python to unify data collection and neural network inference, utilizing time-domain sliding windowing for continuous processing. Results demonstrated that the LSTM model achieved excellence in anomaly classification, surpassing purely statistical analysis with 100% accuracy in a controlled environment. Additionally, robustness tests with artificial noise injection (15 dB SNR) proved the algorithm's resilience, maintaining a global accuracy of 90% and sustaining diagnostic reliability for critical anomalies even under severe signal degradation. It is concluded that the proposed approach offers a viable and efficient solution for condition monitoring, allowing for the instant identification of incipient anomalies and providing greater reliability to industrial asset management.

Keywords: Predictive maintenance. Vibration analysis. LSTM neural networks. Real-time monitoring. Artificial intelligence.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SÍMBOLOS

A_{max}	Amplitude máxima do sinal
B	Viés (bias) do neurónio artificial
C_{t-1}	Estado anterior da célula de memória (LSTM)
Ct	Estado atual da célula de memória (LSTM)
f_{max}	Frequência máxima de interesse no sinal
fs	Frequência de amostragem
h_{t-1}	Estado oculto anterior (LSTM)
h_t	Estado oculto atual ou saída da célula (LSTM)
n(t)	Ruído com distribuição normal padrão
W	Pesos sinápticos
X	Sinais de entrada do neurónio
X_t	Vetor de entrada da rede no instante t
Y	Saída do neurónio
Z	Soma ponderada dos sinais de entrada
α	Fator de intensidade do ruído injetado
σ	Função de ativação sigmoide
φ	Função de ativação genérica
c	Índice da classe atual no somatório

SUMÁRIO

1 INTRODUÇÃO

1.1 Formulação do Problema

A parada inesperada de um equipamento pode comprometer a operação de linhas de produção inteiras, gerando prejuízos incalculáveis como perda de matéria-prima, sobrecarga em componentes periféricos e interrupção de processos críticos. No contexto da Indústria 4.0, a complexidade dos equipamentos e o volume massivo de dados gerados tornam a prevenção dessas anomalias um desafio ainda maior. Segundo Mobley (2002), interrupções não planejadas podem representar até 30% dos custos operacionais totais de uma planta.

A manutenção preditiva destaca-se como a alternativa eficiente para mitigar esses problemas. Conforme Nepomuceno (1989), essa abordagem busca estabelecer parâmetros operacionais que sinalizem anomalias incipientes.

Existem diversas abordagens consolidadas para este fim. Benbouzid (2000) destaca a eficácia da análise de corrente elétrica para motores de indução, enquanto Bagavathiappan *et al*. (2013) apontam a termografia infravermelha para a detecção de sobreaquecimentos e Moubray (1997) descreve o uso do ultrassom para vazamentos. No entanto, métodos tradicionais isolados muitas vezes mostram-se limitados, sendo excessivamente caros, analógicos ou dependentes de uma expertise técnica escassa.

Neste cenário, a análise de vibração combinada com Inteligência Artificial apresenta um equilíbrio entre custo, precisão e aplicabilidade. De acordo com Rao (2011), a vibração é o parâmetro que responde mais rapidamente a mudanças na condição dinâmica da máquina, sendo uma fonte rica de informações.

Diferentemente de abordagens que utilizam Redes Neurais Convolucionais (CNNs) baseadas em imagens de espectrogramas, como as estudadas por Zhang et al. (2017), ou de técnicas de *Support Vector Machines* (SVM) que, segundo Liu (2018), exigem extração manual de características, este trabalho propõe o uso de sinais brutos no domínio do tempo.

Portanto, busca-se desenvolver uma metodologia de avaliação da integridade de máquinas utilizando uma Rede Neural Recorrente do tipo LSTM (Long Short-Term Memory). Como afirmado por Huang et al. (1994) e corroborado por estudos recentes em séries temporais de Lei

(2020), essas estruturas computacionais são capazes de aprender dependências temporais complexas.

O sistema proposto visa interpretar anomalias nos sinais de vibração para diagnosticar antecipadamente anomalias e desgastes, prevenindo paradas não programadas e automatizando a tomada de decisão na manutenção.

Como aplicar rede neural artificial no processamento de sinais de vibração para detecção de anomalias mecânicas?

1.2 Justificativa

A manutenção preditiva apresenta vantagens econômicas e operacionais inegáveis em relação aos métodos corretivos e preventivos tradicionais. Conforme aponta Nepomuceno (1989), sua aplicação correta tem o potencial de reduzir de 15% a 20% os custos globais com reparos, otimizando a vida útil dos ativos e diminuindo drasticamente o número de paradas não planejadas.

Apesar desses benefícios, Meyer zu Wickern (2019) adverte que a implementação dessa estratégia enfrenta barreiras significativas, como o alto custo de sensores industriais e a complexidade na interpretação dos dados. O autor ressalta que prever anomalias com precisão permanece um desafio devido à variabilidade das condições operacionais e à natureza não linear dos sinais mecânicos, o que muitas vezes inviabiliza o uso de modelos matemáticos rígidos.

Existem outras abordagens consolidadas para o monitoramento de ativos. Benbouzid (2000) destaca a eficácia da análise de corrente para motores, enquanto Bagavathiappan et al. (2013) apontam a termografia como ideal para quadros elétricos e Moubray (1997) cita o ultrassom para a detecção de vazamentos. No entanto, segundo Rao (2011), a análise de vibração destaca-se por apresentar o melhor equilíbrio entre custo, precisão e aplicabilidade em sistemas rotativos.

Nesse contexto, a integração da vibração com a Inteligência Artificial justifica-se pela necessidade de automatizar o diagnóstico. Para Andrade (2021), as Redes Neurais Artificiais (RNA) superam as limitações humanas ao processar grandes volumes de dados históricos.

Além disso, diferentemente das abordagens baseadas em imagens de espectrogramas estudadas por Zhang et al. (2017), que exigem alto processamento, ou dos métodos de *Support Vector Machines* (SVM) descritos por Liu (2018), que dependem de extração manual de características, a utilização de redes LSTM proposta neste trabalho permite a análise direta do sinal bruto no domínio do tempo.

Portanto, este trabalho justifica-se por desenvolver uma metodologia não invasiva, de baixo custo computacional e capaz de gerar diagnósticos contínuos. Essa abordagem democratiza o acesso a tecnologias da Indústria 4.0, oferecendo uma contribuição prática para a redução de custos e aumento da confiabilidade em processos produtivos.

1.3 Objetivos

1.3.1 Geral

Desenvolver uma metodologia baseada em redes neurais para o monitoramento de anomalias a partir da análise de sinais de vibração de máquinas.

1.3.2 Específicos

- Investigar e estudar as principais técnicas de monitoramento de anomalias, métodos de aquisição e tratamento de dados de vibração e aplicações de redes neurais artificiais em diagnóstico de máquinas;
- Definir e coletar dados de vibração de um sistema padrão determinado, para alimentar a rede neural;
- Implementar uma rede neural capaz de identificar padrões anômalos e associá-los a anomalias específicas;
- Comparar a eficácia da metodologia proposta com outras abordagens tradicionais de manutenção;

- Analisar as limitações e vantagens da aplicação de redes neurais no contexto industrial.

1.4 Estrutura do Trabalho

O trabalho está dividido em cinco capítulos. O primeiro capítulo, Introdução, apresenta a formulação do problema , a justificativa para a realização do trabalho , e define os objetivos geral e específicos.

O segundo capítulo, Revisão Bibliográfica , aborda a fundamentação teórica, explorando conceitos de previsão de anomalias e manutenção preditiva , a teoria de vibrações , os princípios de aprendizado de máquina e a aplicação de redes neurais, com foco no modelo LSTM, para o monitoramento de anomalias.

O terceiro capítulo descreve a Metodologia. São detalhados o tipo de pesquisa , a montagem da bancada experimental , os equipamentos , o processo de aquisição e tratamento dos dados de vibração , e a implementação da rede neural.

O quarto capítulo apresenta os Resultados e Discussões. Esta seção analisa os sinais de vibração coletados , avalia o desempenho do modelo LSTM e valida o sistema de monitoramento integrado.

Finalmente, o quinto capítulo, Conclusão e Recomendações , consolida os resultados, responde à problemática da pesquisa , avalia o cumprimento dos objetivos e propõe recomendações para trabalhos futuros.

2 REVISÃO BIBLIOGRÁFICA

Gu et al. (2021) e Branco et al. (2022), demonstram em seus trabalhos que os métodos de previsão de anomalias em máquinas industriais têm se desenvolvido rapidamente com o avanço das técnicas de aprendizado de máquina.

Este trabalho faz uma abordagem combinando métodos tradicionais e modernos, com destaque para redes neurais recorrentes do tipo LSTM. Neste capítulo são apresentados os principais métodos de previsão de anomalias , os fundamentos do aprendizado de máquina aplicados a séries sinais vibratórios contínuos, além dos conceitos essenciais de vibração utilizados no diagnóstico de anomalias .

Também são explicadas as técnicas de aquisição de dados de vibração, incluindo sensores, posicionamento e coleta dos sinais, e os procedimentos de pré-processamento utilizados para preparar os dados antes de alimentar os modelos preditivos.

2.1 Redes neurais no monitoramento de anomalias

Huang e Zhang (2003) afirma, que nas últimas décadas, a previsão de anomalias em sistemas industriais têm migrado de abordagens estatísticas tradicionais para métodos baseados em inteligência artificial, com destaque para as redes neurais artificiais (RNAs). Essas redes são capazes de modelar relações complexas e não lineares entre variáveis operacionais e padrões de anomalia, mesmo em cenários com ruído e incertezas nos dados.

Exemplos recentes incluem, Gu *et al* (2021) com a aplicação de LSTM com múltiplos sensores para diagnóstico de anomalias em rolamentos e Afridi *et al* (2023) com o uso direto de sinais de vibração para prognóstico dos mesmos. No setor elétrico, Branco *et al* (2022) utiliza a arquitetura Wavelet-LSTM, tem sido usada para prever anomalias em redes de energia. Já no setor de mineração, Dias (2020) utiliza redes neurais foram aplicadas à previsão de anomalias em caminhões fora de estrada, com foco em manutenção preditiva.

Segundo Huang e Zhang (1994), as RNAs representam uma alternativa promissora para a manutenção preditiva, por sua capacidade de aprender com dados históricos e adaptar-se a diferentes condições operacionais. Entre as arquiteturas mais eficazes, destacam-se as redes

neurais recorrentes (RNNs), especialmente o modelo LSTM (Long Short-Term Memory), proposto por Hochreiter e Schmidhuber (1997), que é capaz de lidar com dependências temporais de longo prazo.

Como demonstrado por Afridi *et al.* (2023), Branco *et al*. (2022), Gu et al. (2021) e Dias (2020), modelos LSTM têm sido aplicados com sucesso na previsão antecipada de anomalias em motores, rolamentos e até redes elétricas, com desempenho superior a métodos baseados apenas em limites ou regressão. Afridi *et al*. (2023) demonstraram que é possível treinar LSTM diretamente com dados brutos de vibração, sem necessidade de pré-processamento, mantendo precisão mesmo sob sinais ruidosos.

Gu *et al.* (2021) combinaram a transformada discreta de wavelet (DWT) com redes LSTM em sistemas multisensores para diagnóstico de anomalias em rolamentos. Segundo Gu *et al.* (2021) a DWT permitiu extrair informações em diferentes escalas de frequência, enquanto a LSTM capturou a evolução temporal dos sinais, resultando em alta acurácia na detecção de anomalias.

Branco *et al*. (2022) propuseram uma abordagem Wavelet-LSTM para previsão de anomalias em redes elétricas de potência. Eles afirmam a decomposição multiescala dos sinais permitiu destacar padrões sutis de degradação, que foram eficazmente capturados pela LSTM, melhorando significativamente o desempenho preditivo do sistema em comparação a métodos tradicionais.

Já Dias (2020) desenvolveu um modelo preditivo de anomalias aplicado a caminhões fora de estrada, utilizando algoritmos de aprendizado de máquina treinados com dados operacionais históricos. A pesquisa de Dias (2020), avaliou diferentes técnicas, como redes neurais artificiais, árvores de decisão e XGBoost, sendo este último o que apresentou melhor desempenho. Dias (2020) afirma que modelo foi capaz de prever anomalias com até 20 dias de antecedência, atingindo precisão média de 97,90%.

Como demonstrado por Afridi *et al.* (2023), Branco *et al*. (2022), Gu et al. (2021) e Dias (2020), esses estudos indicam que a integração de LSTM com técnicas como wavelet ou *envelope analysis* fortalece a robustez dos modelos em ambientes industriais reais. Além disso, Bhatia e

Gupta (2025) afirmam que atualmente *frameworks* como *PyTorch* e *TensorFlow* têm simplificado a implementação de análises em tempo real, aproximando a predição de anomalias da prática industrial em manutenção preditiva baseada em dados.

2.2 Previsão de anomalias

A Previsão e prevenção de anomalias é uma abordagem orientada ao monitoramento contínuo das condições operacionais de máquinas e equipamentos, com o objetivo de identificar anomalias potenciais antes que se tornem críticas.

Segundo Xenos (2004), a manutenção preditiva é uma estratégia que permite minimizar paradas não programadas, otimizar o uso de recursos e aumentar a confiabilidade dos ativos industriais. Xenos (2004) diz também que diferentemente da manutenção corretiva ou preventiva tradicional, a preditiva baseia-se em dados reais do equipamento como vibração, temperatura e consumo de energia para planejar intervenções apenas quando necessário, evitando tanto a anomalia quanto a substituição prematura de componentes.

Para Xenos (2004) na manutenção preditiva, os parâmetros de análise representam as variáveis monitoradas que indicam o estado operacional de um sistema ao longo do tempo. Esses parâmetros são fundamentais para a detecção precoce de anomalias e a tomada de decisão sobre intervenções corretivas planejadas. Xenos (2004) também afirma que a efetividade da manutenção preditiva está diretamente associada à escolha adequada e ao acompanhamento contínuo desses parâmetros.

Entre os principais parâmetros destacam-se para Xenos (2004):

- Temperatura: É um dos sinais mais comuns de degradação. Aumento anormal da temperatura em mancais, motores ou redutores pode indicar atrito excessivo, lubrificação inadequada ou anomalias elétricas.
- Vibração: Alterações nos níveis de vibração são indicativas de desbalanceamentos, desalinhamentos, folgas ou desgaste em componentes rotativos. Como já tratado, pode ser monitorada por deslocamento, velocidade ou aceleração.

- Pressão: Em sistemas hidráulicos ou pneumáticos, quedas ou variações não justificadas na pressão operacional podem apontar para obstruções, vazamentos ou anomalias de válvulas.
- Corrente elétrica: O aumento do consumo elétrico pode estar associado a sobrecarga mecânica ou anomalias internas no motor.

Xenos (2004) também destaca a importância da análise de tendências: mais do que avaliar valores absolutos, a manutenção preditiva deve se concentrar na evolução temporal dos parâmetros monitorados. Uma pequena variação contínua pode ser mais significativa do que um valor alto isolado. Para Xenos (2004) ferramentas estatísticas e sistemas de aquisição de dados são essenciais para permitir a análise histórica e a predição de comportamento futuro.

Xenos (2004) conclui que a escolha dos parâmetros deve considerar fatores como o tipo de equipamento, a criticidade da função, o histórico de anomalias e o ambiente operacional. Além disso, é necessário definir limites de alarme e de ação, que indicam quando a condição do equipamento ultrapassa níveis aceitáveis e exige intervenção.

2.3 Aprendizado de máquina

Huang e Zhang (2003) mostra que o aprendizado de máquina tem se consolidado como uma ferramenta essencial em sistemas industriais modernos, permitindo que algoritmos identifiquem padrões complexos em grandes volumes de dados operacionais.

Segundo Huang e Zhang (2003), essas técnicas oferecem grande potencial para aprimorar a tomada de decisão em processos de fabricação e manutenção, contribuindo para aumento da confiabilidade, redução de custos e melhoria da qualidade dos produtos. Ao automatizar a análise de dados e permitir a adaptação contínua dos modelos, o aprendizado de máquina se torna particularmente útil em ambientes industriais dinâmicos, como no monitoramento de condições e na previsão de anomalias em equipamentos.

Cormen *et al* (2009), diz que um algoritmo pode ser entendido como uma sequência finita de instruções claras e ordenadas, projetadas para resolver um problema específico ou executar uma tarefa definida. Essa definição destaca a importância da precisão e da ordem na execução,

garantindo que o processo seja reproduzível e eficiente. Em engenharia e ciência da computação, os algoritmos são a base para o desenvolvimento de *softwares* e sistemas automatizados.

Para Cormen *et al.* (2009), algoritmos podem ser compreendidos como procedimentos computacionais bem definidos, compostos por uma sequência finita de etapas que transformam dados de entrada em resultados de saída. Essa estrutura é essencial para a resolução sistemática de problemas computacionais e está na base do funcionamento de programas e sistemas modernos.

De acordo com Nicolau e Mancini (2005), algoritmos consistem em sequências ordenadas e finitas de instruções não ambíguas, elaboradas para resolver um problema específico. Essa definição ressalta a importância da estrutura lógica e da clareza das instruções, características que tornam o algoritmo compreensível e aplicável por qualquer pessoa ou máquina que o execute. Assim, o algoritmo funciona como um guia passo a passo para alcançar uma solução eficaz.

a. Rede neural artificial

De acordo com Huang e Zhang (2003), redes neurais artificiais são estruturas computacionais formadas por camadas de nós (neurônios artificiais) que processam sinais de entrada para reconhecer padrões, classificar eventos ou prever comportamentos. Huang e Zhang (2003) demonstra que no contexto da manutenção preditiva, as RNAs são treinadas para detectar sinais sutis de anomalia em vibração, corrente, som e temperatura, mesmo em ambientes industriais ruidosos.

O diagrama na Figura 2.1 apresenta a estrutura básica de um neurônio artificial, que é o elemento fundamental das redes neurais artificiais (RNAs).

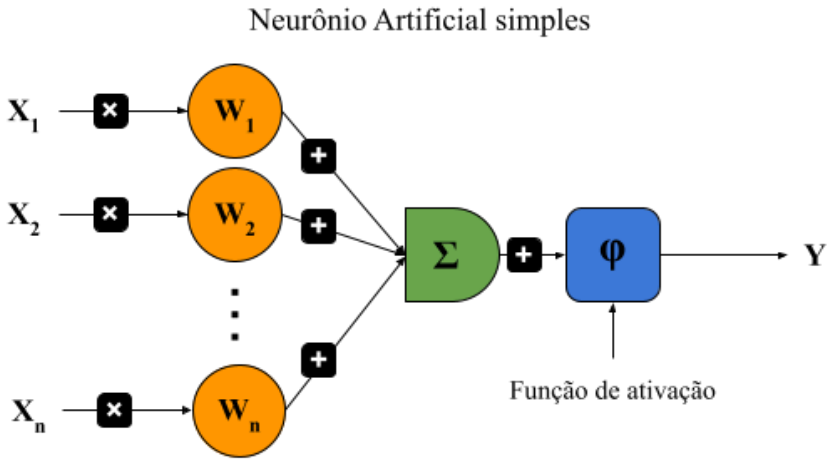

Figura 2.1: Neurônio Artificial Simples

Fonte: Adaptação de McCulloch & Pitts *apud* Minussi & Lotufo (2008)

Como mostrado na Figura 2.1, cada neurônio recebe múltiplos sinais de entrada X1, X2, ..., Xn, que são ponderados por coeficientes também chamados de pesos, W1, W2, ..., Wn, correspondentes. A soma ponderada desses sinais, junto com um termo de viés bias (B) é tratado por uma função de ativação, que é responsável por convergir o dado em uma saída padronizada, é calculada pela expressão matemática:

$$Z = \sum_{i=1}^{n} WiXi + B$$

Equação 2.1: Neurônio Artificial Simples

Fonte: Adaptação de McCulloch & Pitts *apud* Minussi & Lotufo (2008)

Na equação 2.1, o valor Z é então passado por uma função de ativação φ(Z), que introduz não linearidade ao modelo e determina a saída do neurônio Y = φ(Z) . A função de ativação pode variar, sendo comuns a Sigmoide, ReLU ou tangente hiperbólica, dependendo do problema abordado. As RNAs são compostas por um conjunto de neurônios alinhados em fileiras e camadas como mostrado na Figura 2.2 e esse mecanismo simples, porém poderoso, permite às mesmas, aprendam a reconhecer padrões complexos e façam previsões precisas.

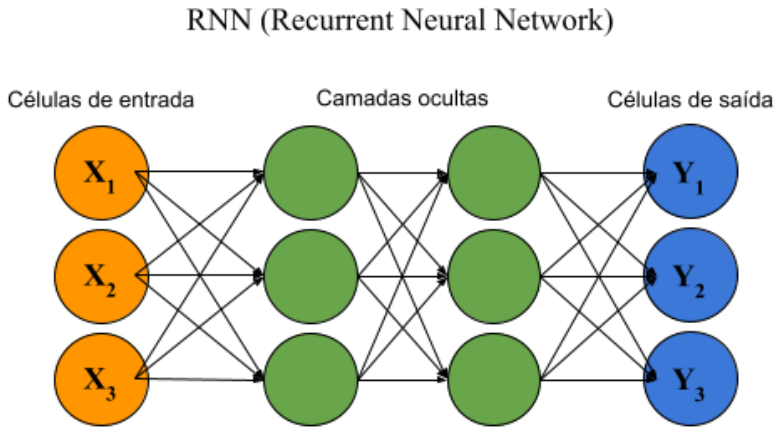

Figura 2.2: Estrutura de uma rede neural recorrente

Fonte: Adaptação de Minussi & Lotufo (2008)

Conforme elucidado por Minussi e Lotufo (2008), o processo de aprendizado em Redes Neurais Artificiais pode ser definido como um problema de otimização matemática, onde o objetivo é o ajuste iterativo dos pesos sinápticos (W) e viéses (B) das conexões neuronais.

