UNIVERSIDADE FEDERAL DO ABC

CENTRO DE ENGENHARIA, MODELAGEM E CIÊNCIAS SOCIAIS APLICADAS - CECS

ESTI902-17 - TRABALHO DE GRADUAÇÃO III

João Victor Martins Reis 11201810522

SISTEMA PREDITIVO PARA ANÁLISE DE VIBRAÇÕES E TEMPERATURA EM MOTORES ELÉTRICOS COM REDE MESH BLUETOOTH

Prof Dr. Marco Aurelio Cazarotto Gomes

Santo André - SP

2024

João Victor Martins Reis 11201810522

SISTEMA PREDITIVO PARA ANÁLISE DE VIBRAÇÕES E TEMPERATURA EM MOTORES ELÉTRICOS COM REDE MESH BLUETOOTH

Relatório apresentado para projeto prático da disciplina de Trabalho de graduação III, ministrada pelo Prof Dr. Marco Aurelio Cazarotto Gomes.

Santo André - SP

2024

Sumário

1 Resumo

Este trabalho apresenta o projeto de desenvolvimento de produto com a implementação de um sistema embarcado inovador para manutenção preditiva (PdM - *Predictive Maintenance*) e monitoramento em tempo real de vibração e temperatura, voltado à análise e previsão de falhas em motores elétricos, aplicado a ambientes industriais. A solução proposta integra técnicas avançadas de Inteligência Artificial (IA) a uma rede Bluetooth Mesh de dispositivos de Internet das Coisas Industrial (IIoT), operando em microcontroladores (MCUs) Arm® Cortex®.

O projeto abrange o desenvolvimento completo do firmware, hardware e case mecânico, resultando em um produto versátil e robusto para aplicações de PdM em motores elétricos em diferentes cenários industriais. O projeto de hardware teve como base uma placa eletrônica fornecida pela empresa Pullup Soluções em Sistemas Eletrônicos, a qual integrava sensores de temperatura e aceleração e um microcontrolador Arm com suporte a Bluetooth. Essa plataforma inicial foi essencial para a realização dos testes práticos do sistema e também para a identificação de limitações críticas de hardware, como a quantidade restrita de memória RAM. Essa limitação impacta diretamente a implementação de modelos de aprendizado de máquina, uma vez que esses modelos frequentemente necessitam de grandes quantidades de RAM para armazenar e processar amostras dos sinais de temperatura e vibração. Tais limitações foram consideradas na concepção do novo projeto de hardware, que foi otimizado para superar essas barreiras, mas que, devido a restrições financeiras, não pôde ser produzido na fase final, apesar de estar pronto para fabricação.

A inovação do sistema reside na aplicação de técnicas de *edge computing* com aprendizado de máquina embarcado, permitindo que os dispositivos processem dados localmente e detectem anomalias em tempo real, sem depender de conexões externas. O uso de plataformas como o NanoEdge™ AI Studio foi essencial para a seleção e criação de modelos de IA otimizados, capazes de operar em ambientes com recursos computacionais limitados, mantendo alta eficiência energética. Um caso de uso envolvendo a classificação multiclasse

de anomalias em vibrações de motores elétricos demonstra as capacidades do sistema. Os resultados obtidos comprovam a eficácia da abordagem integrada, habilitando soluções escaláveis, autônomas e energeticamente eficientes, ampliando o alcance das aplicações em IoT industrial.

2 Introdução

Este trabalho tem como objetivo o desenvolvimento e a validação de uma solução baseada na tecnologia de rede Mesh Bluetooth aplicada a sensores industriais de temperatura e aceleração, com foco no monitoramento de motores elétricos. A proposta busca explorar as potencialidades dessa tecnologia, que, embora amplamente utilizada em ambientes residenciais [**1**], ainda se encontra em estágio inicial de aplicação no setor industrial. Além disso, o projeto inclui a implementação e teste de sua arquitetura em um sistema simplificado, permitindo a análise de detecção de falhas em motores elétricos por meio de sensores integrados. Para viabilizar os testes práticos, foi utilizada uma placa fornecida pela empresa Pullup Soluções em Sistemas Eletrônicos, o que contribuiu significativamente para a execução financeira do projeto.

A escolha dos sensores de temperatura e aceleração justifica-se pela sua relevância no monitoramento de falhas de equipamentos eletromecânicos, desempenhando um papel essencial na prevenção de falhas, na redução de manutenções corretivas inesperadas e no aumento da confiabilidade operacional. Este projeto busca não apenas ampliar o extenader o uso de redes Mesh Bluetooth em ambientes industriais, mas também fomentar a inovação no contexto da Indústria 4.0, ao propor soluções mais eficientes e integradas para os desafios do monitoramento e da manutenção preditiva.

Além disso, o projeto apresenta um potencial significativo para evolução futura. Com a utilização de algoritmos de machine learning, é possível desenvolver um sistema mais robusto de detecção e classificação de falhas em motores elétricos. Tal sistema poderia ser treinado com *datasets* públicos para identificar anomalias e classificar diferentes tipos de falhas, aprimorando ainda mais o monitoramento preditivo e garantindo maior eficiência e confiabilidade nos processos industriais.

Conforme discutido por Singh et al. (2020) [**2**], a implementação de *machine learning* em dispositivos de *edge* é um elemento estratégico, pois possibilita a análise local dos dados coletados, reduzindo a latência e aumentando a eficiência do sistema. Esse aspecto

é fundamental para garantir que os sensores possam processar informações em tempo real, aprimorando a resposta a falhas e o desempenho geral do sistema. Dessa forma, este trabalho apresenta um avanço significativo ao integrar tecnologias emergentes, como redes Mesh Bluetooth e *machine learning*, para atender às demandas da Indústria 4.0 e contribuir para a modernização de processos industriais.

3 Fundamentação Teórica

Este projeto abrange conceitos fundamentais da Internet das Coisas (IoT) , considerando o rápido crescimento do setor e os desafios associados à criação de sensores IoT eficientes. Tecnologias como *Edge Computing* [**3**] e *Fog Computing* desempenham um papel crucial na gestão do grande volume de dados gerados por sensores distribuídos, permitindo que o processamento e a tomada de decisões ocorram próximos à fonte de coleta. Isso reduz a latência e diminui a dependência da comunicação com a nuvem. A escalabilidade é outro fator essencial, pois, à medida que o número de dispositivos IoT cresce, torna-se crucial garantir que a infraestrutura seja capaz de lidar com a expansão sem comprometer o desempenho e a eficiência[**4**]. Para isso, o uso de microcontroladores de baixo consumo de energia e redes *Bluetooth Mesh* se destaca, não apenas por sua eficiência energética, mas também por facilitar a comunicação entre dispositivos em uma rede ampla e distribuída [**1**].

Além disso, a interoperabilidade entre diferentes plataformas IoT, utilizando protocolos de comunicação padronizados é fundamental para garantir a integração eficiente dos dispositivos. Protocolos como Bluetooth e I2C permitem que dispositivos de diversas naturezas e fabricantes comuniquem-se de forma transparente, contribuindo para a flexibilidade e o alcance da solução [**2**].

Por fim, a segurança da rede mesh é uma preocupação central, especialmente em ambientes IoT críticos, onde a comunicação entre dispositivos deve ser segura para evitar interceptações e ataques. Nesse contexto, o uso de mecanismos de criptografia e autenticação é indispensável para garantir a integridade e a confidencialidade dos dados trafegados. Diferente de outras soluções, o Bluetooth Mesh implementa privacidade por design, trocando e mascarando informações relevantes em cada transmissão de dados, tornando-se um aliado poderoso na criação de redes seguras e com privacidade em destaque [**5**]. Portanto, para que essas redes funcionem de forma eficaz e protegida, é essencial adotar uma abordagem de segurança desde a concepção do sistema [**5**].

3.1 Internet das Coisas (IoT)

A *Internet das Coisas* (IoT) refere-se à rede interconectada de dispositivos que se comunicam entre si e com a nuvem, facilitando a troca de dados e automação sem a necessidade de intervenção humana. Essa tecnologia se tornou possível graças ao avanço dos chips de baixo custo e das telecomunicações de alta largura de banda, permitindo que bilhões de dispositivos estejam conectados à internet [**6**]. Os dispositivos na IoT são compostos por sensores que monitoram variáveis como temperatura, umidade, luz, movimento, pressão ou vibração, e atuadores que interagem com o ambiente físico, realizando ações como abrir válvulas ou ligar motores. Juntos, sensores e atuadores formam a base da IoT, permitindo que máquinas e dispositivos interajam com o mundo físico de maneira automatizada e eficiente, viabilizando a automação de processos e o controle de máquinas através de tecnologias de análise de dados e *machine learning*, otimizando a produção e criando novos modelos de negócios [**7**].

Para a transmissão de dados entre sensores a IoT utiliza diversas tecnologias de conectividade, como Wi-Fi, Bluetooth, redes celulares, Zigbee e LoRaWAN, que garantem a comunicação confiável entre os dispositivos em diferentes cenários de aplicação [**5**]. O conceito de *Internet das Coisas Industrial* (IIoT) é a aplicação dessas tecnologias em ambientes industriais, onde a automação e o controle de processos são realizados por sensores conectados à nuvem, utilizando tecnologias como *Machine-to-Machine* (M2M) e *machine learning*. Com o uso de tecnologias avançadas, a IIoT (Industrial internet of things) tem revolucionado a automação industrial, permitindo maior eficiência, redução de custos e novos modelos de negócio. Esse movimento é frequentemente chamado de *Indústria 4.0*, representando a quarta revolução industrial [**6**].

Uma característica crucial dos dispositivos IoT é o baixo consumo de energia, essencial para dispositivos instalados em locais remotos ou de difícil acesso. Para garantir uma operação eficiente e de longa duração, muitos dispositivos dependem de fontes de energia alternativas, como energia solar. O baixo consumo de energia também possibilita a implantação de um maior número de sensores em uma área, permitindo uma cobertura mais detalhada e precisa do ambiente monitorado, contribuindo para a eficiência e escalabilidade dos sistemas IoT [**5**].

3.1.1 Mercado de IoT

De acordo com um estudo realizado pela McKinsey Company (2023), o mercado de IoT tem crescido exponencialmente, trazendo oportunidades significativas para diferentes setores. Esta seção discute como os usuários e fornecedores de tecnologias IoT capturam valor em diferentes cenários, explorando também as implicações econômicas e estratégicas dessas tecnologias. Assim como em outros mercados de tecnologia, o cliente final é quem absorve a maior parcela de valor. Estima-se que proprietários de fábricas que utilizam máquinas orientadas por IoT, operadores de frotas de transporte e consumidores em geral capturam mais de 90% das oportunidades econômicas geradas pelas aplicações de IoT [**8**]. Em muitos casos, esses clientes obtêm valor tanto de maneira direta, como pela automação e otimização de processos, quanto de maneira indireta, adquirindo máquinas mais eficientes, desenvolvidas com base em dados de IoT extraídos de produtos anteriores.

Ainda segundo a McKinsey (2023), para os fornecedores de tecnologia IoT, grande parte do valor criado deve ser direcionado a serviços e software, enquanto uma parcela menor será atribuída ao hardware. Este ambiente dinâmico exige que as organizações revisem continuamente suas estratégias, ajustando sua forma de competir e operar. Nesse contexto, escalar a transformação digital tornou-se uma necessidade.

As aplicações de IoT no setor B2B, segundo o estudo da McKinsey (2023), apresentam um potencial econômico maior do que no setor B2C. Apesar das aplicações voltadas ao consumidor final, como monitores de atividade física e sistemas de automação residencial, terem recebido maior atenção da mídia, o maior valor das tecnologias IoT está em aplicações empresariais. Setores como mineração, petróleo, gás e construção se beneficiam significativamente dessas soluções, mesmo que o impacto direto para os consumidores seja menor. Além disso, quando sistemas de IoT voltados ao consumo, como dispositivos de saúde conectados, são integrados a sistemas B2B, como os oferecidos por provedores de saúde e seguradoras, o potencial de valor aumenta consideravelmente [**8**].