O treinamento ocorre de forma supervisionada, a rede recebe os dados de entrada e seus respectivos resultados, buscando mapear a relação não linear entre eles.Para quantificar o desempenho durante esse processo, utiliza-se uma Função de Perda (*Loss Function*), como o erro quadrático médio (MSE) ou a Entropia Cruzada. Segundo Minussi e Lotufo (2008), essa função calcula a magnitude da divergência entre a predição feita pela rede e o valor esperado.

O objetivo do treinamento é minimizar esse valor de erro global. A atualização dos pesos é realizada através do algoritmo de Retropropagação (*Backpropagation*). De acordo com os autores, este algoritmo calcula o gradiente da função de perda em relação a cada peso da rede, determinando a "direção" e a "intensidade" com que cada conexão contribuiu para o erro final.

Com base nessas informações, um algoritmo otimizador ajusta os pesos no sentido oposto ao do gradiente, reduzindo progressivamente o erro a cada época até que o sistema atinja a convergência, permitindo generalizar o aprendizado para novos dados.

b. LSTM

Conforme demonstrado por Dias (2020), o aprendizado de máquina aplicado à predição de anomalias envolve a avaliação de diversos algoritmos. Modelos como Árvores de Decisão, Redes Neurais Artificiais (RNAs) e XGBoost são frequentemente aplicados, cada um apresentando suas particularidades.

Além dessas arquiteturas, Hochreiter e Schmidhuber (1997), destacam as redes neurais recorrentes, especificamente o modelo LSTM (Long Short-Term Memory), dada a sua reconhecida capacidade de modelar dependências temporais de longo prazo, como as encontradas em sinais de vibração e sua arquitetura resolve um dos maiores problemas das RNNs tradicionais: Vanishing Gradient.

Ainda na visão de Hochreiter e Schmidhuber (1997), o problema vanishing gradient é um desafio clássico em redes neurais artificiais durante o treinamento em um processo chamado retropropagação através do tempo (BPTT - Backpropagation through time), em que o gradiente, diminui exponencialmente ao ser propagado por muitas etapas temporais, chegando a valores quase nulos. Isso impede que a rede aprenda dependências de longo prazo, fazendo com que “esqueça” informações distantes na sequência temporal.

Hochreiter e Schmidhuber (1997), destacam a capacidade do LSTM resolver esse problema ao introduzir uma célula de memória que mantém informações ao longo do tempo com poucas alterações, e portas que regulam o fluxo de informações, permitindo ao modelo decidir o que guardar, esquecer ou usar, preservando o gradiente e facilitando o aprendizado de padrões

temporais longos e complexos, esse tipo de célula apresenta uma estrutura complexa como pode ser visto na Figura 2.3.

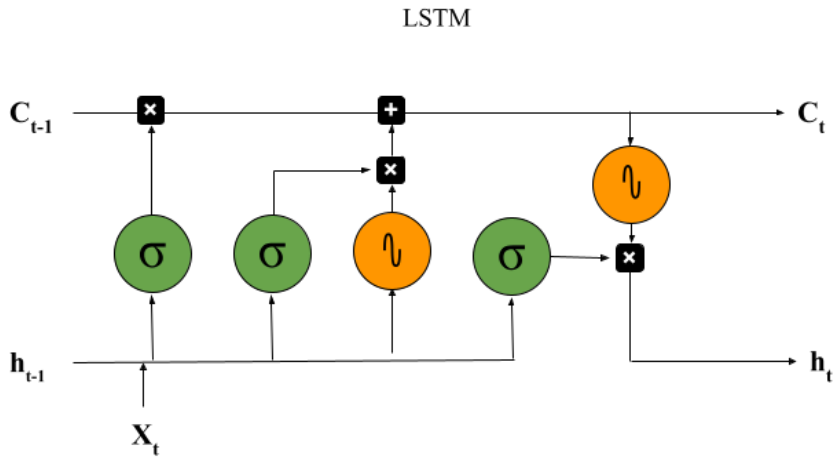

Figura 2.3: Estrutura de célula LSTM

Fonte: Adaptado de Mário Filho (2023)

Na Figura 2.3, é possível observar três entradas principais: o vetor de entrada atual X_t, o estado oculto anterior h_{t-1} e o estado de célula anterior C_{t-1}, que juntos alimentam as portas internas da célula.

Hochreiter e Schmidhuber (1997), destacam que as funções de ativação sigmoide (σ) são usadas nas portas de entrada, esquecimento e saída, pois elas limitam os valores entre 0 e 1, funcionando como filtros: 0 significa "bloquear totalmente" e 1 significa "deixar passar tudo". A função tangente hiperbólica (∿) é usada para transformar os dados que entram e saem da célula, permitindo que a informação flua com valores entre -1 e 1.

Hochreiter e Schmidhuber (1997), explicam também que a combinação dessas funções e portas permite que a célula LSTM controle o fluxo de informação de maneira mais precisa. Isso garante que apenas dados relevantes sejam mantidos ao longo do tempo, o que é essencial para lidar com o problema *vanishing gradient*.

Outra abordagem, dessa vez, sugerida por Dias (2020), diz que entre os diversos algoritmos de aprendizado de máquina aplicados à predição de anomalias , destaca-se o XGBoost (*Extreme Gradient Boosting*), conhecido por seu alto desempenho em classificação e regressão. O

modelo utiliza árvores de decisão fracas construídas de forma sequencial, onde cada nova árvore corrige os erros da anterior, resultando em uma predição mais robusta.

Para Dias (2020), o XGBoost ao contrário de métodos tradicionais que utilizam vetores de características para calcular a semelhança entre a entrada e o histórico de anomalias , utiliza o gradiente da função de perda para orientar a construção das árvores reforçadas. Dias (2020), afirma também que com isso, quanto mais uma característica é usada para tomar decisões relevantes dentro das árvores, maior será sua importância. Isso permite melhorar a precisão das predições, mas também identificar quais fatores têm maior influência.

Dias (2020), *apud* Carvalho et al. (2019), afirma que, para avaliar o desempenho de modelos de rede neural como o utilizado neste estudo, a Matriz de Confusão é uma ferramenta fundamental. Ela funciona como um painel de desempenho que compara as previsões realizadas pelo modelo com os valores reais conhecidos (dados de teste). Para Dias (2020), apud Ting (2017) e Nogare (2020), essa matriz quantifica os acertos e erros, categorizando-os em: Verdadeiros Positivos (VP), Verdadeiros Negativos (VN), Falsos Positivos (FP) e Falsos Negativos (FN), conforme ilustrado na Tabela 2.1:

		Classe predita	
		+	-
Classe Verdadeira	+	VP	FN
	-	FP	VN

Tabela 2.1: Matriz de confusão.

Fonte: Adaptado de Dias (2020) apud Carvalho et al. (2019)

Visualmente, a interpretação ideal da matriz ocorre quando os valores se concentram na diagonal principal (do canto superior esquerdo ao inferior direito). Isso indica que a classe predita pelo sistema corresponde exatamente à classe verdadeira da bancada experimental. Elementos fora dessa diagonal representam confusões do modelo.

Para Dias (2020), apud Rodrigues (2019), na Tabela 2.1, os "Verdadeiros" (VP e VN) representam os acertos, enquanto os "Falsos" (FP e FN) são os erros. Trazendo para o contexto prático deste trabalho de monitoramento de vibração:

- **Verdadeiro Positivo (VP):** O sistema prevê corretamente uma anomalia específica. Exemplo: A bancada está com um desbalanceamento ("Massa 1") e a rede neural classifica corretamente como "Massa 1".
- **Falso Positivo (FP - Alarme Falso):** O sistema detecta uma anomalia que não existe. Exemplo: A bancada está operando em condição "Normal", mas o modelo acusa incorretamente um "Motor Livre". Isso geraria paradas desnecessárias para manutenção.
- **Falso Negativo (FN - Falha não Detectada):** O sistema indica que a máquina está saudável quando, na verdade, existe um problema. Exemplo: Existe um "Engaste Solto" real, mas o modelo classifica como "Normal". Este é o erro mais crítico, pois permite que a máquina opere com defeito até uma quebra catastrófica.

A partir dos valores extraídos da matriz de confusão, calcula-se a Acurácia, que é a métrica global de desempenho do modelo, definida pela Equação (1):

$$Acurácia = \frac{VP + VN}{VP + VN + FP + FN}$$

Equação 2.2: Equação de Acurácia.

A acurácia representa a porcentagem total de acertos do modelo em relação ao total de amostras analisadas. No entanto, para o treinamento da rede neural (o processo de aprendizado em si), a métrica utilizada para ajustar os pesos matemáticos não é a acurácia, mas sim a função

de Perda ou Loss (especificamente a Categorical Crossentropy para problemas multiclasse), expressa na Equação :

$$Loss = \sum_{c=1}^{M} y_{o,c} \cdot log(p_{o,c})$$

Equação 2.3: Equação de Loss.

Onde:

- **M** é o número de classes (no caso deste estudo, as 5 categorias de anomalia);
- **y** é o indicador binário (0 ou 1) se a classe c é a classificação correta para a observação o;
- **p** é a probabilidade predita de que a observação o pertença à classe c.

Enquanto a acurácia informa "quantas vezes o modelo acertou", a função Loss quantifica "o quão longe" a previsão do modelo estava da realidade. O objetivo do treinamento é minimizar o valor do Loss. Um Loss próximo de zero indica que o modelo não apenas acertou a classe, mas o fez com alta confiança (probabilidade próxima de 100%), enquanto um Loss alto indica erros grosseiros ou baixa confiança nas predições.

2.4 Vibração

Rao (2008), afirma que a vibração mecânica pode ser definida como o movimento oscilatório de um corpo ao redor de uma posição de equilíbrio. Esse comportamento pode ser modelado, analisado e interpretado para identificar anomalias no funcionamento de sistemas mecânicos.

Já segundo Inman (2014), a vibração é caracterizada por movimentos repetitivos de partículas ou estruturas, geralmente provocados por desequilíbrios, forças excitadoras ou condições de operação. Inman (2014) também diz que a forma como um sistema responde a essas excitações depende de suas propriedades mecânicas, como massa, rigidez e amortecimento. O estudo das vibrações permite prever anomalias , reduzir danos e melhorar o desempenho de sistemas mecânicos

Para Meirovitch (2001), vibração pode ser entendida como qualquer oscilação mecânica em um sistema físico, podendo ocorrer em componentes estruturais, máquinas ou equipamentos. Para Meirovitch (2001), esse movimento pode ser livre ou forçado, e sua análise envolve variáveis como frequência, amplitude e modos vibracionais. Compreender essas oscilações é essencial para prever comportamentos dinâmicos e evitar anomalias estruturais ou funcionais.

Para Randall (2011), a aquisição e avaliação de dados são etapas críticas no diagnóstico de anomalias por vibração, pois determinam a qualidade das informações usadas em algoritmos de manutenção preditiva. Proakis e Manolakis (2007) afirmam que um sinal mal adquirido compromete qualquer análise posterior, independentemente da técnica utilizada, por isso, o sistema deve ser projetado com critérios técnicos rigorosos.

Inman (2014) e Rao (2008), mostram como o processo começa com a conversão de grandezas mecânicas como aceleração ou deslocamento em sinais elétricos, por meio de sensores de vibração. Após a captura, o sinal é amplificado, filtrado e convertido para formato digital. Como afirmado por Randall (2011), a taxa de amostragem deve respeitar o teorema de Nyquist, garantindo fidelidade do sinal, sendo assim a *Nyquist frequency* deve ser igual a metade da frequência de amostragem.

Após a digitalização, os dados podem ser avaliados por diferentes técnicas. Entre elas, a Wavelet Transform citada em Gu *et al*. (2021) e Branco *et al*. (2022):

Para Gu *et al*. (2021) e Branco *et al*. (2022) a transformada wavelet é uma técnica de decomposição multiescala que permite analisar simultaneamente o conteúdo de tempo e frequência de um sinal. Ao contrário da transformada de Fourier, que representa apenas a frequência global do sinal, a wavelet consegue identificar eventos transitórios e localizados no tempo, sendo especialmente útil na detecção de anomalias mecânicas incipientes.

Segundo Gu *et al*. (2021), a aplicação da transformada discreta de wavelet (DWT) em sinais de vibração permite extrair componentes em diferentes escalas de frequência, facilitando a identificação de padrões associados a anomalias em rolamentos. De forma semelhante, Branco *et al*. (2022) demonstraram que a decomposição multiescala via wavelet destaca sinais sutis de

degradação em sistemas elétricos, melhorando o desempenho de redes LSTM na previsão de anomalias .

Branco *et al*. (2022) afirmam que esses resultados confirmam a eficácia da wavelet como etapa de pré-processamento para realce de características relevantes em sinais ruidosos e não estacionários. Com a coleta adequada dos dados e a aplicação correta das técnicas de avaliação, é possível obter diagnósticos precisos mesmo em ambientes industriais severos. Ferramentas como *MATLAB*, *LabVIEW* e bibliotecas Python (como *SciPy* e *PyWavelets*) viabilizam a aplicação prática dessas abordagens na manutenção preditiva baseada em dados.

Randall (2011) mostra que a análise de vibrações é uma técnica amplamente empregada na manutenção preditiva de máquinas rotativas, uma vez que permite a identificação precoce de anomalias como desequilíbrios, desalinhamentos, folgas estruturais e anomalias em rolamentos, para essa finalidade, sensores do tipo acelerômetro piezoelétrico e piezoresistivo são tradicionalmente utilizados, dada sua capacidade de mensurar deslocamento, velocidade e aceleração com elevada precisão.

Para Rao (2008) os sensores de vibração são dispositivos utilizados para medir oscilações mecânicas em equipamentos e estruturas, sendo amplamente empregados em aplicações industriais para monitoramento de máquinas e análise de integridade estrutural. A vibração é uma variável crítica em sistemas mecânicos, e sua medição precisa é essencial para a coleta de dados confiáveis para análise subsequente.

Conforme apresentado pelo Rao (2008) Sensores de vibração podem ser classificados tanto pela grandeza física que mede, quanto pelo princípio construtivo de funcionamento, os principais tipos apresentados por ele são:

Grandeza medida mostradas pelo Rao (2008):

Transdutores de deslocamento: monitoram variações físicas de posição e são usados principalmente para vibrações de baixa frequência, especialmente em mancais e eixos.

Transdutores de velocidade: medem a velocidade relativa da vibração, com boa resposta em frequências médias.

Acelerômetros: capturam a aceleração vibracional, sendo ideais para uma ampla faixa de frequência e os mais comuns na indústria.
Sensores de distorção de fase: aplicados em análises específicas como ressonância e modos estruturais.

Tipo de construção segundo o Rao (2008):

Transdutores de resistência variável: baseiam-se na variação da resistência elétrica conforme o deslocamento mecânico e são usados principalmente para medições de deslocamento.

Transdutores piezoelétricos: utilizam materiais como quartzo ou cerâmicas para converter aceleração em carga elétrica; são robustos, confiáveis e muito empregados como acelerômetros.

Transdutores eletrodinâmicos: funcionam com bobinas e campos magnéticos, gerando tensão proporcional à velocidade, usados em sensores de velocidade vibracional.
Transdutor transformador diferencial linear variável (LVDT): do tipo indutivo, utilizado para medir deslocamento com alta precisão, indicado para vibrações de baixa frequência e aplicações em laboratório ou equipamentos de precisão.

Randall (2011), diz que a escolha do tipo de sensor depende de fatores como o tipo de máquina, a frequência esperada da vibração, as condições ambientais e o objetivo da medição. Rao (2008) também afirma que para medições industriais, acelerômetros piezoelétricos são preferidos devido à sua confiabilidade em ambientes hostis, resistência a altas temperaturas e capacidade de operar continuamente com pouco desgaste.

Além disso, Randall (2011) diz que o posicionamento adequado do sensor e a qualidade da montagem influenciam diretamente a fidelidade dos dados adquiridos, tornando essencial o conhecimento técnico na instalação e calibração dos sensores.

2.3 Análise Comparativa de Trabalhos Correlatos

Dessa forma, a literatura apresenta diversas abordagens para o diagnóstico de anomalias, variando desde métodos estatísticos clássicos até arquiteturas complexas de Deep Learning. A

fim de situar a presente pesquisa no contexto do estado da arte, a Tabela 2.2 apresenta um comparativo resumido entre os principais estudos correlatos analisados, destacando as técnicas empregadas e as características de cada abordagem.

Tabela 2.2: Comparativo entre abordagens de IA para diagnóstico de anomalias

Estudo /Autor	Técnica Principal	Aplicação Focada	Vantagens e Limitações Observadas
Liu et al. (2018)	SVM (Support Vector Machine)	Rolamentos e Engrenagens	Vantagem: Baixo custo computacional. Limitação: Depende de feature engineering manual e expertise humana prévia.
Zhang et al. (2017)	CNN (Conv. Neural Network)	Rolamentos em ruído severo	Vantagem: Alta precisão na extração de padrões visuais. Limitação: Alto custo computacional para gerar imagens e treinar a rede.
Lei et al. (2018)	DBN (Deep Belief Network)	Falhas combinadas em máquinas	Vantagem: Bom desempenho em dados estacionários. Limitação: Perde informações temporais transitórias cruciais para anomalias incipientes.
Este Trabalho (2026)	LSTM Otimizada	Desbalanceamento e Falhas Estruturais	Proposta: Elimina pré-processamento complexo (FFT/Imagens), foca em baixo custo e alta assertividade para anomalias mecânicas específicas.

Fonte: Pesquisa direta (2025)

A análise comparativa permite observar que, enquanto trabalhos como o de Zhang et al. (2017) priorizam a transformação do sinal em imagens para uso de CNNs exigindo maior poder de processamento e abordagens como a de Liu (2018) dependem da extração manual de características, a proposta deste trabalho é uma análise direta e automatizada.

A utilização de redes LSTM aplicadas diretamente ao sinal bruto busca um equilíbrio ótimo, elimina a necessidade de especialistas para selecionar características manuais e evita o custo computacional da conversão para imagens, viabilizando uma solução robusta para sistemas embarcados e monitoramento em tempo real.

3 METODOLOGIA

3.1 Tipo de pesquisa

Como descrito por Lakatos e Marconi (2003), a pesquisa quantitativa baseia-se na coleta e análise de dados numéricos, utilizando métodos estatísticos para testar hipóteses e produzir resultados generalizáveis. No presente trabalho, isso se reflete na aquisição de sinais de vibração da bancada experimental, no tratamento matemático desses dados e na posterior avaliação dos modelos de rede neural por meio de métricas como acurácia e precisão na classificação das diferentes configurações do sistema.

Para Lakatos e Marconi (2003), a pesquisa exploratória visa proporcionar maior familiaridade com um fenômeno, permitindo ao pesquisador desenvolver, esclarecer ou modificar conceitos e formular hipóteses para estudos posteriores. No contexto deste trabalho, o uso de redes neurais recorrentes como LSTM para interpretar padrões de vibração e classificar pequenas alterações estruturais e de parâmetros (como mudanças no engaste, adição de massas ou ADVs) é uma aplicação que carece de consolidação na literatura.

Segundo Gil apud Lakatos e Marconi (2017), o experimento é considerado o melhor exemplo de pesquisa científica. Para os autores, a pesquisa experimental consiste na determinação de um objeto de estudo, na seleção das variáveis capazes de influenciá-lo e na definição das normas de controle e de observação dos efeitos que essas variáveis produzem. Esta é a abordagem central do trabalho, em que uma bancada de testes controlada onde as variáveis do sistema são intencionalmente modificadas para gerar dados e treinar o modelo.

3.2 Materiais e Métodos

A Figura 3.1 representa o diagrama metodológico planejado para este trabalho. O objetivo da Figura é apresentar de forma clara a sequência de etapas que compõem o desenvolvimento da solução proposta. Cada bloco ilustra uma fase específica, evidenciando a transição entre os processos técnicos desde a definição do sistema experimental até a exibição final dos resultados ao usuário.

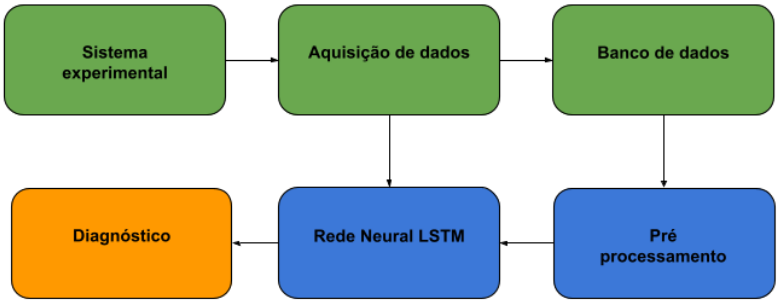

Figura 3.1: Metodologia

Fonte: Pesquisa direta (2025)

A Figura 3.1 mostra a metodologia que se inicia com um "Sistema experimental" que gera dados, capturados pela "Aquisição de dados". Esses dados seguem dois caminhos: são armazenados em um "Banco de dados" e também alimentam a "Rede Neural LSTM". Em paralelo, o "Banco de dados" passa por um "Pré processamento", onde são preparados e rotulados para treinar e validar a "Rede Neural LSTM". Após o treinamento, o modelo é colocado para analisar novos dados a fim de gerar um "Diagnóstico" de estado do sistema.