3.1.2 Riscos e Desafios na IoT

Embora a Internet das Coisas (IoT) tenha revolucionado a maneira como dispositivos se conectam e interagem, proporcionando benefícios como eficiência operacional, automação e geração de insights, sua implementação não está isenta de desafios[**9**]. De acordo com estudos recentes, a crescente adoção de IoT expõe organizações e consumidores a uma série de riscos e limitações que devem ser cuidadosamente gerenciados para garantir o sucesso e a segurança das aplicações . Destaca-se os principais desafios enfrentados por implementações IoT, incluindo questões relacionadas à segurança, interoperabilidade e custos de implantação [**10**].

- **Riscos de segurança e privacidade**: com a difusão dos dispositivos de IoT, a segurança e a privacidade ganham cada vez mais importância. Muitos dispositivos de IoT são vulneráveis a hackers e outras ameaças cibernéticas, e isto pode comprometer a segurança e a privacidade dos dados sensíveis. Os dispositivos de IoT podem coletar também grandes quantidades de dados pessoais, gerando preocupações sobre a privacidade e a proteção dos dados .[**10**]

- **Problemas de interoperabilidade**: os dispositivos de IoT de diferentes fabricantes costumam usar padrões e protocolos diferentes, e isto dificulta o desempenho da comunicação chamada "máquina a máquina". Esta situação pode levar a problemas de interoperabilidade e criar silos de dados que são difíceis de integrar e analisar .[**10**]

- **Sobrecarga de dados**: os dispositivos de IoT geram grandes quantidades de dados, e isto pode sobrecarregar as empresas que não estão preparadas para lidar com isso. Analisar esses dados e extrair insights relevantes pode ser um grande desafio, especialmente para empresas que não contam com as ferramentas e os conhecimentos analíticos necessários.[**10**]

- **Custo e complexidade**: a implementação de um sistema de IoT pode ser cara e complexa, exigir grandes investimentos em hardware, software e infraestrutura. Gerenciar e manter um sistema de IoT também pode ser um desafio que exige qualificação e conhecimentos especializados .[**10**]

- **Desafios regulamentares e jurídicos**: à medida que os dispositivos de IoT se tornam mais difundidos, surgem desafios regulamentares e jurídicos. As empresas precisam estar em conformidade com várias regulamentações de proteção de dados, privacidade e cibersegurança, que podem variar de acordo com o país .[**10**]

3.2 Edge computing:

Edge computing ou computação de borda é um tipo de arquitetura de TI onde os dados do cliente são processados no limite da rede, ou o mais próximo possível da fonte de dados [**4**]. Com o processamento mais próximo, os usuários se beneficiam de serviços mais rápidos e confiáveis, enquanto as empresas usufruem da flexibilidade da cloud computing híbrida [**11**].

A edge computing é utilizada em vários setores, como o de telecomunicações, manufatura, transporte e serviços públicos. Os motivos que levam as pessoas a adotarem a edge computing incluem[**12**]:

- **Redução da latência e aumento da velocidade:** Colocar o poder de computação próximo à borda economiza tempo.

- **Segurança:** Os dados são analisados localmente e protegidos pela segurança de uma rede local ou pelo sistema fechado de um provedor de serviços.

- **Economia de custos:** As empresas podem otimizar o fluxo de dados para sistemas centrais e reter a maior parte dos dados brutos na borda, onde são úteis. Isso reduz a largura de banda e os custos .

- **Confiabilidade remota:** Dispositivos de borda armazenam e processam dados localmente, trabalhando com data centers de borda para superar problemas de conectividade intermitente .

- **Escalabilidade rápida:** A instalação de data centers de borda e dispositivos IoT pode permitir que as empresas escalem rapidamente suas operações .

Portanto, a edge computing desempenha um papel fundamental na habilitação de aplicações IoT, fornecendo processamento local, reduzindo a latência e aumentando a eficiência

e confiabilidade dos sistemas [**12**].

3.2.1 Edge computing na industria

Muitos casos de uso para a edge computing são baseados na necessidade de processar dados localmente e em tempo real, ou seja, situações onde a transmissão de dados a um datacenter para processamento causa níveis inaceitáveis de latência [**4**].

Para exemplificar o uso da edge computing para processar dados em tempo real, podemos pensar em uma fábrica moderna. No chão da fábrica, os sensores de Internet das Coisas (IoT) geram um fluxo contínuo de dados que pode ser usado para evitar interrupções e melhorar as operações. Estima-se que uma fábrica moderna com 2.000 equipamentos possa gerar 2.200 terabytes de dados por mês [**11**]. Portanto, é mais rápido e econômico processar esse volume de dados quando ele está mais próximo do equipamento, em vez de transmiti-lo para um centro de processamento de dados remoto. No entanto, é recomendável que o equipamento esteja conectado por meio de uma plataforma de dados centralizada. Assim, o equipamento pode, por exemplo, receber atualizações de software padronizadas e compartilhar dados filtrados que ajudam a melhorar as operações em outros locais da fábrica [**12**].

3.3 Machine Learning e Edge Computing

A aplicação de métodos e técnicas de Inteligência Artificial (IA) na borda (*edge computing*) é essencial para aumentar o desempenho e as capacidades de sistemas inteligentes e dispositivos da Internet das Coisas Industrial (IIoT) utilizados em processos de manufatura industrial. O conceito de processamento na borda com IA é frequentemente refletido na estrutura emergente de diferentes camadas de *edge computing*: **micro-edge**, **deep-edge** e **meta-edge**. Essas camadas representam um contínuo de processamento, inteligência e conectividade, integrando dispositivos de sensoriamento, processamento e comunicação próximos aos ativos industriais monitorados, gateways e controladores inteligentes, e dispositivos de computação de uso geral no local [**13**].