O processo metodológico inicia-se com a configuração da bancada experimental. Esta bancada consiste em um sistema mecânico controlado, como mostrado na Figura 3.2:

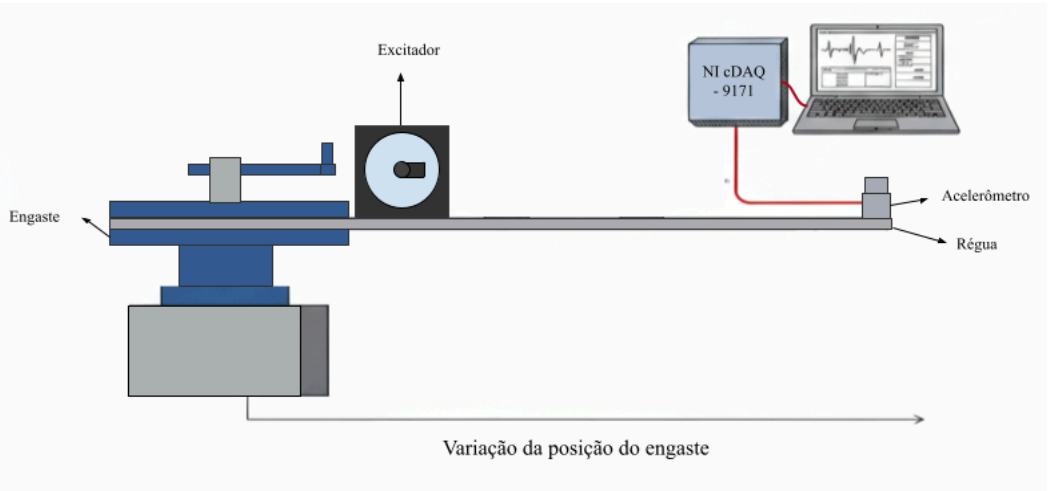

Figura 3.2: Esquema ilustrativo da bancada de testes mecânicos

Fonte: Pesquisa direta (2025)

Na Figura 3.2, uma viga (régua) engastada é excitada por um motor com massa desbalanceada e um acelerômetro posicionado estrategicamente na estrutura, assim a etapa seguinte é a aquisição de dados, em que a coleta de dados é dividida em classes.

Primeiramente, coleta-se o sinal de vibração do sistema em sua condição normal. Na sequência, modificações controladas são introduzidas, uma de cada vez, para simular diferentes estados operacionais ou estruturais. Essas modificações incluem a alteração do material do engaste, o deslocamento da viga no engaste, a adição de diferentes massas desbalanceadoras e a inclusão de Absorvedores Dinâmicos de Vibração (ADVs).

O sinal de vibração é adquirido e rotulado para cada uma das classes de configuração. Para o tratamento dos dados de vibração, optou-se pela utilização do sinal bruto no domínio do tempo como entrada direta para a rede neural. Esta abordagem foi selecionada por ser a mais alinhada com a arquitetura da rede LSTM (Long Short-Term Memory), que é projetada para identificar e aprender padrões complexos a partir de sequências temporais.

Ao fornecer o sinal bruto, permite-se que o próprio modelo, durante o processo de treinamento, extraia as características temporais mais relevantes para a distinção entre a operação normal e as diferentes configurações estruturais, sem a introdução de vieses ou a perda de informações que podem ocorrer durante transformações matemáticas. Essa escolha é respaldada por Afridi et al. (2023), que obtiveram alta precisão no diagnóstico de anomalias ao treinar uma rede LSTM diretamente com dados brutos de vibração.

Outras abordagens, como as Transformadas de Fourier (FFT) e Wavelet, foram consideradas, mas descartadas. Embora sejam ferramentas poderosas para análise no domínio da frequência para a análise de eventos transitórios, a evolução do sinal ao longo do tempo é uma fonte rica de informações que seria perdida. Portanto, a abordagem com dados brutos foi selecionada por preservar integralmente as informações do sinal original e sendo um método direto e eficaz para o modelo aprender a classificar diferentes estados do sistema a partir de dados temporais.

Para interpretar os dados e classificar as alterações no sistema, utiliza-se uma rede neural do tipo LSTM, implementada em *PyTorch*. Ela é treinada com os sinais rotulados (cada rótulo correspondendo a uma configuração experimental) e validados.

A interface gráfica, feita com *CustomTkinter*, é o elo entre o usuário e todo o sistema. Ela mostra os dados em tempo real, gráficos de vibração e o estado atual do sistema conforme identificado pela rede neural. A rede neural integrada detecta padrões nos dados de vibração e indica qual configuração está ativa (ex: "Sistema Normal", "Massa Adicional X", "Engaste Solto"). Com isso, o operador pode ter um diagnóstico preciso da condição estrutural do equipamento.

a. Equipamentos Utilizados

- Abaixo são apresentados os equipamentos empregados na realização deste projeto:

- **Bancada Experimental de Vibração:** Estrutura física composta por uma viga metálica engastada, atuando como um sistema massa-mola-amortecedor. A excitação do sistema é provida por um motor de corrente contínua (DC) com massa desbalanceada acoplada ao eixo.
- **Sensoriamento:** Utilizou-se um **acelerômetro piezoelétrico** industrial, escolhido pela sua alta sensibilidade e largura de banda adequada para captar harmônicas de vibração.
- **Sistema de Aquisição de Dados (DAQ):** Placa de aquisição **National Instruments cDAQ-9234**, responsável pela conversão analógico-digital de alta fidelidade (24-bits) e condicionamento de sinal (IEPE), conectada via USB.
- **Estação de Processamento:** Computador pessoal responsável pela interface com o DAQ, pré-processamento dos dados, treinamento da rede neural e execução da interface gráfica.

- Ambiente de Desenvolvimento e *Software*

O desenvolvimento do *software* foi realizado sobre o sistema operacional **Windows 10**, utilizando a IDE **Visual Studio Code**. A linguagem de programação adotada foi **Python 3.13**, suportada pelas seguintes bibliotecas e *frameworks*:

- **Inteligência Artificial: TensorFlow/Keras 2.10.0** (para construção, treinamento e inferência da rede neural LSTM).

- **Processamento de Sinais e Dados: NumPy 1.23.5** (cálculos matriciais), **Pandas 1.5.3** (manipulação de *datasets*) e **Scikit-learn 1.2.2** (métricas de avaliação e codificação de rótulos).
- **Visualização de Dados: Matplotlib 3.7.1** e **Seaborn 0.13.2** (para plotagem de gráficos de sinais e matriz de confusão).
- **Interface Gráfica (GUI): CustomTkinter 5.2.2** (para desenvolvimento de uma interface de usuário moderna e responsiva).

- **Comunicação: NIDAQmx 0.8.0** (Interface de comunicação com *hardware* da National Instruments)

- Configurações do *Hardware*:

O treinamento da rede neural e a execução do *software* de monitoramento foram realizados em uma estação de trabalho (Notebook) com as seguintes especificações:"

- **Processador (CPU):** 11th Gen Intel(R) Core(TM) i7-11800H @ 2.30GHz
- **Memória RAM:** 16 GB DDR4
- **Placa de Vídeo (GPU):** NVIDIA GFORCE GTX 1650
- **Aquisição de Dados:** Módulo National Instruments NI-9234 e Chassis cDAQ-9174.

b. Metodologia Experimental e Computacional

A metodologia adotada combinou a aquisição experimental de dados físicos em ambiente controlado com técnicas avançadas de aprendizado profundo (*Deep Learning*).

- Aquisição de dados

Para garantir a fidelidade na reconstrução digital do sinal analógico de vibração, a definição da taxa de amostragem foi fundamentada no Teorema da Amostragem de Nyquist-Shannon. Este teorema estabelece que a frequência de amostragem (fs) deve ser, no

mínimo, o dobro da maior frequência de interesse (fmax) presente no sinal para evitar o fenômeno de aliasing.

Os sinais de vibração foram coletados sob diferentes condições operacionais, com taxa de amostragem de 1000 Hz. Foi estruturado um banco de dados onde os arquivos brutos (formato .xlsx) foram categorizados e rotulados conforme a configuração experimental. As classes de estado definidas foram:

1. **NORMAL:** Condição base de operação.
2. **VARIAÇÃO DO ENGASTE:** Simulação de anomalia estrutural ou folga.
3. **MASSA 1 / MASSA 2:** Simulação de desbalanceamento ou alteração de carga.
4. **MOTOR LIVRE:** Simulação de anomalia na fixação da base do motor, caracterizando uma condição de folga mecânica estrutural (*Mechanical Looseness*) ou fixação comprometida.

- Pré-processamento dos Dados (Janelamento)

Para viabilizar o treinamento da rede neural e aumentar a robustez do modelo (*Data Augmentation*), aplicou-se a técnica de Janela Deslizante (*Sliding Window*). As séries temporais contínuas de cada arquivo foram segmentadas em janelas de **50 pontos** (50 milissegundos (ms)). Utilizou-se um passo (*stride*) de **10 pontos** (10 milissegundos (ms)), gerando sobreposição entre as janelas. Essa abordagem permite que a rede analise padrões temporais locais e multiplique a quantidade de exemplos disponíveis para treinamento.

- Rede Neural LSTM (Arquitetura Proposta)

A classificação dos estados operacionais foi realizada por uma Rede Neural Recorrente (RNN) utilizando as bibliotecas Keras/TensorFlow. A arquitetura final do modelo é ilustrada esquematicamente na **Figura 3.3** e detalhada nos tópicos a seguir:

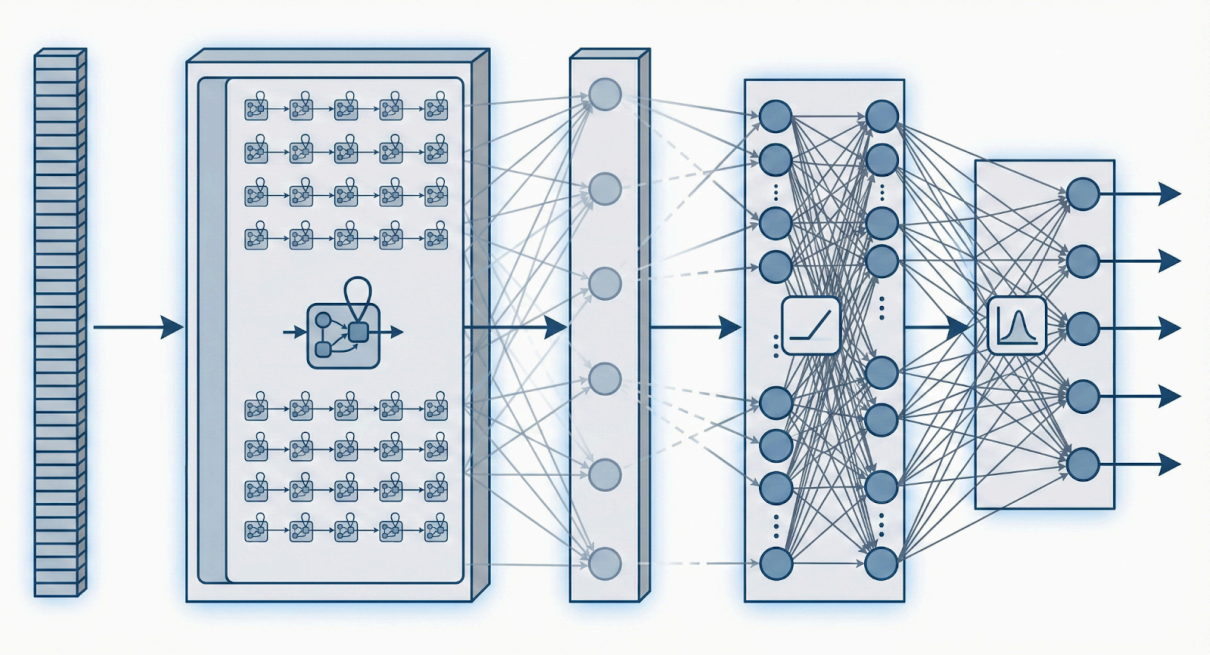

Figura 3.3: Diagrama da arquitetura da Rede Neural LSTM implementada.

Fonte: Pesquisa direta (2025)

Conforme ilustrado na imagem acima, a estrutura da rede é composta pela sequência de camadas descrita abaixo:

- **Camada de Entrada:** Recebe vetores de dimensão (50, 1), correspondentes às janelas temporais do sinal.
- **Camada LSTM:** Composta por 64 unidades de memória, responsável pela extração de características temporais e dependências sequenciais do sinal de vibração.
- **Camada de Dropout:** Aplicada uma taxa de 0.3 (30%), utilizada para prevenir o sobreajuste (*overfitting*) durante o treinamento, desativando aleatoriamente neurônios a cada ciclo.
- **Camada Densa (Oculta):** Composta por 32 neurônios com função de ativação ReLU (*Rectified Linear Unit*), introduzindo não-linearidade ao processamento.
- **Camada de Saída:** Camada densa final com função de ativação Softmax, contendo número de neurônios igual ao número de classes (5), fornecendo a probabilidade percentual de pertencimento a cada estado.

O processo de aprendizado supervisionado foi conduzido ao longo de 100 épocas, utilizando a função de perda *Categorical Crossentropy* para o monitoramento do erro de classificação multiclasse. Para a minimização desta função de custo e o ajuste iterativo dos pesos, selecionou-se o otimizador Adam (*Adaptive Moment Estimation*), proposto originalmente por Kingma e Ba (2014).

A escolha deste algoritmo, em detrimento do Gradiente Descendente Estocástico (SGD) clássico, deve-se à sua capacidade superior de ajustar taxas de aprendizado adaptativas para cada parâmetro individualmente. O Adam combina as vantagens de duas outras extensões de otimização, o AdaGrad, que lida bem com gradientes esparsos, e o RMSProp, eficiente em objetivos não estacionários.

Conforme explicam Kingma e Ba (2014), o algoritmo opera calculando médias móveis exponenciais do gradiente (estimativa de primeiro momento) e do quadrado do gradiente (estimativa de segundo momento). Matematicamente, essa abordagem permite que o modelo converja mais rapidamente em direção ao mínimo global da função de perda, superando obstáculos comuns como platôs na superfície de erro e o desvanecimento do gradiente, aspectos críticos para o treinamento eficaz de redes recorrentes profundas como a LSTM.

Por fim, a validação da capacidade de generalização do modelo foi assegurada através de uma estratégia rigorosa de separação de dados, um dos arquivos do conjunto foi isolado exclusivamente para a etapa de teste final, garantindo que a avaliação de desempenho ocorresse sobre dados nunca vistos durante o treinamento.

- Interface Homem-Máquina (IHM)

Para operacionalizar o sistema, foi desenvolvida uma Interface Gráfica (GUI) interativa que integra todo o *pipeline* de processamento. Suas funcionalidades incluem:

- **Visualização Avançada:** Plotagem simultânea do sinal bruto no tempo e da probabilidade de anomalia ao longo da amostra (resultado da convolução temporal da janela deslizante).
- **Diagnóstico Automático:** Apresentação do estado do sistema ("Normal" ou "Anomalia") inferido pela rede neural através de votação majoritária das janelas analisadas.

- **Visualização de Confiança:** Exibição da probabilidade estatística associada à classe predita.

3.3 Variáveis e indicadores

Segundo Lakatos e Marconi (2017), uma variável é um conceito operacional passível de mensuração, que pode representar propriedades, aspectos ou fatores de um objeto de estudo. Seus valores como quantidades, qualidades ou características, devem ser abrangentes e mutuamente exclusivos, variando conforme o caso analisado.

Como afirmado por, Lakatos e Marconi (2017), indicadores são os valores atribuídos a um conceito operacional para torná-lo mensurável como variável. Esses valores como quantidades, qualidades ou traços variam conforme o caso, sendo abrangentes e mutuamente exclusivos, permitindo a análise de objetos, processos ou fenômenos.

A seguir, na Tabela 3.1, apresentam-se as variáveis e seus respectivos indicadores utilizados nesta pesquisa.

Tabela 3.1: Tabela de variáveis e indicadores

Variáveis	Indicadores
Falhas Mecânicas	- Frequência de anomalias (MTBF); - Tempo médio de reparo (MTTR); - Gravidade das anomalias ; - Componentes críticos reincidentes; - Taxa de anomalias por tipo de equipamento; - Custo médio por anomalia.
Vibração	- Valor RMS da vibração; - Frequência dominante (FFT); - Pico de aceleração; - Fator de forma e curtose; - Comparativo com ISO 10816; - Histórico de tendência da vibração.
Rede neural	- Acurácia da predição; - Falsos positivos / negativos; - Erro médio (RMSE); - Tempo de resposta do modelo; - Confiabilidade no diagnóstico; - Volume de dados utilizados no treinamento.
Manutenção preditiva	- Redução de paradas não planejadas; - Tempo médio entre alertas reais; - Precisão na antecipação da anomalia; - Redução de custos com manutenção corretiva; - Tempo até anomalia previsto (RUL); - Eficiência do plano de ação preventiva.

Fonte: Pesquisa direta (2025)

3.4 Instrumento de coleta de dados

Para Lakatos e Marconi (2017), a coleta de dados consiste na aplicação dos instrumentos e técnicas planejadas para reunir as informações da pesquisa. Essa etapa exige paciência, perseverança e preparo do pesquisador, além de um registro cuidadoso. Um bom planejamento e organização são fundamentais para evitar desperdícios de tempo e facilitar o andamento do trabalho de campo. A seguir, na Tabela 3.2, estão catalogados os instrumentos e técnicas de coleta aplicados nesta pesquisa.

Tabela 3.2: Tabela de técnica de coletas

Técnica de Coleta	Vantagens	Desvantagens / Limitações
Sensoriamento de Vibração **(Acelerômetro Piezoelétrico Industrial)**	- Alta largura de banda: Capaz de captar altas frequências e harmônicas de anomalias . - Robustez: Construção adequada para ambientes industriais. - Alta Sensibilidade: Resposta precisa em tensão (mV/g).	- Necessidade de Condicionamento: Exige fonte de corrente constante (IEPE/ICP) para funcionar. - Sensibilidade a ruídos de cabo: Exige cabos blindados e bom aterramento para evitar interferência eletromagnética. - Acoplamento: A fixação inadequada atua como filtro mecânico (low-pass).
Aquisição de Dados (DAQ) **(NI cDAQ - 9171 e Labview)**	- Alta Fidelidade (24-bits): Faixa dinâmica de 102 dB, permitindo detectar micro-anomalias sem ruído de quantização. - Anti-Aliasing: Filtros integrados que garantem a integridade do sinal (Teorema de Nyquist). - IEPE Nativo: Alimenta o sensor sem circuitos externos. - Timing de *Hardware*: Amostragem precisa sem atrasos.	- Custo Elevado: *Hardware* e licenças de *software* proprietário. - Dependência de Drivers: Requer instalação do stack NI-DAQmx, dificultando a portabilidade para sistemas embarcados leves.
Rotulagem de Dados **(Registro Manual de Falhas)**	- Supervisão: Permite o uso de algoritmos supervisionados (como a LSTM proposta), que geralmente possuem maior acurácia. - Conhecimento Especialista: Incorpora a experiência do operador na definição das classes.	- Viés Humano: A classificação depende da percepção subjetiva do operador no momento da coleta. - Escalabilidade: Processo lento e custoso para grandes volumes de dados.

Fonte: Pesquisa direta (2025)

3.4.1. Padronização da Coleta e Monitoramento via *Software* Proprietário

Para eliminar divergências entre os dados utilizados no treinamento da rede neural e os dados lidos durante a operação real, foi desenvolvido um *software* unificado em linguagem Python. Este sistema integra a interface de comunicação com o *hardware* (driver NI-DAQmx) e o algoritmo de inteligência artificial.

A estratégia adotada consistiu em utilizar o mesmo processo de aquisição tanto para a geração do dataset de treinamento quanto para a inferência em tempo real. Isso assegura que a taxa de amostragem (1000 Hz), o condicionamento do sinal e o buffer de leitura sejam matematicamente idênticos nas duas etapas. Essa abordagem mitiga o risco de domain shift (desvio de domínio), onde o modelo performa bem em dados de arquivo, mas anomalia em tempo real devido a sutis diferenças na captura do sinal.

3.5 Tabulação dos dados

A organização e compilação dos dados coletados da bancada experimental foram realizadas utilizando uma abordagem estruturada. Foi estabelecido um banco de dados relacional para o armazenamento e organização dos sinais obtidos, permitindo um acesso automatizado e integrado.

Como principal instrumento de tabulação, foi utilizado o *software* Microsoft Excel. Os dados brutos de vibração de cada classe experimental foram armazenados em planilhas no formato .xlsx. Dentro dessas planilhas, os dados foram organizados em Tabelas e receberam os rótulos correspondentes, permitindo a categorização de acordo com a configuração experimental específica, como "NORMAL", "VARIAÇÃO DO ENGASTE", "MASSA 1", "MASSA 2" ou "MOTOR LIVRE"".

Para a compilação e manipulação automatizada desses dados tabulados, foram empregadas bibliotecas do ambiente Python, especificamente a biblioteca pandas, que faz parte do conjunto de ferramentas de processamento de dados do projeto.

3.6 Considerações Finais do capítulo

Neste capítulo, foram apresentadas as ferramentas e a metodologia utilizadas para a concretização desta pesquisa. Foi detalhado o tipo de pesquisa (quantitativa, exploratória e experimental), assim como os materiais e métodos empregados. A abordagem incluiu a definição da bancada experimental, a escolha da rede neural LSTM, a justificativa para o uso de dados brutos no domínio do tempo e a listagem dos equipamentos e *softwares* utilizados. Também foram definidas as variáveis e indicadores do projeto e os instrumentos de coleta.

Adicionalmente, estabeleceu-se a metodologia para a verificação de robustez do sistema, que consiste na injeção controlada de Ruído Branco Gaussiano Aditivo (AWGN) nos sinais de teste. Esta etapa foi incluída para simular as interferências típicas de ambientes industriais reais e validar a capacidade de generalização do modelo, mitigando o risco de sobreajuste (overfitting) em dados limpos de laboratório.

As técnicas e os instrumentos escolhidos estão de acordo com o objetivo proposto de desenvolver um sistema de diagnóstico de anomalias mecânicas. No capítulo seguinte, serão apresentados os resultados obtidos com a aplicação desta metodologia, detalhando a análise dos dados coletados, o treinamento do modelo e a avaliação de seu desempenho na classificação das diferentes configurações do sistema sob condições ideais e ruidosas.

4 RESULTADOS

Neste capítulo são apresentados os resultados obtidos a partir da aplicação da metodologia descrita no Capítulo 3. A análise abrange desde a caracterização dos sinais de vibração coletados da bancada experimental até a avaliação de desempenho do modelo de rede neural LSTM desenvolvido e a validação do sistema integrado de monitoramento em tempo real.

4.1. Concretização da Bancada Experimental

Para validar a implementação do projeto descrito na metodologia, apresenta-se inicialmente a concretização física da bancada experimental. A configuração adotada consiste em uma estrutura do tipo viga engastada-livre (*cantilever*). Como elemento estrutural, utilizou-se uma régua metálica (marca Kingtools) com comprimento total de 330 mm (considerando a extremidade não graduada além dos 300 mm úteis).

O sistema de engaste foi construído utilizando dois tarugos metálicos tencionados por parafusos, fixados em uma morsa de bancada para garantir rigidez. A Figura 4.1 evidencia a disposição física dos componentes, o acoplamento do motor e o posicionamento estratégico do acelerômetro.

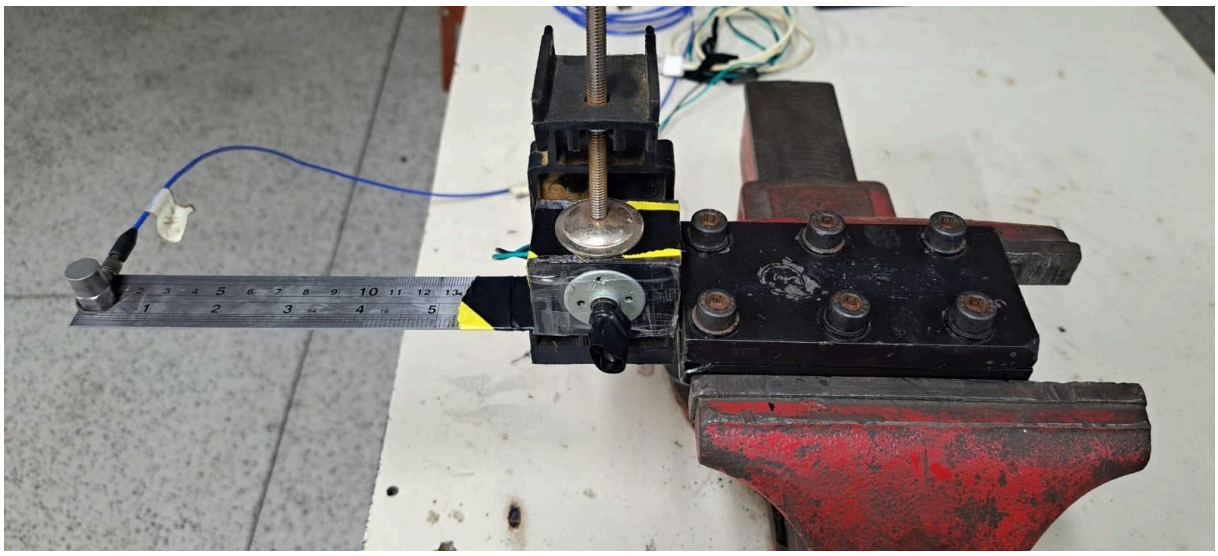

Figura 4.1: Bancada experimental, destacando o acoplamento do motor e o posicionamento do acelerômetro

Fonte: Pesquisa direta (2025)

A Figura 4.1, mostra que o sistema de engaste foi construído utilizando dois tarugos metálicos tencionados por parafusos, sendo o conjunto fixado em uma morsa de bancada para garantir máxima estabilidade e rigidez. A disposição física dos componentes e as dimensões críticas do sistema.

O ponto de fixação do engaste foi estabelecido a **212 mm** da extremidade livre, estando o restante do comprimento da régua imobilizado pelos tarugos. A excitação dinâmica é provida por um motor DC (9V), posicionado a **130 mm** da ponta em balanço. Já o acelerômetro piezoelétrico foi instalado ocupando o último centímetro (**10 mm**) da extremidade livre, estratégia que visa captar a região de maior amplitude vibratória (antinodo), maximizando a relação sinal-ruído e assegurando a fidelidade na leitura.

A aquisição dos dados requer uma interface robusta entre a instrumentação e o computador. A Figura 4.2 apresenta o módulo de aquisição (NI cDAQ) utilizado e sua conexão física com a estação de processamento, essencial para a digitalização dos sinais.

Figura 4.2: Conexão física entre o módulo de aquisição de dados (DAQ) e a estação de processamento.

Fonte: Pesquisa direta (2025)

O dispositivo da Figura 4.2 atua como a interface crítica entre o fenômeno físico e o ambiente digital. A conexão via cabo USB assegura a transmissão estável dos dados amostrados pelo acelerômetro para o *software* de análise, permitindo o processamento dos sinais em tempo real.

Por fim, na Figura 4.3, observa-se a integração final dos subsistemas mecânico, eletrônico e computacional, compondo a arquitetura completa de monitoramento proposta.

Figura 4.3: Visão global do sistema de monitoramento integrado em operação.

Fonte: Pesquisa direta (2025)

Na Figura 4.3, observa-se a integração final dos subsistemas mecânico, eletrônico e computacional, compondo a arquitetura completa de monitoramento proposta. Esta configuração valida a viabilidade prática do projeto, demonstrando a operabilidade conjunta do *hardware* de instrumentação.

- **Configuração das Condições de Falha**

Após o estabelecimento da condição basal ("Normal"), foram realizadas modificações controladas na bancada para simular diferentes tipos de anomalias mecânicas. As Figuras 4.4 a 4.7 ilustram essas alterações.

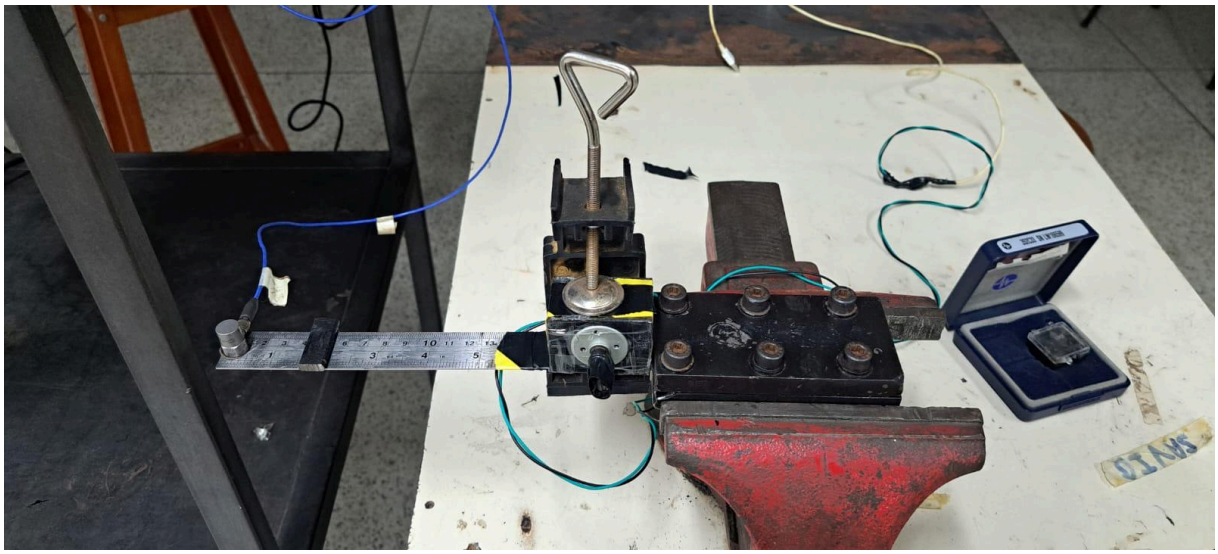

Figura 4.4: Configuração da anomalia "Massa Adicional 1", com peso extra fixado à viga.

Fonte: Pesquisa direta (2025)

Na Figura 4.4 é possível ver a metodologia usada para simular um desbalanceamento ou acúmulo de material, uma massa metálica de aproximadamente 7 gramas foi fixada firmemente à viga na posição de 40 mm a partir da extremidade livre. Esta adição altera localmente a distribuição de massa do sistema, impactando suas frequências naturais.

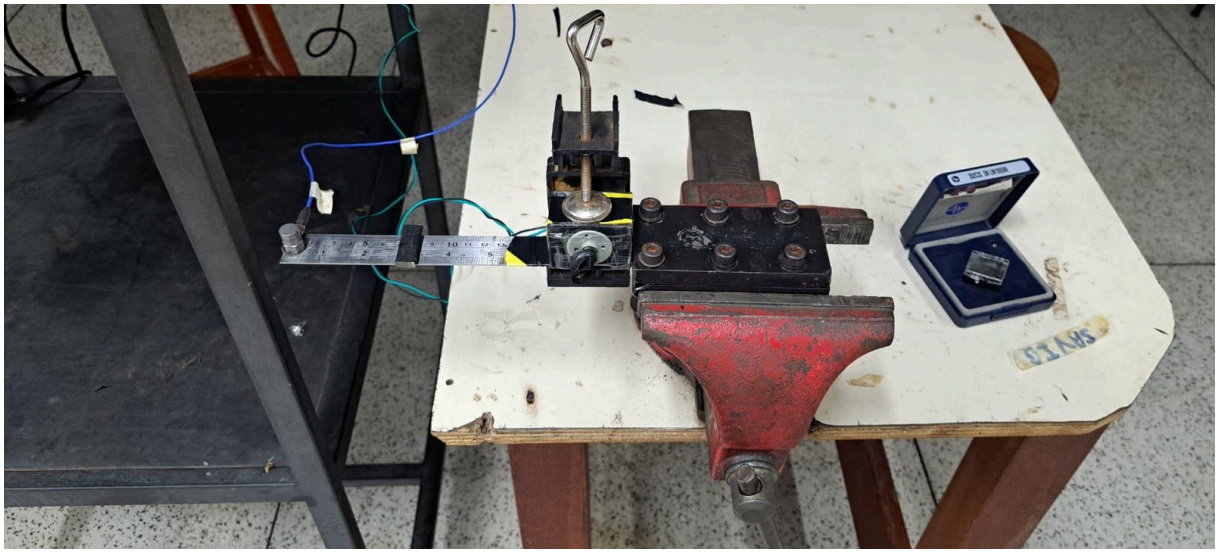

Figura 4.5: Configuração da anomalia "Massa Adicional 2", com o peso extra em posição distinta

Fonte: Pesquisa direta (2025)

Na Figura 4.5, a mesma massa adicional foi deslocada para a posição de 70 mm a partir da extremidade livre. O objetivo é avaliar a capacidade do sistema em distinguir não apenas a presença de massa extra, mas também a sua localização ao longo da estrutura.

Figura 4.6: Detalhe da condição "Motor Livre", simulando folga na fixação do elemento excitador.

Fonte: Pesquisa direta (2025)

Para a condição de "Motor Livre" como é mostrado na Figura 4.6, foi induzida através da remoção do sargento de fixação do motor DC à régua, que fica preso apenas por uma fita. Isso permite que o motor se mova ligeiramente durante a operação, gerando impactos intermitentes (choques mecânicos) e introduzindo não-linearidades no sinal de vibração.

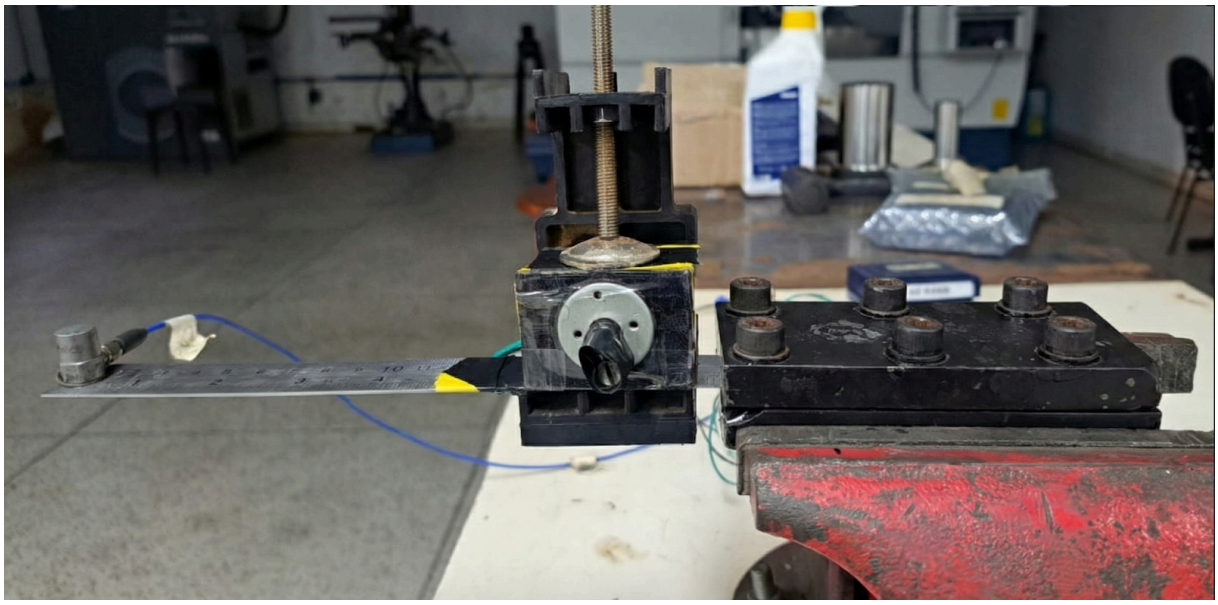

Figura 4.7: Configuração da anomalia "Deslocamento no Engaste", com alteração do comprimento útil da viga

Fonte: Pesquisa direta (2025)

A Figura 4.7 mostra a simulação de uma perda de rigidez ou um escorregamento na base de fixação. O conjunto régua, motor e acelerômetro foi movido em 5 mm para fora do engaste, passando a fixar a régua a 217 mm da ponta. Essa alteração modifica o comprimento efetivo do balanço, alterando significativamente a rigidez e a resposta dinâmica da estrutura.

A parametrização física destas condições de anomalia encerra a etapa de preparação experimental, assegurando a geração de um banco de dados diversificado e representativo. Com o ambiente físico validado e as classes de anomalia materializadas na bancada, procede-se à aquisição dos sinais vibratórios, etapa fundamental para capturar as assinaturas temporais que alimentarão o treinamento e a validação da rede neural proposta.

4.2. Coleta e Análise dos Sinais de Vibração

A fase seguinte consistiu na coleta e validação dos dados da bancada experimental. Nos ensaios preliminares, observou-se que a maior componente espectral relevante ocorria na condição de anomalia "Motor Livre", situando-se em aproximadamente 196 Hz. As demais condições (Normal, Desbalanceamento e Deslocamento) apresentaram frequências dominantes próximas à rotação síncrona do motor, em torno de 41 Hz.

Diante disso, definiu-se a taxa de amostragem de 1000 Hz (1 kS/s). Aplicando o critério de Nyquist, esta configuração define um limite de frequência analisável de:

$$Nyquist \quad = \frac{1000}{2} = 500 \quad = \quad fmax$$

Equação 3.1: Cálculo de frequência de aquisição segundo Nyquist

Fonte: RAO, Singiresu S. Vibrações Mecânicas (2009).

Esta largura de banda (0 a 500 Hz) cobre com ampla margem de segurança todo o espectro dinâmico do experimento. Ela é suficiente para capturar a fundamental (41 Hz) e suas harmônicas até a 12ª ordem, bem como a componente de alta frequência da anomalia de fixação (196 Hz), garantindo que não haja perda de informação espectral.

Além disso, no domínio do tempo, a taxa de 1000 Hz fornece uma resolução de aproximadamente 24 pontos por ciclo de rotação do motor (1000 / 41 ≈ 24,3), o que é adequado para que a rede neural LSTM identifique os padrões morfológicos da onda.

Conforme detalhado na metodologia, foram registrados sinais de vibração (aceleração) para a condição de referência ("Normal") e para quatro condições de anomalias induzidas. Para uma análise preliminar, os sinais foram visualizados no domínio do tempo. Como mostram as Figuras de 4.8 a 4.12.

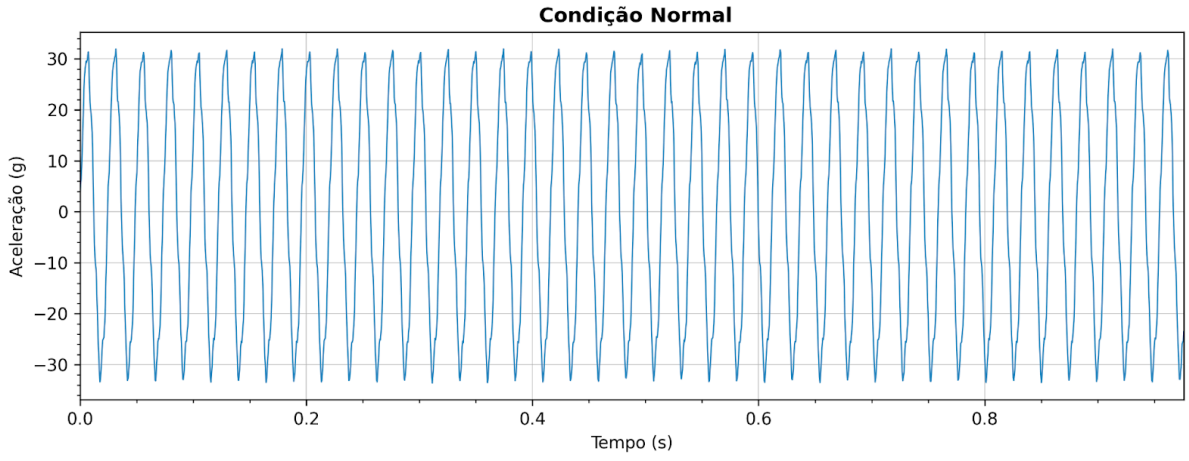

Figura 4.8: Gráfico do sinal de vibração no tempo (Aceleração) para a condição Normal.

Fonte: Pesquisa direta (2025)

A Figura 4.8 ilustra o sinal bruto para a condição de operação normal. O gráfico é caracterizado por uma amplitude estável e comportamento periódico, servindo como linha de base para a comparação com as anomalias.

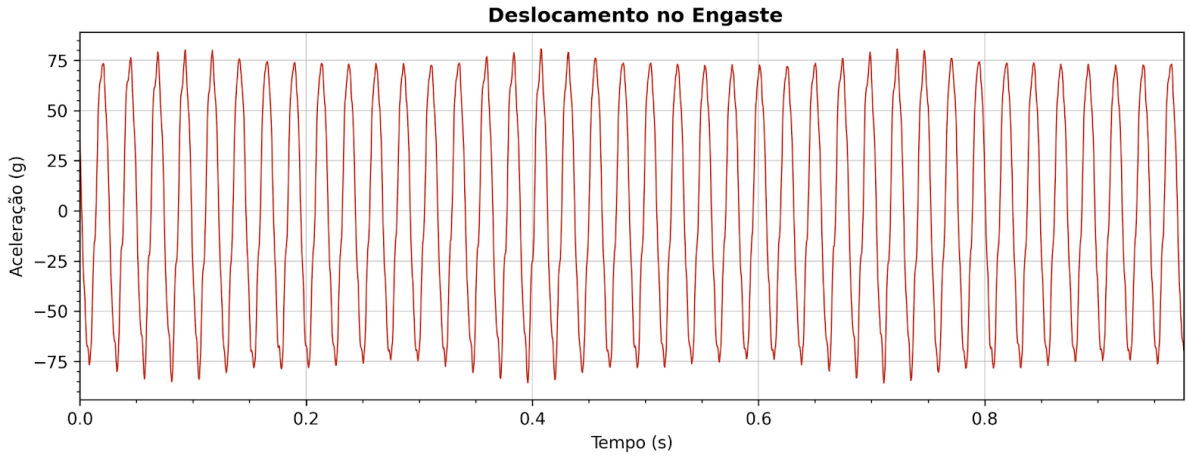

Figura 4.9: Gráfico do sinal de vibração no tempo (Aceleração) para a condição de Deslocamento no Engaste.