3.3.1 Camadas do Processamento na Borda

- **Micro-Edge:** Composto por dispositivos pequenos, como sensores e atuadores, equipados com microcontroladores (MCUs) baseados em núcleos Arm® Cortex®-M (*M0, M0+, M3, M4, M7*) ou arquiteturas abertas como RISC-V. Esses dispositivos incluem circuitos com memória, portas seriais, periféricos e capacidades sem fio, sendo projetados para realizar tarefas específicas de sistemas embarcados. Nos dispositivos *micro-edge*, o desenvolvimento de funcionalidades de IA é um processo desafiador, mas com grande potencial, especialmente em aplicações industriais. A implementação de modelos de aprendizado de máquina (ML) e deep learnig (DL) nesses dispositivos embarcados é vantajosa para operações de monitoramento de condições, manutenção preditiva (PdM) e manutenção prescritiva (PsM) em motores e equipamentos industriais [**13**].

- **Deep-Edge:** Consiste em gateways e controladores inteligentes que processam os dados de dispositivos *micro-edge*. Essa camada atua como um intermediário entre o processamento local e sistemas de análise mais amplos, oferecendo maior capacidade computacional e conectividade [**13**].

- **Meta-Edge:** Inclui dispositivos de computação no local (*on-premise*) usados para análises mais avançadas e processamento multiuso, proporcionando conectividade ampliada e integração com sistemas corporativos [**13**].

3.4 Machine Learning

Machine Learning (ML) desempenha um papel central em sistemas inteligentes modernos, permitindo que dispositivos processem dados localmente e identifiquem padrões de forma autônoma. No contexto deste projeto, ML foi utilizado para implementar um sistema de detecção de anomalias em sensores embarcados, otimizando o monitoramento em tempo real e reduzindo a dependência de processamento em nuvem [**14**].

3.4.1 NanoEdge™ AI Studio

O *NanoEdge™ AI Studio*, desenvolvido pela STMicroelectronics, é uma ferramenta projetada para criar modelos de **detecção de anomalias** em sistemas embarcados. Com foco em dispositivos de **recursos limitados**, como microcontroladores, ele permite a implementação de **algoritmos de machine learning embarcado**, processando dados **localmente e em tempo real**, sem depender de conectividade com a nuvem.

A ferramenta utiliza técnicas de aprendizado de máquina, predominantemente **não supervisionado**, para identificar padrões em dados normais e detectar desvios, mesmo sem a necessidade de dados rotulados sobre anomalias. Para isso, realiza um **benchmark** de diferentes modelos de aprendizado de máquina, testando sua eficiência para encontrar a abordagem mais adequada ao problema. Os principais paradigmas avaliados incluem:

- **Aprendizado Supervisionado**: Modelos treinados com dados rotulados aprendem a associar padrões de entrada (como sinais de vibração) com classes específicas de falha. Exemplos comuns incluem **Support Vector Machines (SVM), Random Forest (RF)** e **Multi-Layer Perceptron (MLP)**.

- **Aprendizado Não Supervisionado**: Utilizado para **detecção de anomalias**, onde os dados não possuem rótulos e o modelo identifica padrões desconhecidos ou atípicos. Durante a fase de **benchmark**, o *NanoEdge AI Studio* avalia diferentes modelos, como **Incremental Principal Component Analysis (IPCA), Iterated Conditional Modes (ICM), Minimum Message Length (MML), Gaussian Mixture Models (GMM)** e **k-Means Clustering**, selecionando a abordagem mais eficiente de acordo com os dados disponíveis [**15**].

Essas características tornam o *NanoEdge™ AI Studio* uma solução ideal para **manutenção preditiva (PdM)** e **monitoramento de condições** em ambientes industriais. Além disso, a ferramenta **automatiza o processo de seleção de algoritmos** e gera **bibliotecas otimizadas** que podem ser integradas diretamente ao firmware, simplificando o desenvolvimento e reduzindo a necessidade de conhecimento avançado em *Machine Learning* [**15**].

O *NanoEdge™ AI Studio* destaca-se por sua **facilidade de uso** e **recursos otimizados**, oferecendo vantagens significativas para o desenvolvimento de sistemas embarcados, tais como:

- **Otimização para Hardware de Recursos Limitados**: O software é projetado para criar algoritmos altamente eficientes, capazes de operar em microcontroladores com restrições de memória e processamento.
- **Automação do Processo de Desenvolvimento**: Automatiza tarefas como **seleção de algoritmos, validação e geração de bibliotecas**, reduzindo significativamente o tempo de desenvolvimento.
- **Foco em Detecção de Anomalias e Manutenção Preditiva**: Oferece suporte nativo para aplicações de **detecção de anomalias, monitoramento de condições** e **PdM**, tornando-o uma solução ideal para o escopo deste projeto.

3.4.2 Desenvolvimento do Modelo de Aprendizado

O desenvolvimento de um modelo modelo de machine learning segue o pipeline de aprendizado de máquina estruturado pelo NanoEdge™ AI Studio, composto pelas seguintes etapas:

- **Aquisição de Dados**: Coleta de sinais de vibração dos sensores MEMS em condições normais e anômalas.
- **Processamento de Dados**: Transformação dos dados brutos em características significativas, como normalização e análise no domínio da frequência utilizando FFT .
- **Seleção do Algoritmo**: Comparação automatizada de algoritmos candidatos, considerando as restrições de hardware e os requisitos de precisão.
- **Validação do Modelo**: Testes com novos dados para verificar o desempenho do modelo em cenários reais.
- **Geração da Biblioteca**: Geração de uma biblioteca otimizada para integração ao firmware do dispositivo.

3.5 Análise de Vibrações em Motores

A análise de vibrações é uma das técnicas mais eficazes de manutenção preditiva, sendo amplamente utilizada para diagnosticar e monitorar a condição de motores e outros equipamentos rotativos. Seu objetivo principal é identificar problemas mecânicos antes que eles se transformem em falhas catastróficas, aumentando a confiabilidade e reduzindo custos de manutenção. A técnica baseia-se na coleta, processamento e interpretação de sinais de vibração gerados por falhas específicas em componentes da máquina [**16**, **17**].

3.5.1 Princípios da Análise de Vibrações

A vibração de um motor é definida como o movimento oscilatório de suas partes, que pode ser causado por desalinhamento, desbalanceamento, folgas ou desgaste de componentes como rolamentos e engrenagens. Para monitorar e analisar essas vibrações, é necessário um sistema que inclua sensores de vibração, como acelerômetros, equipamentos de aquisição de dados, software de análise e computadores para processamento das informações coletadas [**16**].

De acordo com Mobley (2002), a análise de vibrações é considerada uma ferramenta essencial na manutenção preditiva devido à sua capacidade de [**17**]:

- Detectar falhas em seus estágios iniciais;
- Oferecer insights detalhados sobre o desempenho mecânico do equipamento;
- Melhorar o planejamento das atividades de manutenção, reduzindo o tempo de inatividade não planejado .

3.5.2 Faixas de Frequência e Tipos de Falhas

Cada tipo de falha em motores apresenta uma assinatura de frequência específica, o que permite que sejam diagnosticadas de forma precisa. Abaixo, descrevemos as principais falhas e as faixas de frequência associadas: [**16**]

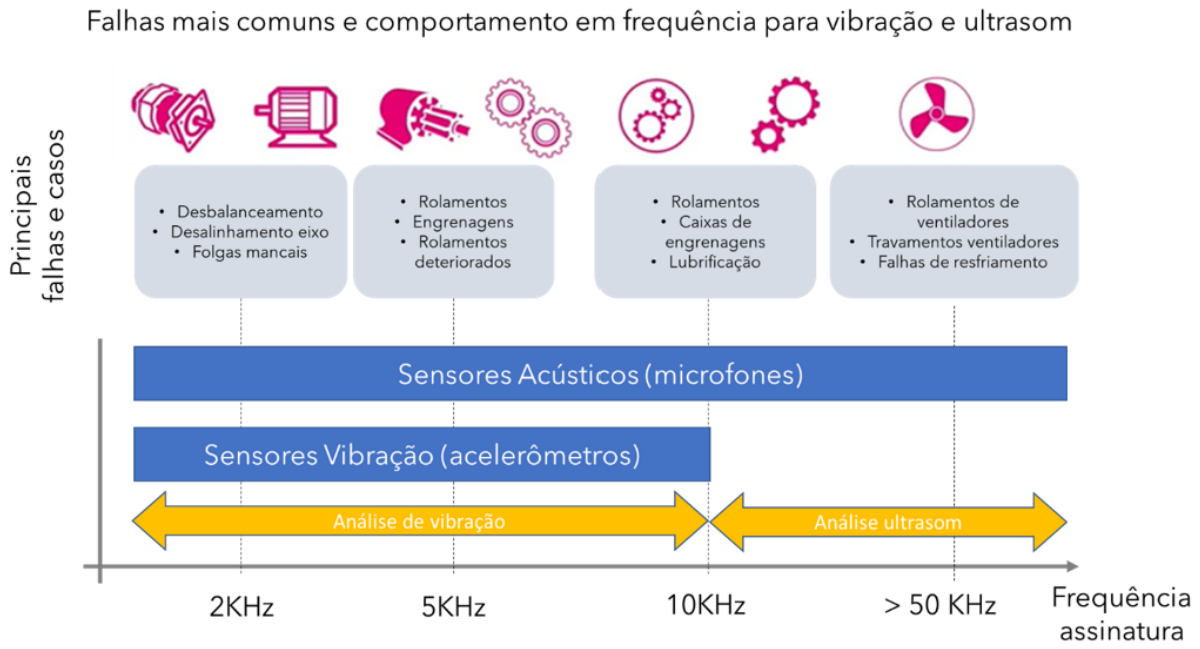


Figura 1: faixa de frequência de vibrações[**16**]

Baixa Frequência (< 2 kHz)

- **Falhas:** Desbalanceamento, desalinhamento de eixo e folgas em mancais.
- **Características:** Essas falhas geram vibrações suaves, geralmente detectadas em frequências baixas devido ao movimento irregular das partes rotativas.
- **Sensores Utilizados:** Sensores de vibração (acelerômetros) .

Frequência Moderada (2 kHz - 10 kHz)

- **Falhas:** Desgaste em rolamentos, danos em engrenagens e problemas iniciais de lubrificação.
- **Características:** Essas falhas apresentam vibrações localizadas e intermitentes, que podem ser identificadas por acelerômetros e análise de espectro .

Alta Frequência (> 10 kHz - 50 kHz)

- **Falhas:** Falhas críticas em engrenagens, lubrificação inadequada e desgaste severo de rolamentos.
- **Características:** Vibrações de alta frequência indicam níveis de atrito elevados e degradação acelerada de componentes. Sensores acústicos ou ultrassônicos são indicados para esta faixa .

Ultrafrequência (> 50 kHz)

- **Falhas:** Travamentos severos e falhas estruturais críticas.
- **Características:** Produzem ruídos ultrassônicos que exigem sensores acústicos especializados para serem detectados.