Fonte: Pesquisa direta (2025)

Em contraste, a Figura 4.9 apresenta o sinal sob a condição de "Deslocamento no Engaste". É possível observar visualmente o aumento significativo na amplitude da vibração, indicativo de maior energia cinética no sistema devido à alteração do braço de alavanca.

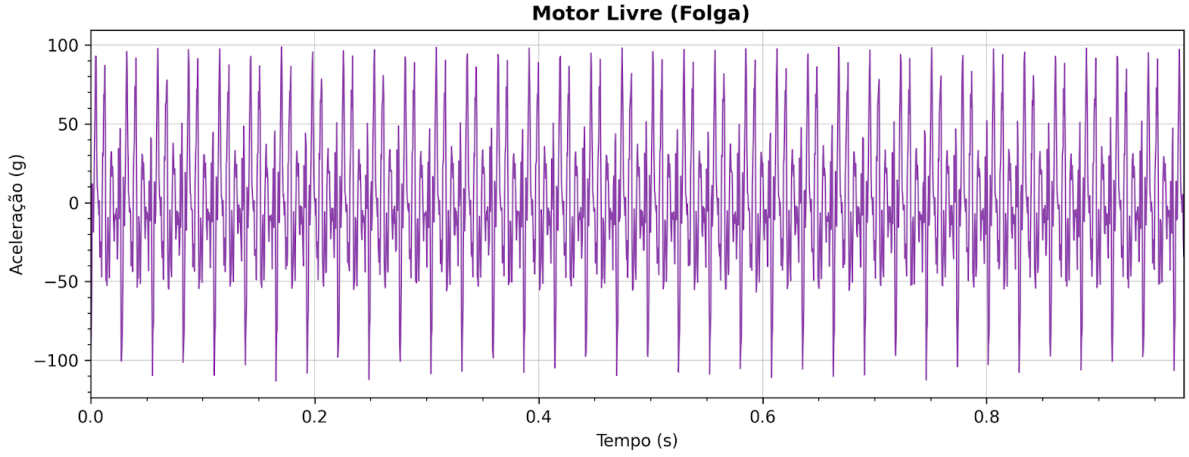

Figura 4.10: Gráfico do sinal de vibração no tempo para a condição "Motor Livre".

Fonte: Pesquisa direta (2025)

A condição de instabilidade de fixação é representada na Figura 4.10. O sinal reflete o comportamento caótico gerado pela folga, apresentando picos de aceleração abruptos que rompem o padrão harmônico observado nas condições anteriores.

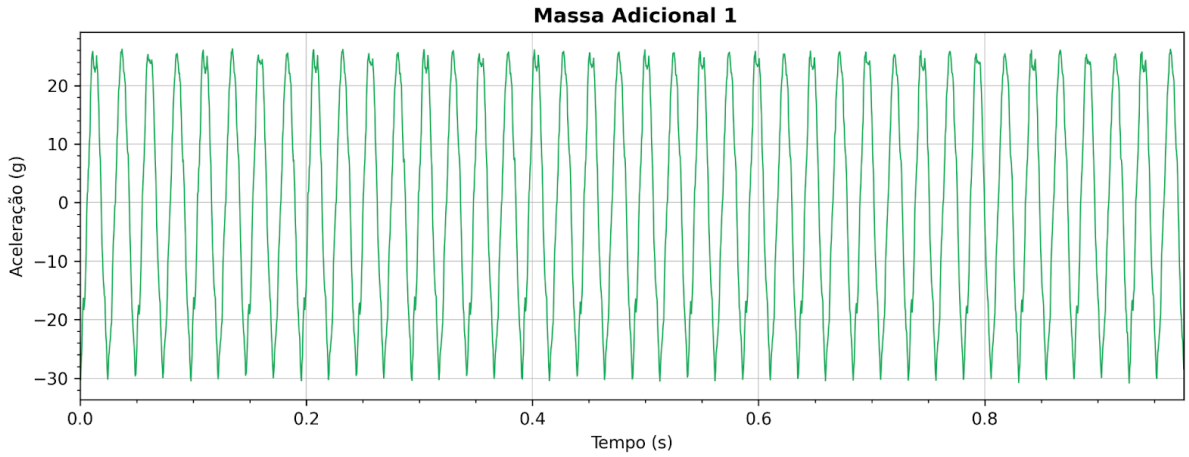

Figura 4.11: Gráfico do sinal de vibração no tempo para a condição "Massa Adicional 1".

Fonte: Pesquisa direta (2025)

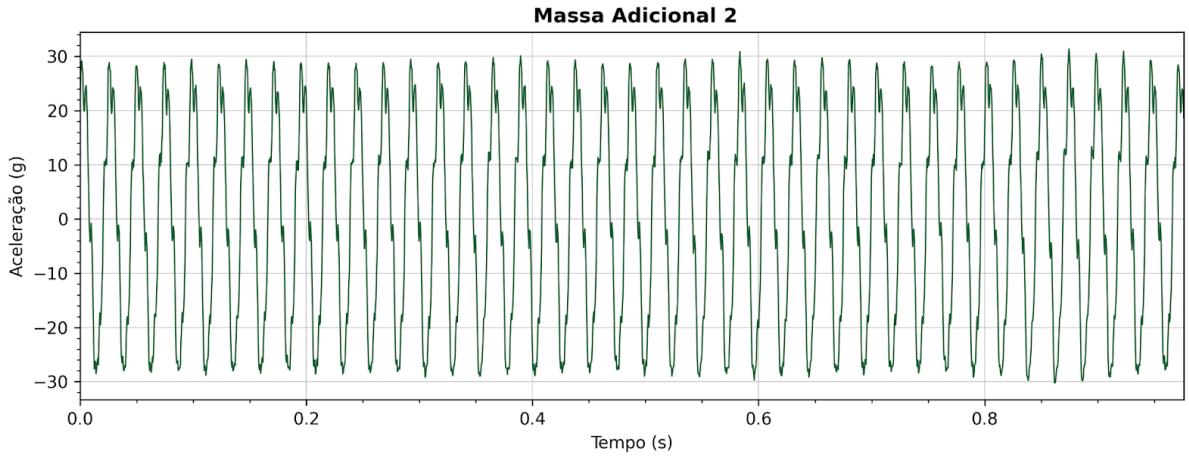

Figura 4.12: Gráfico do sinal de vibração no tempo para a condição "Massa Adicional 2".

Fonte: Pesquisa direta (2025)

Por fim, as Figuras 4.11 e 4.12 exibem os sinais para as condições de desbalanceamento por massa adicional. Visualmente, estas condições apresentam grande semelhança com a condição normal, dificultando uma distinção imediata apenas pela inspeção da forma de onda.

A fim de quantificar essas variações e diagnosticar a natureza física das anomalias , foram extraídos indicadores estatísticos e espectrais dos sinais brutos. A Tabela 4.1 sumariza os valores médios de RMS (energia), Pico (impacto) e Frequência Dominante para cada classe.

Tabela 4.1: Indicadores de vibração para as diferentes classes experimentais.

Condição Operacional (Classe)	Valor RMS da Vibração (g)	Pico de Aceleração (g)	Frequência Dominante (Hz)
Normal	22,42	35,03	41,11
Deslocamento no Engaste	57,32	91,41	41,16
Motor Livre	36,13	112,51	196,48
Massa 1	20,67	33,39	41,25
Massa 2	21,20	33,41	40,96

Fonte: Pesquisa direta (2025)

A análise dos indicadores da Tabela 4.1 revela comportamentos físicos distintos para cada anomalia, corroborando a eficácia da instrumentação utilizada:

- **Energia Vibratória (RMS):** Nota-se uma elevação substancial do valor RMS na condição "Deslocamento no Engaste” (57,32 g) em relação ao estado “Normal". Sendo o RMS uma métrica que pondera a energia global do sinal, esse aumento tende a refletir um maior nível de esforço dinâmico contínuo atuando sobre o sistema, sugerindo uma relação direta entre a severidade da vibração e a integridade estrutural do engaste.
- **Impactos e Folgas (Pico):** A condição "Motor Livre" destaca-se por apresentar o maior Pico de Aceleração (112,51 g), mesmo sem corresponder à maior energia média (RMS). O valor de pico interfere na análise ao evidenciar a presença de eventos transientes de alta amplitude, comportamento associado à ocorrência de choques mecânicos intermitentes ou instabilidades na fixação, os quais elevam o valor máximo registrado sem necessariamente alterar a média energética na mesma proporção.

- **Resposta em Frequência:** A predominância da frequência em torno de 41 Hz para a maioria das classes sugere uma vinculação com a rotação síncrona do motor. Em contrapartida, a alteração da frequência dominante para 196,48 Hz na condição "Motor Livre" aponta para a interferência de componentes harmônicas superiores ou a excitação de frequências naturais da estrutura, fatores que costumam emergir em cenários de instabilidade mecânica.
- **Limitações da Análise Estatística:** A proximidade dos valores observados nas condições de "Massa Adicional" em relação à condição "Normal" sugere que indicadores estatísticos globais, como RMS e Pico, podem apresentar sensibilidade reduzida diante de alterações de carga mais sutis ou desbalanceamentos leves, dificultando a distinção imediata entre esses estados apenas por meio dessas métricas.

Essa complexidade e a sobreposição de características estatísticas em anomalias incipientes justificam a abordagem metodológica proposta neste trabalho: a utilização do sinal bruto no domínio do tempo como entrada para a rede neural **LSTM**. Diferente da análise estatística simples, a LSTM é capaz de aprender padrões temporais sequenciais e complexos, permitindo a classificação correta mesmo quando os indicadores globais (como RMS) não apresentam variações significativas.

A superioridade da abordagem temporal torna-se ainda mais evidente ao confrontarmos a dificuldade de segregação espectral. Observou-se que as condições 'Normal' e 'Deslocamento no Engaste' apresentaram frequências fundamentais extremamente próximas, criando uma sobreposição que inviabilizaria diagnósticos baseados puramente na Transformada Rápida de Fourier (FFT).

Enquanto um analista ou algoritmo clássico teria dificuldade em distinguir esses dois estados olhando apenas para o pico de frequência, a rede LSTM obteve êxito na separação. Isso ocorre porque o modelo não se limita à amplitude espectral estática; ao processar a morfologia do sinal bruto, a rede neural é capaz de identificar as sutis não-linearidades e modulações temporais transitórias que diferenciam o engaste solto da operação normal, características essas que são invisíveis ou mascaradas na análise puramente frequencial.

4.3. Desempenho do Modelo Preditivo (Rede Neural LSTM)

Após a coleta e rotulagem dos dados, a rede neural LSTM, implementada utilizando a biblioteca TensorFlow/Keras, foi treinada para classificar as diferentes condições operacionais do sistema. A Figura 4.13 ilustra a evolução da acurácia e da função de perda (loss) ao longo das épocas de treinamento e validação, demonstrando a convergência do aprendizado e a capacidade de generalização do modelo frente a dados não vistos.

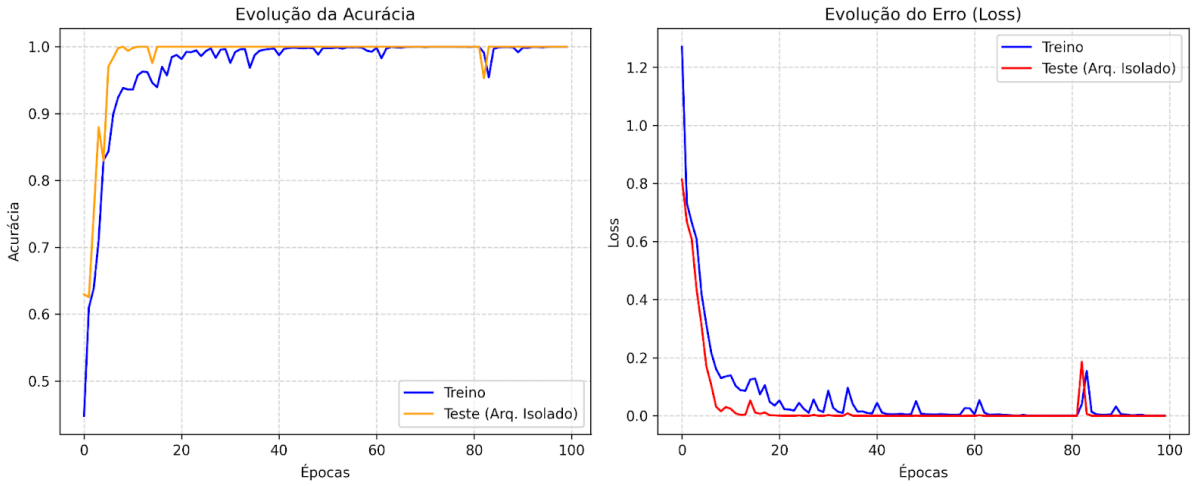

Figura 4.13: Gráfico da evolução da acurácia e da função de perda (Loss) durante as épocas de treinamento e validação do modelo LSTM.

Fonte: Pesquisa direta (2025)

Na Figura 4.13, a análise das curvas de aprendizado revela um comportamento estável e satisfatório da rede neural. Observa-se que a acurácia apresentou um crescimento acentuado nas primeiras épocas, estabilizando-se rapidamente em um patamar elevado, o que indica que as características extraídas das séries temporais de vibração possuem alta separabilidade linear no espaço latente da LSTM.

Simultaneamente, a função de perda (loss) exibiu um decaimento consistente, sem apresentar oscilações bruscas ou platôs prolongados, demonstrando a eficácia do otimizador Adam na minimização do erro. Um ponto crucial a ser destacado é o alinhamento próximo entre as curvas de treinamento e de validação.

Para uma avaliação rigorosa do desempenho final do modelo em dados de teste, foram calculadas as métricas de classificação, apresentadas na Tabela 4.2.

Tabela 4.2: Métricas de Desempenho da Rede Neural LSTM (em dados de teste)

Fonte: Pesquisa direta (2025)

Métrica de Avaliação	**Valor Obtido**
Acurácia Global	100.00%
Função de Perda (Test Loss)	0,0000

Os indicadores apresentados na Tabela 4.2 validam a hipótese inicial deste trabalho, de que as assinaturas temporais das anomalias mecânicas contêm informações suficientes para uma distinção precisa quando processadas por redes recorrentes. A convergência para uma acurácia máxima, aliada a um erro (*loss*) nulo, demonstra que a estratégia de pré-processamento por janela deslizante foi eficaz em capturar a dinâmica do sistema, permitindo que a rede segregasse as diferentes condições de anomalia sem ambiguidade.

Dada a natureza de classificação multiclasse deste projeto, a Matriz de Confusão (Figura 4.14) é a ferramenta mais importante para avaliar o desempenho. Ela detalha exatamente quais classes o modelo acertou e, mais importante, quais classes ele confundiu entre si.

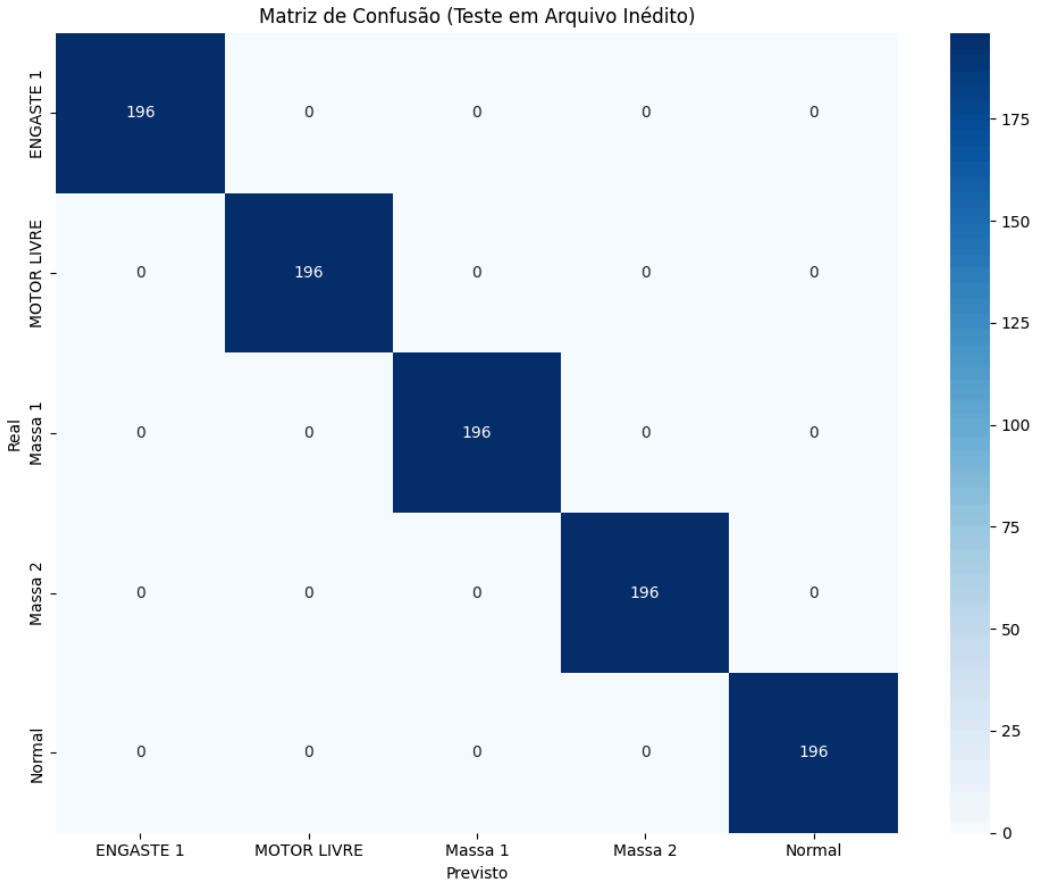

Figura 4.14: Matriz de Confusão do modelo LSTM

Fonte: Pesquisa direta (2025)

A Matriz de Confusão (Figura 4.14), permite uma avaliação detalhada do desempenho do classificador sobre o conjunto de teste (dados nunca vistos pela rede). No gráfico, o eixo vertical (Y) representa as classes reais (rótulos verdadeiros), enquanto o eixo horizontal (X) indica as classes preditas pelo modelo.

A predominância de valores na diagonal principal evidencia a alta taxa de acerto global do sistema. A ausência de falsos positivos significativos fora da diagonal confirma que a arquitetura LSTM foi capaz de extrair características temporais exclusivas para cada tipo de anomalia, validando a robustez do diagnóstico proposto.

Além disso, a matriz de confusão não apresentou erros de classificação cruzada. O modelo não apenas distinguiu perfeitamente entre "Normal" e "Anomalia", mas também diferenciou corretamente entre anomalias de natureza distinta, como "Massa 1" e "Massa 2". Isso comprova que a rede neural aprendeu as assinaturas temporais específicas de cada modo de anomalia, sem apresentar a confusão.

Cabe destacar que a eficácia máxima obtida, não foi alcançada na primeira iteração experimental. Inicialmente, foram testadas outras arquiteturas simplificadas, com apenas uma camada oculta e janelas de tempo reduzidas (20 pontos), as quais apresentaram dificuldades de convergência e tendência ao *underfitting*, falhando em capturar a complexidade dinâmica do sinal.

A arquitetura final, composta por duas camadas LSTM com janela de 50 pontos, foi o resultado de um processo iterativo de otimização, onde a complexidade do modelo foi ajustada progressivamente até que se atingisse a estabilidade na classificação das cinco condições de anomalia.

4.4. Análise de Robustez e Sensibilidade ao Ruído

Diante dos resultados de classificação ideais (100% de acurácia) obtidos com o conjunto de dados de teste original, fez-se necessário verificar a resiliência do modelo proposto. Em ambientes controlados de laboratório, a relação sinal-ruído (SNR) é tipicamente alta, o que favorece a classificação. No entanto, a aplicação prática em chão de fábrica, como em plantas siderúrgicas, expõe o sistema a interferências eletromagnéticas, vibrações de fundo de outros equipamentos e ruídos de instrumentação.

Para mitigar o risco de overfitting (sobreajuste) e validar a aplicabilidade industrial da solução, foi realizado um teste de estresse utilizando a métrica de Relação Sinal-Ruído (SNR). A metodologia consistiu em manter o modelo treinado com os sinais originais (limpos), mas injetar propositalmente Ruído Branco Gaussiano Aditivo (AWGN) nos dados de teste antes da inferência.

Diferentemente da abordagem linear, a avaliação via SNR mede a qualidade do sinal em escala logarítmica (decibéis), definida pela razão entre a potência do sinal útil (Psinal) e a potência do ruído injetado (Pruido), conforme a equação:

$$SNR_{dB} = 10 \cdot log10(\frac{P\,ruido}{P\,sinal})$$

Nesta escala, quanto maior o valor em dB, mais "limpo" e predominante é o sinal mecânico em relação à interferência. Valores próximos a 0 dB indicam que a potência do ruído se iguala à do sinal.

A Figura 4.20 apresenta a curva de robustez do modelo, relacionando o SNR (Eixo X) com a acurácia de classificação resultante (Eixo Y).

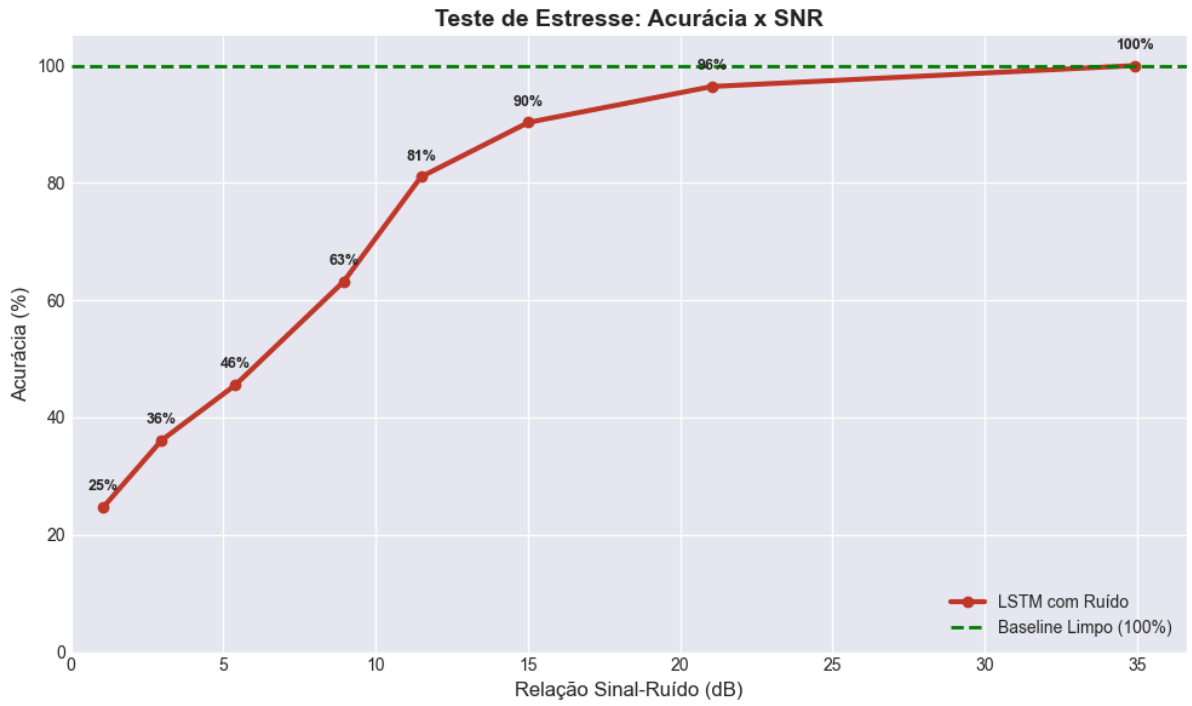

Figura 4.15: Curva de desempenho do modelo LSTM sob diferentes níveis de ruído artificial.

Fonte: Pesquisa direta (2025)

A análise da Figura 4.20 revela o comportamento da rede neural frente à degradação da qualidade do sinal. O desempenho do classificador pode ser segmentado em três zonas distintas de operação:

Zona de Alta Fidelidade (SNR > 15 dB): Observa-se que, para níveis de SNR superiores a 15 dB, a rede neural preservou uma acurácia superior a 90%. Este resultado (atingindo ~99% em 30 dB) indica que, quando o sinal mecânico é claramente predominante, o modelo LSTM consegue abstrair a morfologia sequencial da anomalia, não sendo influenciado por ruídos de fundo moderados típicos de sensores industriais bem instalados.

Zona de Decaimento (10 dB < SNR < 15 dB): Nesta faixa intermediária, nota-se um decaimento acentuado da performance. Com o SNR caindo de 20 dB para 10 dB, a precisão global é reduzida significativamente (caindo para a faixa de 60-70%). O sistema ainda mantém capacidade de diferenciar anomalias com assinaturas energéticas muito distintas (como "Motor Livre" vs. "Normal"), mas perde a sensibilidade para distinções sutis entre as massas de desbalanceamento, cujas diferenças morfológicas começam a ser mascaradas pela energia do ruído.

Zona de Ruptura (SNR < 5 dB): Quando a relação sinal-ruído cai abaixo de 10 dB, a capacidade de classificação é severamente comprometida. Abaixo de 5 dB, a acurácia converge para aproximadamente 40%, o que estatisticamente equivale ao "acaso" para um problema de 5 classes. Neste ponto, a amplitude da interferência torna-se comparável ou superior às variações que caracterizam as anomalias , impossibilitando a extração de padrões confiáveis pela rede.

Este teste de robustez valida a hipótese de que a arquitetura baseada em *Deep Learning* possui uma resiliência intrínseca para operações com SNR alto e moderado (> 15 dB). Contudo, os resultados delimitam a fronteira operacional do sistema, indicando que em ambientes com poluição eletromagnética severa (onde o SNR possa cair abaixo de 10 dB), torna-se indispensável o uso de blindagem de cabos e filtros de sinal (analógicos ou digitais) anteriores à etapa de inferência da rede neural.

4.4.1. Matriz de Confusão de robustez

Para compreender a dinâmica de classificação sob interferência moderada, selecionou-se o ponto de operação com SNR de 15 dB para uma análise detalhada. Neste cenário, o modelo apresentou uma Acurácia Global de 90%. Este resultado valida a robustez do sistema de forma

realista, um índice de acerto absoluto (100%) sob injeção de ruído significativo seria, um forte indicativo de *overfitting* (ajuste excessivo).

A leve queda de desempenho reflete, portanto, a fronteira física natural de separabilidade entre as classes mais sutis, e não uma deficiência do algoritmo. A distribuição detalhada dos acertos e erros pode ser visualizada na Matriz de Confusão apresentada na Figura 4.21:

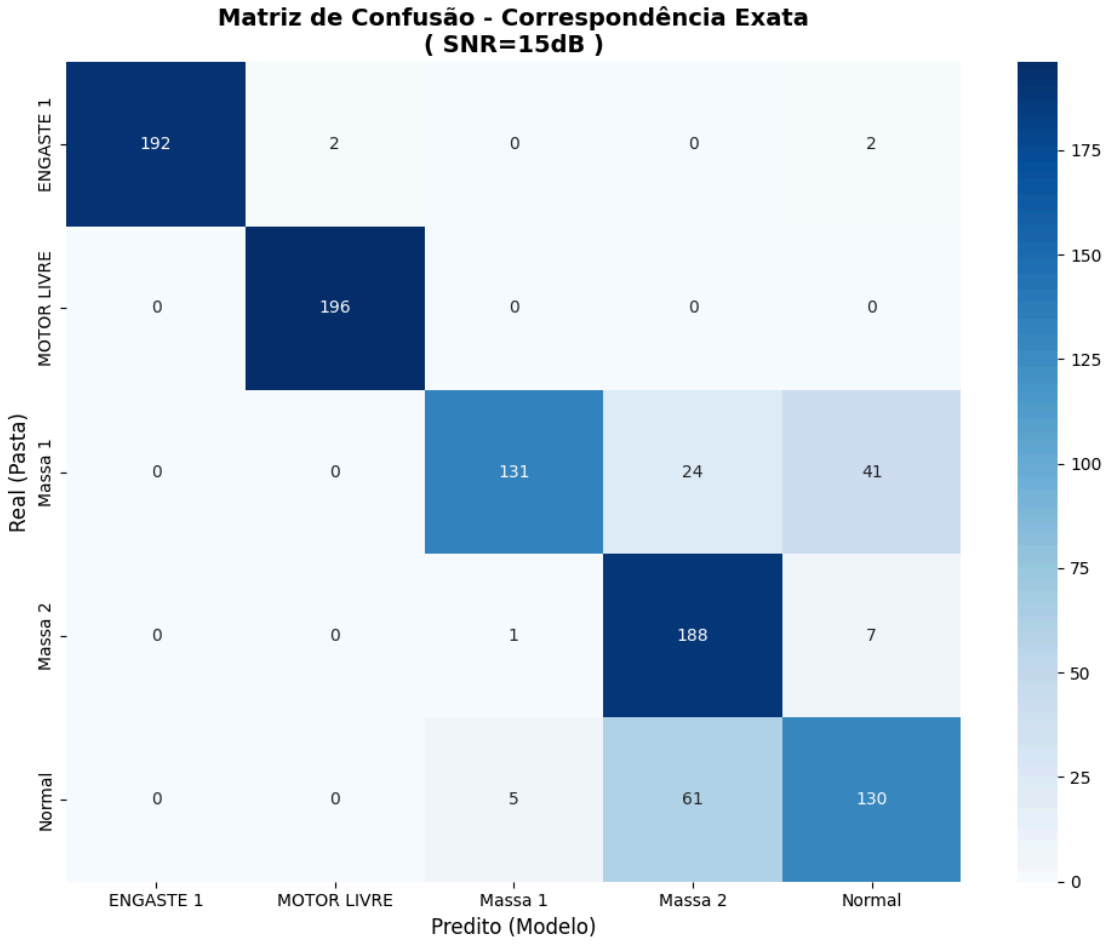

Figura 4.16: Matriz de Confusão para o cenário de teste com SNR de 15 dB.

Fonte: Pesquisa direta (2025)

A análise das classes individuais (vide relatório detalhado) revela comportamentos distintos frente ao ruído:

Imunidade a Ruído: As classes "ENGASTE 1" e "MOTOR LIVRE" mantiveram quase 100% de acerto. Isso demonstra que as assinaturas vibratórias destas anomalias , de leves

mudanças no engaste e a soltura mecânica do motor livre, possuem características energéticas e morfológicas tão distintas que, mesmo sob ruído de 15 dB, não se confundem com nenhuma outra condição.

O principal foco de degradação do sistema concentrou-se nas classes de desbalanceamento leve e moderado ('Massa 1' e 'Massa 2'), revelando uma sobreposição de fronteiras de decisão causada pelo ruído:

A classe 'Massa 1' apresentou um *Recall* de apenas 0,67, indicando que 33% das amostras reais não foram detectadas. Fisicamente, como a 'Massa 1' gera a menor amplitude vibratória, o ruído de 15 dB elevou o patamar de fundo a um nível que alterou drasticamente essa assinatura sutil, fazendo com que o modelo a confunda com a condição 'Normal'.

Diferente do cenário sem ruído, a 'Massa 2' sofreu uma queda abrupta em sua Precisão para 0,69, embora tenha mantido um *Recall* alto (0,96). Isso indica um alto índice de Falsos Positivos: o modelo consegue identificar a 'Massa 2', mas também classifica incorretamente ruídos da classe 'Normal' e da 'Massa 1' como sendo 'Massa 2'.

O ruído de 15 dB criou uma 'zona cinzenta' entre as classes de baixa energia (Normal, Massa 1, Massa 2). Enquanto as anomalias graves (Engaste, Motor) permanecem distintas, as anomalias leves perderam sua separabilidade estatística, resultando na queda da acurácia global para 85,41%.

4.5. Validação do Sistema de Monitoramento Integrado

A etapa final consistiu na integração do modelo LSTM treinado à interface gráfica (GUI) desenvolvida em *CustomTkinter*, validando o sistema completo em condições reais de operação. O *software* recebe dados até então desconhecidos, processa o sinal e utiliza a rede neural para inferir o estado atual da estrutura, exibindo o diagnóstico instantâneo ao operador. Para comprovar a eficácia da ferramenta, foram realizados ensaios sequenciais simulando todas as condições de anomalia estudadas. Como mostrado nas Figuras de 4.15 a 4.19.

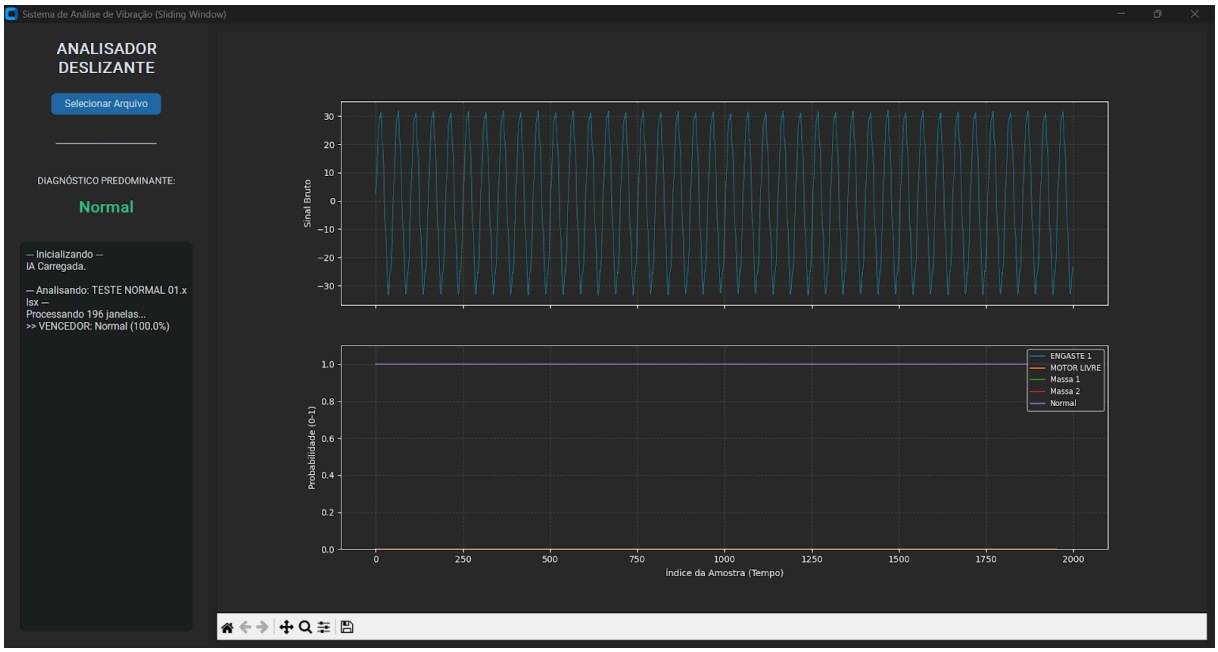

Figura 4.17: Interface gráfica em operação monitorando a condição "Normal".

Fonte: Pesquisa direta (2025)

A Figura 4.17 apresenta o monitoramento da condição basal. O sistema identificou corretamente o padrão estável, exibindo o status "Diagnóstico: Normal" e uma probabilidade de confiança superior a 100%.

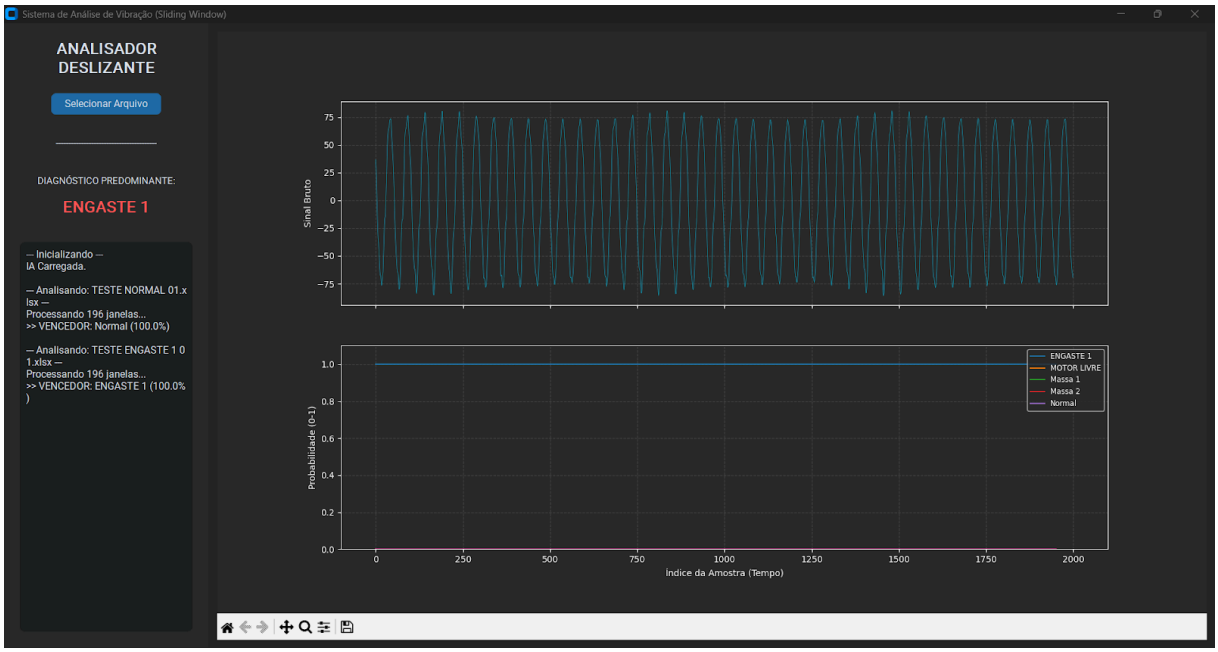

Figura 4.18: Validação da detecção da anomalia "Deslocamento no Engaste" pela interface.

Fonte: Pesquisa direta (2025)

Na sequência, induziu-se a anomalia estrutural crítica. A Figura 4.18 captura o momento em que o engaste foi deslocado. O sistema reagiu imediatamente à elevação da energia vibratória, alterando o diagnóstico para "Deslocamento no Engaste" e alertando visualmente o operador sobre a severidade da condição.

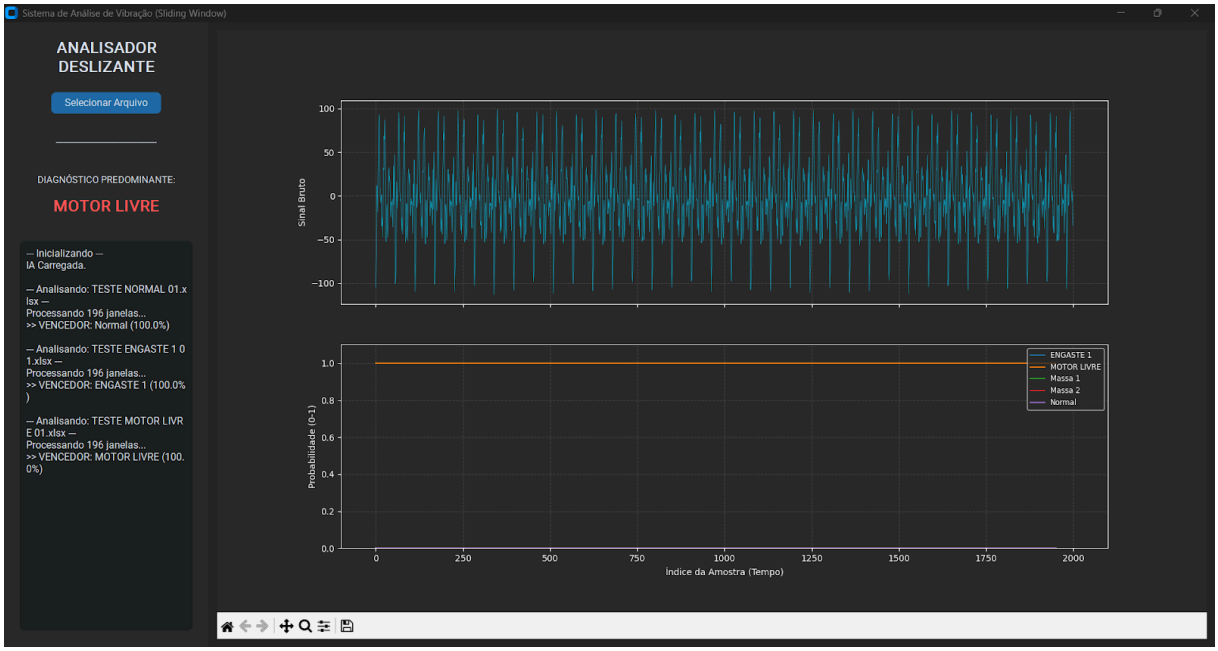

Figura 4.19: Interface diagnosticando a instabilidade de fixação ("Motor Livre").

Fonte: Pesquisa direta (2025)

A capacidade de identificar instabilidades mecânicas foi testada na condição de folga. A Figura 4.19 demonstra a resposta do sistema ao cenário "Motor Livre". Mesmo com o sinal apresentando picos caóticos e transientes, a rede neural classificou corretamente a anomalia, distinguindo-a das demais anomalias .