3.5.3 Benefícios da Análise de Vibrações

A análise de vibrações proporciona diversos benefícios, incluindo:[**16**] [**17**]

- **Detecção Precoce:** Identifica falhas em estágio inicial, permitindo correções antes que ocorram danos maiores ;

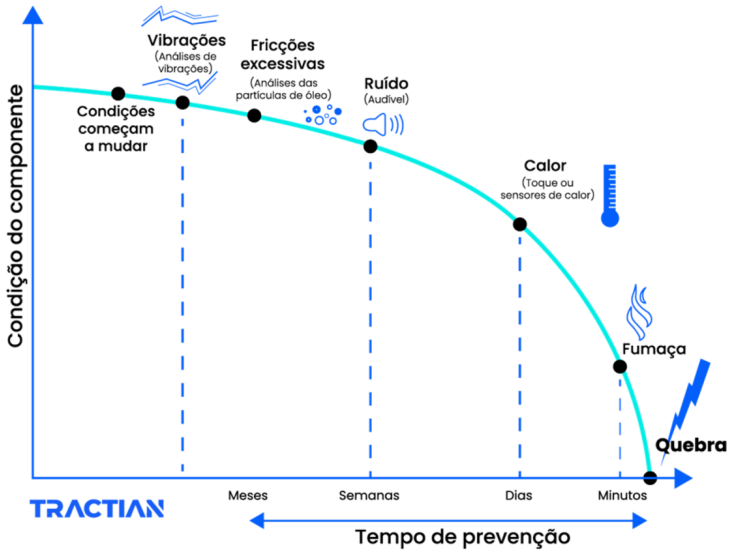


Figura 2: curva de falhas de máquinas[**18**]

- **Planejamento de Manutenção:** Facilita a programação de intervenções em momentos mais convenientes, reduzindo o impacto na produção;
- **Melhoria da Qualidade e Eficiência:** Reduz a reincidência de problemas mecânicos ao fornecer insights sobre condições críticas como desalinhamentos e desbalanceamentos.

Além disso, a Transformada Rápida de Fourier (FFT) desempenha um papel central ao decompor os sinais de vibração em componentes de frequência, possibilitando diagnósticos mais precisos de falhas complexas [**16**].

3.6 Bluetooth low energy:

A tecnologia Bluetooth é um sistema de comunicação sem fio de curto alcance, projetado para substituir cabos que conectam dispositivos eletrônicos portáteis e fixos. As principais características dessa tecnologia incluem robustez, baixo consumo de energia e baixo custo. A especificação do Bluetooth oferece flexibilidade através de funcionalidades opcionais, permitindo a diferenciação entre produtos. Existem dois tipos de sistemas Bluetooth: *Basic Rate* (BR) e *Low Energy* (LE). Ambos os sistemas incluem descobrimento de dispositivos, estabelecimento e mecanismos de conexão [**20**].

A especificação do Bluetooth Core Amended 5.4 traz atualizações significativas para o BLE, focando em melhorias de segurança, eficiência e novas funcionalidades para dispositivos em redes com muitos nós, como em soluções de malha (mesh). O padrão IEEE relacionado é o IEEE 802.15.1, que abrange tanto o Bluetooth Clássico quanto o BLE.

3.6.1 Camadas da Rede:

A arquitetura de rede BLE segue o modelo OSI, dividindo-se em três camadas principais[**19**]:

- **Camada Física (PHY):** A camada física do BLE (Bluetooth LE PHY) corresponde diretamente à camada física do modelo OSI. Esta camada define como os bits são transmitidos pelo meio físico, como frequências de operação e taxas de transmissão.
- **Camada de Enlace de Dados (Data Link Layer - DLL):** A camada de enlace no modelo OSI mapeia para os protocolos L2CAP (Logical Link Control and Adaptation Protocol)

e LL (Link Layer) no BLE, que gerenciam a ligação lógica e a comunicação entre dispositivos.

- **Camadas Superiores:** Nas camadas superiores da pilha Bluetooth LE, encontram-se serviços de aplicação, como perfis de aplicação e modos de operação dos dispositivos. Protocolos como GAP (Generic Access Profile), GATT (Generic Attribute Profile), ATT (Attribute Protocol) e SMP (Security Manager Protocol) desempenham papéis críticos no gerenciamento da conexão, modo de operação e segurança entre os dispositivos.

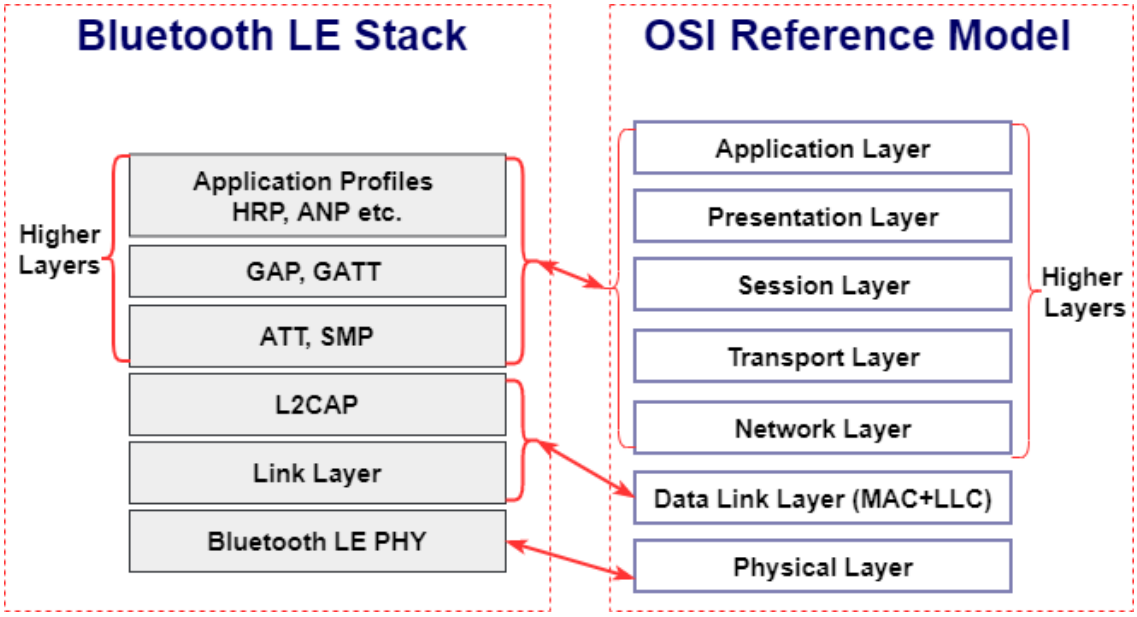


Figura 3: ilustra como as camadas do Bluetooth LE se alinham com o modelo OSI, demonstrando o mapeamento de funcionalidades de cada uma.[**19**]

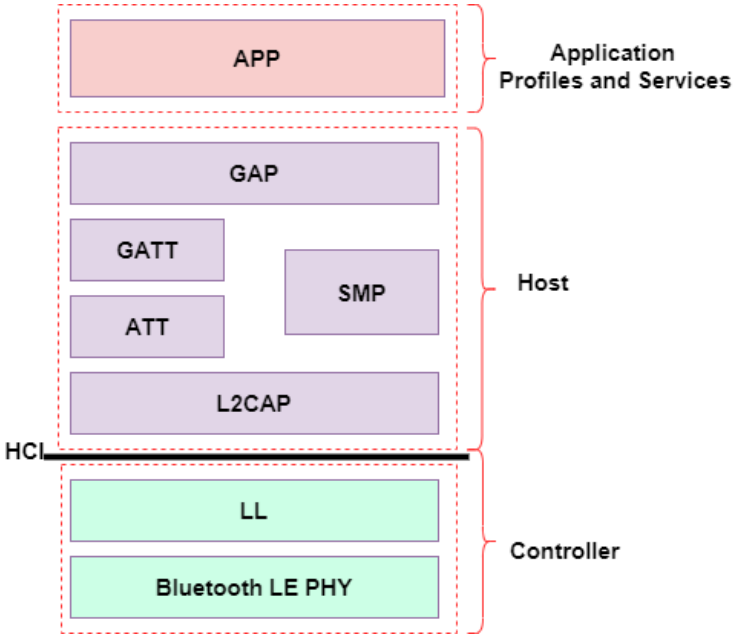


Figura 4: Pilha de protocolo Bluetooth LE[**19**]

A funcionalidade da pilha de protocolo Bluetooth LE é dividida entre três camadas principais: o controller, o host e application (Perfis e Serviços de Aplicativo)

Controller:

A camada do controlador compreende o Bluetooth LE PHY, o LL e a interface de controle do host (HCI) do lado do controlador[**20**].

1. **Bluetooth LE PHY:** A interface física de Bluetooth LE opera na mesma banda de frequência não licenciada de 2,4 GHz como o Wi-Fi®. Essa interface possui as seguintes características:

   - Opera em uma faixa de frequência de 2,4000 GHz a 2,4835 GHz.
   - A largura de banda do canal é de 2 MHz, com um total de 40 canais, numerados de 0 a 39.
   - Os pacotes de dados do usuário são transmitidos em canais na faixa de 0 a 36, enquanto os pacotes de publicidade são transmitidos nos canais 37, 38 e 39.
   - Usa um esquema de modulação conhecido como deslocamento de frequência gaussiano (GFSK) Para lidar com sobrecarga de dispositivos, o Bluetooth emprega a

técnica de frequency hopping spread spectrum (FHSS), alternando rapidamente entre 79 canais (no Bluetooth Clássico) ou 40 canais (no BLE)

2. **Link Layer (LL):** O Link Layer (LL) desempenha funções semelhantes à camada de controle de acesso ao meio (MAC) no modelo OSI. Ele interage diretamente com o Bluetooth LE PHY e gerencia o estado da conexão do rádio, determinando se um dispositivo atua como Central, Periférico, Anunciador ou Scanner.

3. **HCI do Lado do Controlador:** A Interface de Controle do Host (HCI) no lado do controlador é responsável pela comunicação entre o host (o dispositivo que utiliza o Bluetooth) e o controlador (a parte do dispositivo responsável pela comunicação Bluetooth).

Host:

O host compreende a HCI do lado do hospedeiro, L2CAP, protocolo de atributos (ATT), perfil de atributos genéricos (GATT), protocolo do gerenciador de segurança (SMP) e perfil de acesso genérico (GAP)[**20**].

1. **HCI do Lado do Hospedeiro:** Semelhante à HCI do lado do controlador, a HCI do lado do hospedeiro gerencia a comunicação entre o host e o controlador, mas neste caso, do ponto de vista do host.

2. **L2CAP:** O Protocolo de Controle de Acesso ao Link Lógico (L2CAP) encapsula dados das camadas superiores do Bluetooth LE no formato de pacote Bluetooth LE padrão para transmissão e também extrai dados desses pacotes na recepção.

3. **ATT:** O Protocolo de Transferência de Atributos (ATT) é responsável pela transferência de dados de atributos entre clientes e servidores em perfis baseados em GATT.

4. **GATT:** O Perfil de Atributos Genéricos (GATT) fornece uma estrutura de referência para todos os perfis baseados em GATT, coordenando a troca de perfis em uma conexão Bluetooth LE e definindo os papéis em uma arquitetura cliente-servidor.

5. **SMP:** O Protocolo do Gerenciador de Segurança (SMP) aplica algoritmos de segurança para criptografar e descriptografar os pacotes de dados, e define os papéis de iniciador e respondedor em uma conexão.

6. **GAP:** O Perfil de Acesso Genérico (GAP) especifica os papéis, modos e procedimentos de um dispositivo, incluindo o estabelecimento de conexão e a segurança.

Camada de Aplicativo (APP):

A camada de aplicativo é a interface direta do usuário, definindo perfis que permitem a interoperabilidade entre várias aplicações Bluetooth. Os fabricantes podem definir perfis proprietários para casos de uso não cobertos pelos perfis padrão.[**20**]

3.6.2 Rede mesh Bluetooth:

A rede Mesh Bluetooth é construída sobre a especificação Bluetooth Low Energy (LE), utilizando componentes como a **Link Layer** e a **Physical Layer** do Bluetooth LE. A interação entre o host e o controlador é realizada por meio do **Host Controller Interface (HCI)**, que define comandos e eventos padrão para essa comunicação [**20**].