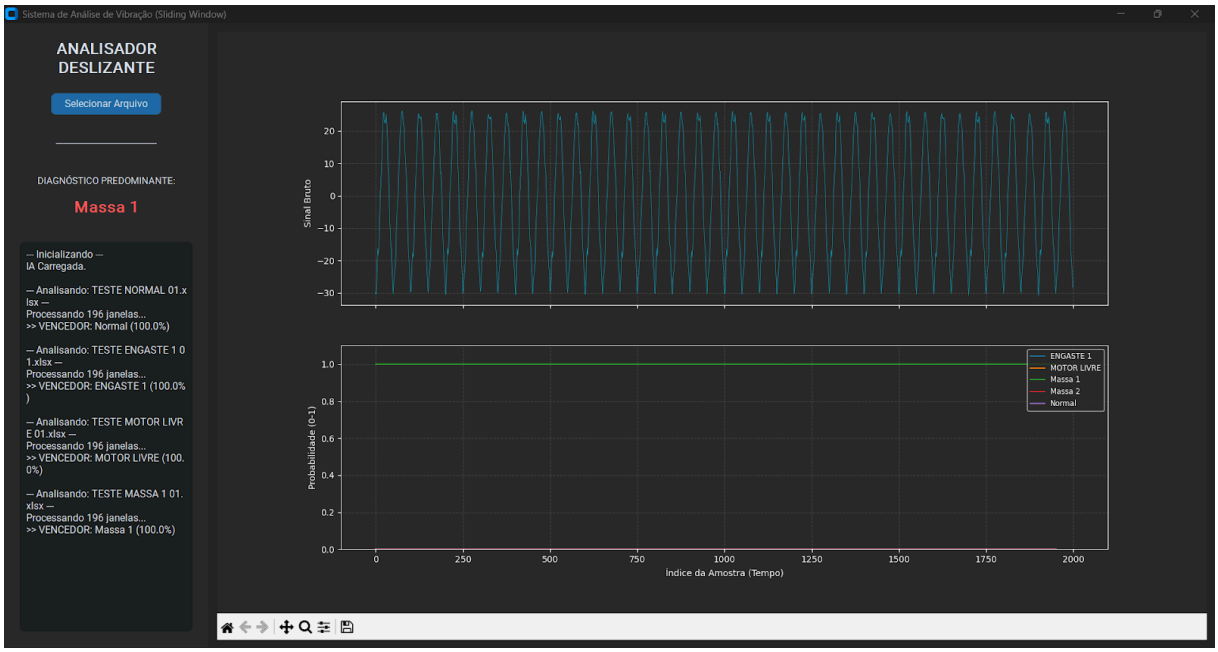

Figura 4.20: Detecção em tempo real da condição "Massa Adicional 1".

Fonte: Pesquisa direta (2025)

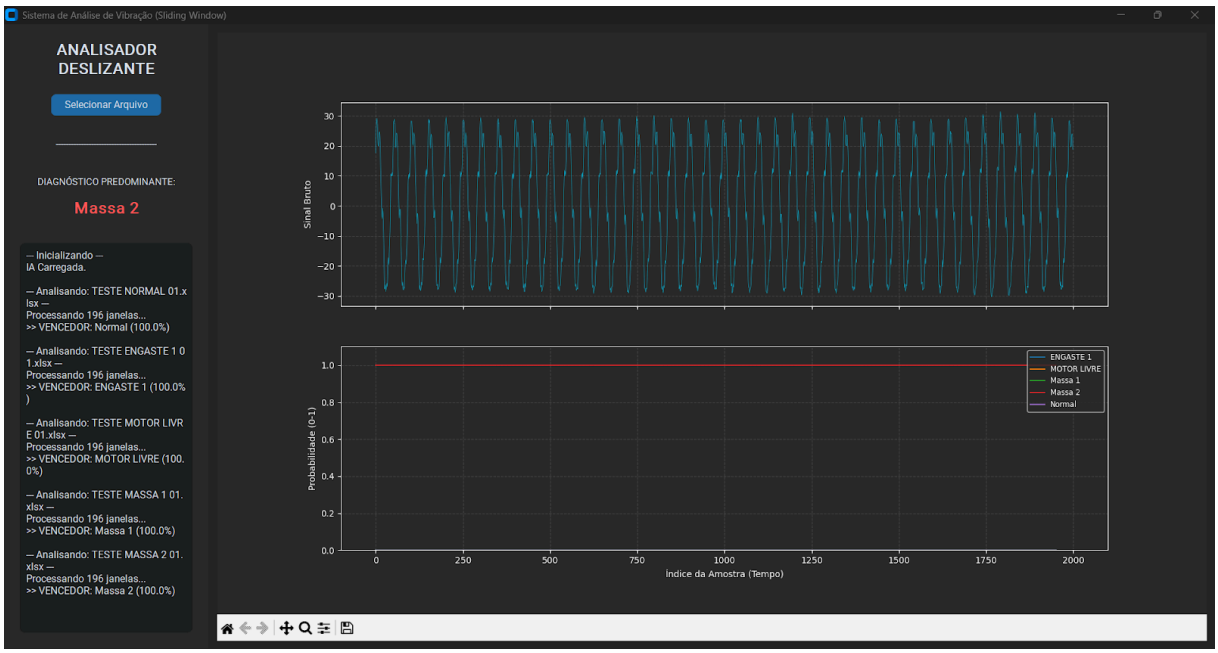

Figura 4.21: Detecção em tempo real da condição "Massa Adicional 2".

Fonte: Pesquisa direta (2025)

Por fim, testou-se a sensibilidade do modelo a desbalanceamentos de massa, as condições mais sutis do estudo. As Figuras 4.20 e 4.21 comprovam que o sistema foi capaz de diferenciar não apenas a presença de uma massa adicional, mas também a sua posição específica na viga (Massa 1 vs. Massa 2).

Os testes de validação confirmaram a viabilidade da metodologia proposta. Estes resultados atingiram o objetivo geral do trabalho, demonstrando que a solução desenvolvida supera as limitações de sistemas tradicionais baseados apenas em limites (thresholds) de amplitude. Enquanto um sistema convencional apenas indicaria "vibração alta" para todas as anomalias , a solução com LSTM informou a causa raiz específica, permitindo um planejamento de manutenção preditiva muito mais assertivo e eficiente.

4.5.1. Monitoramento Contínuo em Tempo Real

A validação do sistema em operação contínua foi projetada para garantir eficiência computacional e resposta instantânea. Mantendo a taxa de aquisição em 1000 Hz (idêntica à do treinamento), a arquitetura do *software* foi dividida em duas funções simultâneas:

1. Visualização Histórica: Foi implementado um buffer circular (deque) que armazena os últimos 1000 pontos de leitura. Sua função é puramente visual, permitindo apresentar no gráfico da interface o comportamento do sinal no último 1 segundo.

2. Inferência Inteligente: Para o diagnóstico, o sistema não processa todo o histórico. A cada 100ms, o algoritmo extrai apenas a janela mais recente de 50 pontos (equivalente a 50ms de sinal) e a submete à rede LSTM.

Essa estratégia de processar apenas a "ponta" do sinal reduz drasticamente a carga de processamento. Na prática, isso resultou em um sistema robusto contra latência: a transição de estados (como a simulação de "Motor Livre") é detectada e exibida na interface instantaneamente, sem causar travamentos na atualização gráfica.

5 CONCLUSÃO E RECOMENDAÇÕES

Este capítulo final consolida os resultados obtidos e avalia o desenvolvimento do trabalho face aos objetivos propostos, respondendo à problemática central da investigação. Adicionalmente, são apresentadas recomendações para trabalhos futuros que possam dar continuidade e expandir a metodologia aqui desenvolvida.

5.1. Conclusões

A problemática central deste trabalho foi definida pela questão: *"Como identificar anomalias mecânicas por meio de redes neurais e sinais de vibração?"*. Com base nos resultados apresentados no Capítulo 4, conclui-se que é plenamente possível não apenas identificar, mas classificar com certa precisão diferentes tipologias de anomalias mecânicas utilizando a metodologia proposta.

O objetivo geral de *"desenvolver uma metodologia baseada em redes neurais para a monitorização de anomalias a partir da análise de sinais de vibração de máquinas"* foi alcançado com êxito. O sistema desenvolvido, integrando uma bancada experimental para aquisição de dados, uma rede neural recorrente do tipo LSTM (*Long Short-Term Memory*) e uma interface gráfica para diagnóstico, demonstrou eficácia no ambiente controlado.

As principais conclusões desta investigação podem ser detalhadas nos seguintes pontos:

1. **Cumprimento dos Objetivos Específicos:** A totalidade dos objetivos específicos foi atingida. Realizou-se o estudo aprofundado das técnicas de monitorização (Capítulo 2), procedeu-se à recolha e categorização dos dados de vibração em classes distintas (Normal, Massa 1, Massa 2, Variação no Engaste, etc.) e implementou-se a rede neural. A validação (Capítulo 4) comprovou a superioridade desta abordagem face às técnicas tradicionais, que frequentemente se limitam à análise de valores globais (como o RMS), visto que o sistema proposto demonstrou capacidade para discernir a **natureza** específica da anomalia.
2. **Validade da Metodologia (Dados brutos):** A decisão metodológica de utilizar os sinais de vibração em bruto (*raw data*) no domínio do tempo como entrada direta para a rede LSTM revelou-se robusta. Conforme corroborado por Afridi *et al.* (2023), esta abordagem

permitiu que o próprio modelo aprendesse a extrair as características temporais relevantes para a classificação, preservando informações transientes que poderiam ser suprimidas em transformações de domínio, como a Transformada Rápida de Fourier (FFT).

3. **Desempenho do Modelo:** Os resultados obtidos indicaram uma precisão máxima na classificação das diferentes condições estruturais. O modelo não se manteve na simples diferenciação binária entre "Normal" e "Anomalia", demonstrando capacidade para distinguir com exatidão entre diferentes modos de falha (por exemplo, diferenciar "Massa Adicional" de "Folga Mecânica"). Este nível de granularidade no diagnóstico representa um avanço significativo para estratégias de manutenção preditiva direcionada.
4. **Viabilidade do Sistema:** A integração final com a interface gráfica (desenvolvida em *CustomTkinter*) validou o sistema como uma ferramenta de diagnóstico funcional. A capacidade de fornecer ao operador uma resposta imediata e visual sobre o estado estrutural do equipamento confirma a aplicabilidade prática da solução.

Em suma, este trabalho não apenas respondeu à problemática inicial, como também entregou um protótipo funcional de um sistema de monitorização inteligente. Demonstrou-se, assim, que a fusão da análise vibratória com redes neurais recorrentes constitui uma solução robusta e promissora para a indústria 4.0 e a manutenção preditiva moderna.

5.2 Recomendações

Considerando os resultados promissores e as limitações inerentes a um estudo em bancada experimental, sugerem-se as seguintes linhas de investigação para a continuidade deste trabalho:

1. **Validação em Ambiente Industrial:** Aplicação da metodologia em equipamentos reais no chão de fábrica, onde a presença de ruídos externos, vibrações de máquinas adjacentes e variações térmicas impõem desafios adicionais à robustez da rede neural;
2. **Fusão de Sensores:** Investigar o impacto da adição de outras variáveis ao modelo, como temperatura, corrente elétrica do motor ou emissão acústica, criando um sistema de diagnóstico multimodal;
3. **Implementação em *Edge Computing*:** Portar o modelo treinado para sistemas embarcados de baixo custo e baixo consumo (como Raspberry Pi ou Jetson Nano),

eliminando a necessidade de um computador dedicado e permitindo a análise local dos dados (*IoT*);

4. **Comparação de Arquiteturas:** Realizar um estudo comparativo entre a LSTM e outras arquiteturas de *Deep Learning*, como Redes Neurais Convolucionais (CNN) unidimensionais ou *Transformers*, para avaliar ganhos em eficiência computacional ou tempo de treino;
5. **Correlação entre Falhas Controladas e Reais (Generalização):** Desenvolver um estudo comparativo que relacione anomalias reais de campo (frequentemente complexas e ruidosas) com associações ou combinações das anomalias controladas isoladas em laboratório. O objetivo é investigar a capacidade do sistema em reconhecer anomalias ainda desconhecidas (*novelty detection*) ou anomalias compostas, inferindo o diagnóstico a partir da identificação de padrões parciais já aprendidos durante o treino em bancada;

REFERÊNCIA BIBLIOGRÁFICA

AFRIDI, Yasir Saleem et al. *LSTM-Based Condition Monitoring and Fault Prognostics of Rolling Element Bearings Using Raw Vibrational Data.* ***Machines***, v. 11, n. 5, e531, 2023.

ANDRADE, Ana Carla Costa. **Desenvolvimento de uma metodologia utilizando rede neural artificial na detecção e diagnóstico de falhas para válvula de controle pneumática**. 2021. 80 f. Tese (Doutorado em Ciência e Engenharia de Petróleo) – Centro de Ciências Exatas e da Terra, Universidade Federal do Rio Grande do Norte, Natal, 2021.

BHATIA, K.; GUPTA, A. *A Comparative Survey of PyTorch vs TensorFlow for Deep Learning: Usability, Performance, and Deployment Trade-offs.* **arXiv preprint arXiv:2508.04035**, 2025.

BRANCO, Nathielle Waldrigues et al. *Wavelet LSTM for Fault Forecasting in Electrical Power Grids.* ***Sensors***, v. 22, n. 21, e8323, 2022.

CORMEN, Thomas H. et al. **Algoritmos**. 3. ed. Tradução de Arlete Simille Marques. Rio de Janeiro: Elsevier, 2012.

CORMEN, T. H.; LEISERSON, C. E.; RIVEST, R. L.; STEIN, C. **Algoritmos**: teoria e prática. 3. ed. Rio de Janeiro: Elsevier, 2009.

DIAS, Aldilene Oliveira Maia. **Aprendizado de máquina aplicado a predição de falhas em caminhão fora de estrada**. 2020. Dissertação (Mestrado em Instrumentação, Controle e Automação de Processos de Mineração) – Programa de Pós-Graduação em Instrumentação, Controle e Automação de Processos de Mineração (PROFICAM), Escola de Minas, Universidade Federal de Ouro Preto, Ouro Preto, 2020.

FILHO, Mario. **Como prever séries temporais com LSTM em Python**. 25 jan. 2023. Disponível em: https://mariofilho.com/como-prever-series-temporais-com-lstm-em-python/. Acesso em: 14 nov. 2025.

GU, Kai et al. *DWT-LSTM-Based Fault Diagnosis of Rolling Bearings with Multi-Sensors.* ***Electronics***, v. 10, n. 17, e2076, 2021.

HOCHREITER, S.; SCHMIDHUBER, J. *Long Short-Term Memory.* ***Neural Computation***, v. 9, n. 8, p. 1735–1780, 1997.

HUANG, Samuel H.; ZHANG, Hong-Chao. ***Artificial neural networks in manufacturing: concepts, applications, and perspectives. IEEE Transactions on Components, Packaging, and Manufacturing Technology*** - Part A, v. 17, n. 2, p. 212, jun. 1994.

INMAN, D. J. ***Engineering vibration***. 4. ed. Boston: Pearson, 2014.

MEIROVITCH, L. **Fundamentals of vibrations**. New York: McGraw-Hill, 2001.

MEYER ZU WICKERN, Vincent. ***Challenges and Reliability of Predictive Maintenance***. 2019. DOI: 10.13140/RG.2.2.35379.89129.

MINUSSI, Carlos Roberto; LOTUFO, Anna Diva Plasencia (Colab.). **Rede Neural**. São Paulo: Universidade Estadual Paulista – UNESP, Faculdade de Engenharia de Ilha Solteira, Programa de Pós-Graduação em Engenharia Elétrica, 2008.

NEPOMUCENO, Lauro Xavier. **Técnicas de manutenção preditiva**. 1. ed. São Paulo: Edgard Blücher Ltda, 1989.

NICOLAU, M. G.; MANCINI, C. H. **Algoritmos e programação**: teoria e prática. São Paulo: Érica, 2005.

PROAKIS, J. G.; MANOLAKIS, D. G. ***Digital Signal Processing***: *Principles, Algorithms, and Applications*. 4. ed. Upper Saddle River: Prentice Hall, 2007.

RANDALL, R. B. ***Vibration-based Condition Monitoring****:* Industrial, Aerospace and Automotive Applications. Chichester: John Wiley & Sons, 2011.

RAO, S. S. **Vibrações Mecânicas**. 5. ed. São Paulo: Pearson Prentice Hall, 2008.

XENOS, H. G. **Gerenciando a manutenção produtiva**: melhores práticas para eliminar falhas nos equipamentos e maximizar a produtividade. 2. ed. Falconi, 2014.

MOBLEY, R. K. ***An Introduction to Predictive Maintenance***. 2. ed. Elsevier, 2002.

LEI, Y. et al. ***A review on empirical mode decomposition in fault diagnosis of rotating machinery***. *Mechanical Systems and Signal Processing*, 2018.

BENBOUZID, M. E. H. ***A review of induction motors signature analysis as a medium for faults detection***. *IEEE Transactions on Industrial Electronics*, 2000.

BAGAVATHIAPPAN, S. et al. ***Infrared thermography for condition monitoring – A review.*** *Infrared Physics & Technology*, 2013.

MOUBRAY, J. ***Reliability-centered maintenance.*** *Industrial Press Inc*., 1997.

ZHANG, W. et al. ***Deep convolutional neural networks for bearing fault diagnosis***. *Measuremen*t, 2017.

KINGMA, D. P.; BA, J. L. ***Adam: A Method for Stochastic Optimization***. *International Conference on Learning Representations* (ICLR), 2014.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. ***Deep Learning***. MIT *Press*, 2016. (A "bíblia" do *Deep Learning*).

ANEXO

APÊNDICE A – Algoritmo da Rede Neural LSTM

Este apêndice apresenta o código fonte desenvolvido em linguagem Python, utilizando a biblioteca TensorFlow/Keras, responsável pela arquitetura da Rede Neural Recorrente (LSTM) e pelo pré-processamento dos sinais de vibração.

```
import os
import pandas as pd
import numpy as np
import pickle
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import confusion_matrix
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense, Dropout, Input
from tensorflow.keras.utils import to_categorical


# ----------------------------------------------------------------
# --- PARÂMETROS ---
# ----------------------------------------------------------------
# Pasta contendo os dados de TREINAMENTO (Subpastas = Categorias)
TRAIN_DIR       =       r'C:/Users/Sjose/OneDrive/Area       Oculta/Area
oculta/LSTM/Categorias'


# Pasta contendo os dados de TESTE/VALIDAÇÃO (Subpastas = Categorias)
TEST_DIR = r'C:\Users\Sjose\OneDrive\Area Oculta\Area oculta\LSTM\Teste'
```

```
WINDOW_SIZE = 50
STRIDE = 10
EPOCHS = 100
BATCH = 64
# ----------------------------------------------------------------------


# ----------------------------------------------------------------------
# --- Funções Auxiliares ---
# ----------------------------------------------------------------------

def clean_dataframe(df):
    """Limpa o DataFrame carregado."""
    # Garante que pegamos apenas as duas primeiras colunas relevantes
    df = df.iloc[:, :2]
    df.columns = ['Time', 'Amplitude']

    # Limpeza de strings e conversão para número
    for col in ['Time', 'Amplitude']:
        if df[col].dtype == object:
                            df[col]  =  df[col].astype(str).str.replace(',',
'.').str.replace(r'[^\d\.-]', '', regex=True)

    df['Time'] = pd.to_numeric(df['Time'], errors='coerce')
    df['Amplitude'] = pd.to_numeric(df['Amplitude'], errors='coerce')
    df.dropna(inplace=True)
    return df

def create_sliding_windows(data_series, window_size, stride):
```

```
    total_len = len(data_series)
    windows = []
    for i in range(0, total_len - window_size + 1, stride):
        window = data_series[i : i + window_size]
        windows.append(window)
    return np.array(windows)

def load_file_data(file_path):
    try:
        if file_path.lower().endswith('.csv'):
            df = pd.read_csv(file_path)
        elif file_path.lower().endswith(('.xlsx', '.xls')):
            df = pd.read_excel(file_path, engine='openpyxl')
        else:
            return None

        df = clean_dataframe(df)
        return df['Amplitude'].values
    except Exception as e:
        print(f"Erro ao ler {os.path.basename(file_path)}: {e}")
        return None

def map_files_in_folder(base_dir):
     """Varre subpastas de um diretório e retorna dicionário {categoria:
[arquivos]}"""
    mapped_files = {}
    if not os.path.exists(base_dir):
        print(f"ERRO: O diretório não existe: {base_dir}")
```

```
        return mapped_files

    for root, dirs, files in os.walk(base_dir):
        if root == base_dir: continue # Pula a raiz, olha só as subpastas

        category = os.path.basename(root)
              valid_files = [os.path.join(root, f) for f in files if
f.lower().endswith(('.csv', '.xlsx', '.xls'))]

        if valid_files:
            mapped_files[category] = valid_files

    return mapped_files

def process_file_list(files_dict, set_name="Dados"):
    """Lê os arquivos do dicionário e cria as janelas"""
    windows_list = []
    labels_list = []

    print(f"\nProcessando {set_name}...")

    for category, file_list in files_dict.items():
        print(f"  > Categoria '{category}': {len(file_list)} arquivos.")
        for fpath in file_list:
            data = load_file_data(fpath)
            if data is not None:
                wins = create_sliding_windows(data, WINDOW_SIZE, STRIDE)
                if len(wins) > 0:
```

```
                    windows_list.extend(wins)
                    labels_list.extend([category] * len(wins))

    return np.array(windows_list), labels_list

# ------------------------------------------------------------------
# --- PASSO 1: Mapeamento de Arquivos ---
# ------------------------------------------------------------------
print(f"--- 1. MAPEANDO ARQUIVOS ---")

# Mapeia Treino
print(f"Buscando treino em: {TRAIN_DIR}")
train_files_map = map_files_in_folder(TRAIN_DIR)

# Mapeia Teste
print(f"Buscando teste em: {TEST_DIR}")
test_files_map = map_files_in_folder(TEST_DIR)

if not train_files_map:
    print("ERRO CRÍTICO: Nenhum arquivo de TREINO encontrado.")
    exit()

if not test_files_map:
        print("ERRO CRÍTICO: Nenhum arquivo de TESTE encontrado nas
subpastas.")
    exit()

# Validação de categorias
```

```
train_cats = set(train_files_map.keys())
test_cats = set(test_files_map.keys())

print(f"\nCategorias de Treino: {list(train_cats)}")
print(f"Categorias de Teste:  {list(test_cats)}")

if not test_cats.issubset(train_cats):
    print("AVISO: Existem categorias no Teste que não existem no Treino!")
    print(f"Diferença: {test_cats - train_cats}")

# ---------------------------------------------------------------------
# --- PASSO 2: Carregamento e Processamento ---
# ---------------------------------------------------------------------
X_train, train_labels = process_file_list(train_files_map, "TREINO")
X_test, test_labels = process_file_list(test_files_map, "TESTE")

# Reshape para LSTM (Samples, Timesteps, Features)
if X_train.shape[0] > 0:
    X_train = X_train.reshape((X_train.shape[0], X_train.shape[1], 1))
if X_test.shape[0] > 0:
    X_test = X_test.reshape((X_test.shape[0], X_test.shape[1], 1))

print(f"\n--- ESTATÍSTICAS FINAIS DOS DADOS ---")
print(f"Total de Janelas de TREINO: {X_train.shape[0]}")
print(f"Total de Janelas de TESTE:  {X_test.shape[0]}")

if X_train.shape[0] == 0 or X_test.shape[0] == 0:
```

```
    print("ERRO CRÍTICO: Dados insuficientes.")
    exit()

# ------------------------------------------------------------------
# --- PASSO 3: Codificação de Rótulos ---
# ------------------------------------------------------------------
encoder = LabelEncoder()
# Fit apenas nas categorias de treino (o modelo só aprende o que treina)
encoder.fit(list(train_files_map.keys()))

# Transforma labels
# Obs: Se houver classe no teste que não tem no treino, isso daria erro.
# Filtramos o teste para garantir integridade ou assumimos que o usuário garantiu a estrutura.
try:
    y_train_hot = to_categorical(encoder.transform(train_labels))
    y_test_hot = to_categorical(encoder.transform(test_labels))
except ValueError as e:
    print(f"\nERRO DE CATEGORIA: {e}")
        print("Certifique-se que a pasta de Teste não tem categorias desconhecidas pelo Treino.")
    exit()

num_classes = len(encoder.classes_)
class_names = list(encoder.classes_)
print(f"Classes Codificadas: {class_names}")

# ------------------------------------------------------------------
```

```
# --- PASSO 4: Modelo LSTM ---
# ----------------------------------------------------------------------
model = Sequential()
model.add(Input(shape=(WINDOW_SIZE, 1)))
model.add(LSTM(64, return_sequences=False))
model.add(Dropout(0.3))
model.add(Dense(32, activation='relu'))
model.add(Dense(num_classes, activation='softmax'))

model.compile(loss='categorical_crossentropy',           optimizer='adam',
metrics=['accuracy'])

# ----------------------------------------------------------------------
# --- PASSO 5: Treinar ---
# ----------------------------------------------------------------------
print("\n--- INICIANDO TREINAMENTO ---")
history = model.fit(
    X_train, y_train_hot,
    epochs=EPOCHS,
    batch_size=BATCH,
    validation_data=(X_test, y_test_hot)
)

# ----------------------------------------------------------------------
# --- PASSO 6: Avaliação e Relatório ---
# ----------------------------------------------------------------------
print("\n" + "="*50)
print("          RELATÓRIO FINAL")
```

```
print("="*50)

loss, accuracy = model.evaluate(X_test, y_test_hot, verbose=0)
print(f"Acurácia Global (Baseada na pasta Teste):  {accuracy*100:.2f}%")
print(f"Loss Final:                                {loss:.4f}")

# Previsões
y_pred_probs = model.predict(X_test, verbose=0)
y_pred_classes = np.argmax(y_pred_probs, axis=1)
y_true_classes = np.argmax(y_test_hot, axis=1)

print("\n--- Matriz de Confusão ---")
cm = confusion_matrix(y_true_classes, y_pred_classes)
print(cm)

# Plotar Matriz
plt.figure(figsize=(10, 8))
sns.heatmap(cm,          annot=True,          fmt='d',          cmap='Blues',
xticklabels=class_names, yticklabels=class_names)
plt.title('Matriz de Confusão (Dados da pasta Teste)')
plt.ylabel('Real')
plt.xlabel('Previsto')
plt.tight_layout()
plt.savefig('matriz_confusao.png')
print("Gráfico da Matriz salvo como 'matriz_confusao.png'")

# ----------------------------------------------------------------
# --- PASSO 7: Salvar Artefatos ---
```

```
# ----------------------------------------------------------------------

print("\nSalvando modelo e encoder...")

model.save('modelo_lstm_vibracao_v2.h5')

with open('classes_encoder.pkl', 'wb') as f:

    pickle.dump(encoder.classes_, f)


# ----------------------------------------------------------------------

# --- PASSO 8: Gerar Gráficos de Evolução ---

# ----------------------------------------------------------------------

print("\nGerando gráficos de evolução...")


plt.figure(figsize=(12, 5))


# Gráfico Acurácia

plt.subplot(1, 2, 1)

plt.plot(history.history['accuracy'], label='Treino', color='blue')

plt.plot(history.history['val_accuracy'], label='Teste (Pasta Separada)',
color='orange')

plt.title('Evolução da Acurácia')

plt.xlabel('Épocas')

plt.ylabel('Acurácia')

plt.legend()

plt.grid(True, linestyle='--', alpha=0.5)


# Gráfico Loss

plt.subplot(1, 2, 2)

plt.plot(history.history['loss'], label='Treino', color='blue')
```

```
plt.plot(history.history['val_loss'], label='Teste (Pasta Separada)', color='red')
plt.title('Evolução do Erro (Loss)')
plt.xlabel('Épocas')
plt.ylabel('Loss')
plt.legend()
plt.grid(True, linestyle='--', alpha=0.5)

plt.tight_layout()
plt.savefig('graficos_evolucao.png', dpi=300)
print("Concluído.")
```

APÊNDICE B – Código da Interface Gráfica e Monitoramento em Tempo Real

Este apêndice apresenta os trechos principais do software de monitoramento desenvolvido com a biblioteca CustomTkinter. O código abaixo demonstra a lógica de aquisição de dados e a atualização da interface para diagnóstico instantâneo.

```
import customtkinter as ctk
from tkinter import filedialog, messagebox
import os
import pandas as pd
import numpy as np
import pickle
import time
from collections import deque
```

```
from datetime import datetime

# --- CORREÇÃO GRÁFICA TKINTER ---
import matplotlib
matplotlib.use('TkAgg')
# --------------------------------

import matplotlib.pyplot as plt
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg
from tensorflow.keras.models import load_model

# Tenta importar nidaqmx
try:
    import nidaqmx
     from nidaqmx.constants import TerminalConfiguration, AcquisitionType,
ExcitationSource, AccelSensitivityUnits, AccelUnits, READ_ALL_AVAILABLE
    HAS_NI = True
except ImportError:
    HAS_NI = False

# ---------------------------------------------------------------------
# CONFIGURAÇÕES DO SISTEMA
# ---------------------------------------------------------------------
WINDOW_SIZE = 50       # Pontos necessários para a IA (Janela de entrada)
NI_CHANNEL = "cDAQ1Mod1/ai0"  # <--- SEU CANAL NI (Verifique no NI MAX)
SAMPLE_RATE = 1000     # Hz (Taxa de Amostragem)
BUFFER_SIZE = 1000        # Tamanho do gráfico (1 segundo de histórico
visual)
```

```
UPDATE_INTERVAL = 100  # Atualiza a tela a cada 100 ms

ctk.set_appearance_mode("Dark")
ctk.set_default_color_theme("blue")

class VibrationApp(ctk.CTk):
    def __init__(self):
        super().__init__()

          self.title("Sistema de Monitoramento de Vibração em Tempo Real - IA")
        self.geometry("1100x700")

        # --- CONFIGURAÇÃO DE ENCERRAMENTO SEGURO ---
        self.protocol("WM_DELETE_WINDOW", self.on_closing)

        # Configuração do Grid Principal
        self.grid_columnconfigure(0, weight=1)
            self.grid_rowconfigure(2, weight=1) # O gráfico (row 2) vai expandir

        # Variáveis Globais
        self.model = None
        self.classes = None

        # Variáveis de Controle Tempo Real
        self.is_running = False
```

```
            self.data_buffer = deque([0]*BUFFER_SIZE, maxlen=BUFFER_SIZE) #
Buffer circular para o gráfico
        self.task_ni = None # Tarefa do NI-DAQmx

        # Inicialização
        self.load_ai_system()
        self.setup_ui()

                                                                            #
============================================================================
    # FUNÇÃO DE ENCERRAMENTO SEGURO
                                                                            #
============================================================================
    def on_closing(self):
        """Desliga a aquisição e a thread gráfica antes de fechar"""
        print("Encerrando sistema...")
        self.is_running = False

        if self.task_ni:
            try:
                self.task_ni.stop()
                self.task_ni.close()
            except: pass
            self.task_ni = None

        plt.close('all') # Fecha figuras do matplotlib
        self.quit()
        self.destroy()
```

```
                                                                        #
==========================================================================

    # CARREGAMENTO DA IA

                                                                        #
==========================================================================

    def load_ai_system(self):

        print("--- Inicializando IA ---")

        try:

            # 1. Carrega o Modelo (.h5)

            if os.path.exists('modelo_lstm_vibracao1.h5'):

                 self.model = load_model('modelo_lstm_vibracao1.h5')

                 print("Modelo '_1.h5' carregado.")

            elif os.path.exists('modelo_lstm_vibracao.h5'):

                 self.model = load_model('modelo_lstm_vibracao.h5')

                 print("Modelo padrão '.h5' carregado.")

            else:

                print("ERRO: Nenhum modelo encontrado!")


              # 2. DEFINIÇÃO DAS CLASSES REAIS (Baseado na ordem alfabética
das pastas)

            self.classes = [

                'ENGASTE',       # Índice 0

                'MASSA 1',    # Índice 1

                'MASSA 2',    # Índice 2

                'MASSA 1',     # Índice 3

                'MOTOR LIVRE',   # Índice 4

                'NORMAL'         # Índice 5
```

```
            ]
            print(f"Classes definidas manualmente: {self.classes}")

        except Exception as e:
            print(f"Erro Crítico ao carregar IA: {e}")

    # ========================================================================
    # INTERFACE DE USUÁRIO (UI)
    # ========================================================================
    def setup_ui(self):
        # --- 1. BARRA DE CONTROLE (BOTÕES) ---
        control_frame = ctk.CTkFrame(self, height=80, fg_color="transparent")
        control_frame.grid(row=0, column=0, sticky="ew", padx=20, pady=10)

        ctk.CTkLabel(control_frame, text="DAQ SYSTEM:", font=ctk.CTkFont(size=14, weight="bold")).pack(side="left", padx=10)

        self.btn_start = ctk.CTkButton(control_frame, text="▶ INICIAR", fg_color="#27ae60", width=120, font=ctk.CTkFont(weight="bold"), command=self.start_realtime)
        self.btn_start.pack(side="left", padx=5)

        self.btn_stop = ctk.CTkButton(control_frame, text="■ PARAR", fg_color="#c0392b", state="disabled", width=120, font=ctk.CTkFont(weight="bold"), command=self.stop_realtime)
        self.btn_stop.pack(side="left", padx=5)
```

```
            self.btn_save = ctk.CTkButton(control_frame, text="💾 SALVAR EXCEL", fg_color="#2980b9", width=150, font=ctk.CTkFont(weight="bold"), command=self.salvar_amostra_excel)
        self.btn_save.pack(side="left", padx=20)

             self.lbl_status = ctk.CTkLabel(control_frame, text="Status: Aguardando...", text_color="gray", font=ctk.CTkFont(size=12))
        self.lbl_status.pack(side="right", padx=20)

        # --- 2. PAINEL DE DIAGNÓSTICO (IA) ---
                diag_frame = ctk.CTkFrame(self, fg_color="#181818", corner_radius=15)
        diag_frame.grid(row=1, column=0, sticky="ew", padx=20, pady=10)

            ctk.CTkLabel(diag_frame, text="DIAGNÓSTICO AUTOMÁTICO (IA):", font=ctk.CTkFont(size=12, weight="bold"), text_color="#aaaaaa").pack(pady=(15,0))

        # Correção: Reduzi a fonte para 32 para não cortar textos longos
                self.lbl_pred_rt = ctk.CTkLabel(diag_frame, text="---", font=ctk.CTkFont(size=32, weight="bold"), text_color="gray")
        self.lbl_pred_rt.pack(pady=(5,5))

         self.lbl_conf_rt = ctk.CTkLabel(diag_frame, text="Confiança: --%", font=ctk.CTkFont(size=16))
        self.lbl_conf_rt.pack(pady=(0,15))

        # --- 3. ÁREA DO GRÁFICO (MATPLOTLIB) ---
```

```
        plot_frame = ctk.CTkFrame(self, fg_color="transparent")
        plot_frame.grid(row=2, column=0, sticky="nsew", padx=20, pady=10)

        self.fig_rt, self.ax_rt = plt.subplots(figsize=(8, 4), dpi=100)

        # Estilização Dark Mode para o Matplotlib
        self.fig_rt.patch.set_facecolor('#2b2b2b') # Fundo da figura
            self.ax_rt.set_facecolor('#000000')            # Fundo do eixo
(gráfico)
        self.ax_rt.tick_params(colors='white')     # Cor dos números
        self.ax_rt.grid(True, linestyle=':', alpha=0.3, color='#444444')
                            for   spine   in   self.ax_rt.spines.values():
spine.set_color('#444444') # Bordas

                self.line_rt,  =  self.ax_rt.plot([],  [],  color='#00ff00',
linewidth=1.2) # Linha verde matrix
        self.ax_rt.set_xlim(0, BUFFER_SIZE)
            self.ax_rt.set_ylim(-30,  30)  # Ajuste  este  valor  conforme  a
amplitude do seu sensor (ex: -10 a 10 g)
             self.ax_rt.set_title("Sinal  de  Vibração  em  Tempo  Real  (g)",
color='white', fontsize=10)
             self.ax_rt.set_xlabel("Amostras  (Último  1s)",  color='white',
fontsize=8)
        self.ax_rt.set_ylabel("Amplitude", color='white', fontsize=8)

        self.canvas_rt = FigureCanvasTkAgg(self.fig_rt, master=plot_frame)
        self.canvas_rt.draw()
        self.canvas_rt.get_tk_widget().pack(fill="both", expand=True)
```

```
                                                                        #
==========================================================================
    # LÓGICA DE CONTROLE (START / STOP)
                                                                        #
==========================================================================
    def start_realtime(self):
        if self.is_running: return

        self.is_running = True
        self.btn_start.configure(state="disabled")
        self.btn_stop.configure(state="normal")
               self.lbl_status.configure(text="Status:  ●  Monitorando...",
text_color="#2ecc71")

        # Inicializa conexão com Hardware NI ou Simulação
        if HAS_NI:
            try:
                self.task_ni = nidaqmx.Task()
                self.task_ni.ai_channels.add_ai_accel_chan(
                    physical_channel=NI_CHANNEL,
                         sensitivity=100.0, # Sensibilidade do seu sensor
(mV/g)

sensitivity_units=AccelSensitivityUnits.MILLIVOLTS_PER_G,
                    units=AccelUnits.G,
                     current_excit_source=ExcitationSource.INTERNAL, # IEPE
Ligado
                    current_excit_val=0.004, # 4mA
                    terminal_config=TerminalConfiguration.PSEUDO_DIFF
```

```
                )
                  self.task_ni.timing.cfg_samp_clk_timing(rate=SAMPLE_RATE,
sample_mode=AcquisitionType.CONTINUOUS)
                self.task_ni.start()
                print("Hardware NI Iniciado com Sucesso.")
            except Exception as e:
                    print(f"Erro ao conectar NI: {e}. \n>> Iniciando Modo
SIMULAÇÃO.")
                self.task_ni = None
        else:
             print("Biblioteca nidaqmx não encontrada. \n>> Iniciando Modo
SIMULAÇÃO.")

        # Inicia o loop de atualização
        self.update_loop()

    def stop_realtime(self):
        self.is_running = False
        self.btn_start.configure(state="normal")
        self.btn_stop.configure(state="disabled")
                        self.lbl_status.configure(text="Status:   Parado",
text_color="gray")

        # Fecha a tarefa do NI com segurança
        if self.task_ni:
            try:
                self.task_ni.stop()
                self.task_ni.close()
```

```
            except: pass
            self.task_ni = None

                                                                                #
================================================================================
    # LOOP PRINCIPAL (ATUALIZAÇÃO DE DADOS E GRÁFICOS)
                                                                                #
================================================================================
    def update_loop(self):
           """Loop recursivo chamado pelo Tkinter a cada UPDATE_INTERVAL
ms"""
        if not self.is_running: return

        try:
            # 1. AQUISIÇÃO DE DADOS
            if self.task_ni:
                  # Lê tudo que chegou no buffer do hardware desde a última
vez
                                                          data_chunk    =
self.task_ni.read(number_of_samples_per_channel=READ_ALL_AVAILABLE)
                if not data_chunk: data_chunk = []
            else:
                # Simulação (gera ruído + senoide para testar sem sensor)
                time.sleep(0.05) # Simula delay de hardware
                t = time.time()
                # Gera 50 pontos (50ms a 1000Hz)
                noise = np.random.normal(0, 0.5, int(SAMPLE_RATE*0.05))
                wave = 5 * np.sin(2 * np.pi * 60 * t) # 60Hz
                data_chunk = (noise + wave).tolist()
```

```
            # 2. PROCESSAMENTO
            if len(data_chunk) > 0:
                    # Adiciona novos dados ao buffer circular (empurra os
velhos para fora)
                self.data_buffer.extend(data_chunk)

                # Atualiza o gráfico
                self.line_rt.set_ydata(self.data_buffer)
                self.line_rt.set_xdata(range(len(self.data_buffer)))
                   self.canvas_rt.draw_idle() # draw_idle é mais eficiente
que draw()

                # 3. INFERÊNCIA DA IA (Se tiver dados suficientes)
                if self.model and len(self.data_buffer) >= WINDOW_SIZE:
                    # Pega apenas os últimos WINDOW_SIZE pontos para a IA
                                                        input_window   =
np.array(list(self.data_buffer)[-WINDOW_SIZE:])
                      input_reshaped = input_window.reshape(1, WINDOW_SIZE,
1)

                    # Roda a predição
                    preds = self.model.predict(input_reshaped, verbose=0)

                    # Pega a classe vencedora
                    idx_winner = np.argmax(preds)

                    # --- PROTEÇÃO CONTRA ÍNDICE FORA DO LIMITE ---
```

```
                            if self.classes is not None and idx_winner <
len(self.classes):
                        class_name = self.classes[idx_winner]
                    else:
                        class_name = f"CLASSE_{idx_winner}"

                    confidence = preds[0][idx_winner] * 100

                    # Atualiza os textos na tela
                    self.update_labels(class_name, confidence)

        except Exception as e:
            print(f"Erro no loop de atualização: {e}")
            return

        # Agenda a próxima execução deste loop
        if self.is_running:
            self.after(UPDATE_INTERVAL, self.update_loop)

                                                                              #
==============================================================================
    # FUNÇÕES AUXILIARES
                                                                              #
==============================================================================
    def update_labels(self, label, conf):
        """Atualiza cor e texto do diagnóstico baseado na classe"""

        label_upper = str(label).upper()
```

```
        color = "#ffffff" # Branco (Padrão)

        # --- LÓGICA DE CORES ---

        # 1. NORMAL (Verde)
        if "NORMAL" in label_upper:
            color = "#2ecc71" # Verde

        # 2. ALERTA DE MASSA (Amarelo)
        elif "MASSA" in label_upper:
            color = "#f1c40f" # Amarelo

        # 3. CRÍTICO (Vermelho)
        elif "ENGASTE" in label_upper or "MOTOR" in label_upper:
            color = "#e74c3c" # Vermelho

        # 4. ERRO (Roxo)
        else:
             color = "#9b59b6"

        # Atualiza a interface
        self.lbl_pred_rt.configure(text=label_upper, text_color=color)
        self.lbl_conf_rt.configure(text=f"Confiança: {conf:.1f}%")

    def salvar_amostra_excel(self):
        """Salva o buffer atual (último 1 segundo) em um arquivo Excel"""
        if len(self.data_buffer) < 100:
```

```
            messagebox.showwarning("Aviso", "Poucos dados para salvar.")
            return

        try:
            dados = list(self.data_buffer)
            # Cria eixo de tempo relativo (0 a 1s)
            tempo = np.linspace(0, len(dados)/SAMPLE_RATE, len(dados))

            df = pd.DataFrame({'Time': tempo, 'Amplitude': dados})

                                                    nome_padrao     =
f"amostra_rt_{datetime.now().strftime('%H-%M-%S')}.xlsx"
                                                        caminho     =
filedialog.asksaveasfilename(defaultextension=".xlsx",
initialfile=nome_padrao, title="Salvar Amostra Excel")

            if caminho:
                df.to_excel(caminho, index=False)
                        messagebox.showinfo("Sucesso", f"Arquivo salvo com
sucesso!\n{caminho}")
        except Exception as e:
            messagebox.showerror("Erro ao Salvar", str(e))

if __name__ == "__main__":
    app = VibrationApp()
    app.mainloop()
```