A arquitetura da rede Mesh Bluetooth, conforme ilustrada na Figura 5, é estruturada em camadas e depende de componentes da especificação central do *Bluetooth Core Specification* (*LE Physical Transport*). Essa arquitetura consiste em duas pilhas principais: a **Mesh Networking Stack** e a **Mesh Provisioning Stack**, conforme descrito em [**20**].

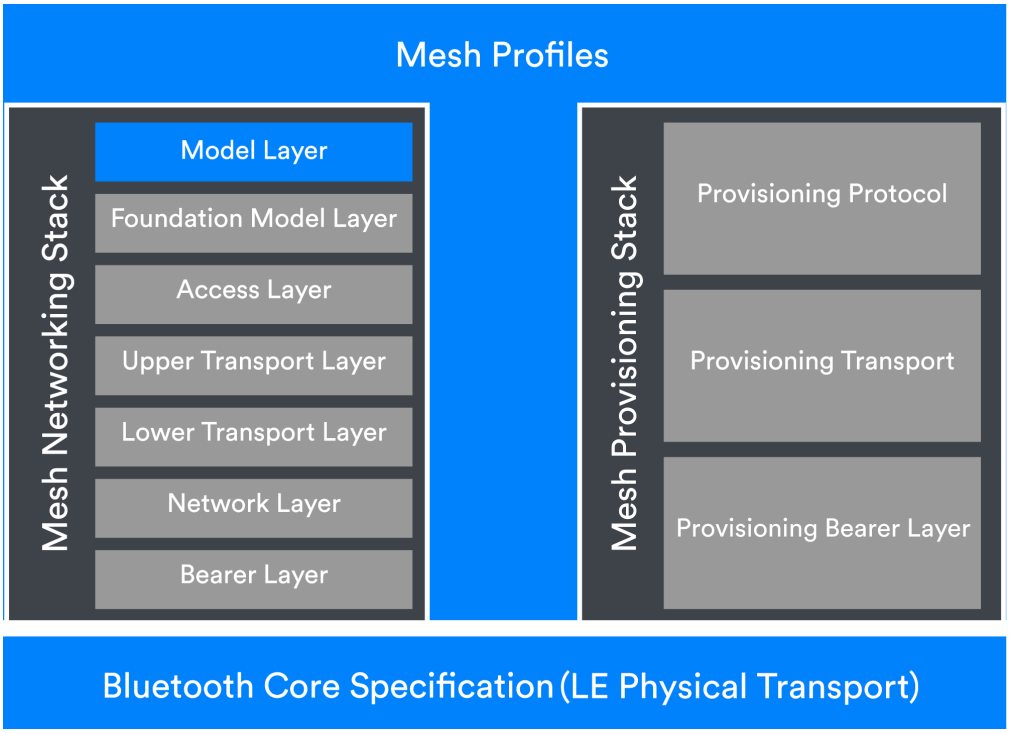


Figura 5: Arquitetura da rede Mesh Bluetooth

1. **Bearer Layer** A camada **Bearer Layer** é responsável por definir como as mensagens da rede Mesh (*Mesh PDUs*) que são transmitidas. Ela suporta dois tipos principais de transporte:

   - **Advertising Bearer**: Utiliza as funções de *advertising* e *scanning* do Bluetooth LE para enviar e receber mensagens da rede Mesh.
   - **GATT Bearer**: Permite que dispositivos que não suportam o *advertising bearer* se comuniquem indiretamente com os nós da rede Mesh. Isso é feito por meio do protocolo *Proxy*, encapsulado em operações GATT (*Generic Attribute Profile*), utilizando o serviço chamado **Mesh Proxy Service**.

   Um nó proxy suporta ambos os bearers, permitindo a conversão e retransmissão de mensagens entre os dois tipos.

2. **Network Layer** A camada **Network Layer** gerencia o roteamento de mensagens na rede Mesh, incluindo:

- **Definição de Endereços**: Define os diferentes tipos de endereços para as mensagens.
- **Filtros de Entrada e Saída**: Controla se as mensagens recebidas da camada bearer devem ser processadas ou descartadas.
- **Suporte a Múltiplos Bearers**: Suporta múltiplas interfaces de rede, incluindo interfaces locais para comunicação dentro do mesmo nó.

Essa camada também implementa as funcionalidades de **Relay** e **Proxy**, garantindo que as mensagens sejam entregues eficientemente entre os nós.

3. **Lower Transport Layer** A **Lower Transport Layer** é responsável pela segmentação e remontagem de mensagens (*PDUs*):

   - **Segmentação**: Divide mensagens longas em pacotes menores que podem ser transmitidos pela rede.
   - **Remontagem**: Recompõe os pacotes segmentados recebidos em uma única mensagem, que é passada para as camadas superiores.

   Isso permite a transmissão de mensagens maiores que o tamanho máximo de um único *PDU*.

4. **Upper Transport Layer** A camada **Upper Transport Layer** cuida da segurança e controle das mensagens:

   - **Encriptação e Autenticação**: Garante que os dados sejam transmitidos de forma segura.
   - **Mensagens de Controle**: Gera e gerencia mensagens internas entre os nós da rede, como aquelas relacionadas à funcionalidade *Friendship*, encaminhamento direcionado (*Directed Forwarding*) e monitoramento de batimentos (*Heartbeats*).

5. **Access Layer** A camada **Access Layer** define como os modelos da aplicação interagem com a camada de transporte superior. Suas principais responsabilidades incluem:

   - **Formatação de Dados**: Define o formato dos dados das mensagens.

- **Controle de Encriptação**: Gerencia o processo de encriptação e desencriptação dos dados.
- **Validação de Mensagens**: Verifica se os dados recebidos pertencem à rede e ao modelo correto antes de encaminhá-los.

6. **Foundation Model Layer** A camada **Foundation Model Layer** implementa os modelos responsáveis pela configuração e gerenciamento da rede Mesh. Ela garante que a rede seja mantida e configurada adequadamente para operar de forma eficiente.

7. **Model Layer** A camada **Model Layer** implementa os modelos da aplicação. Esses modelos definem comportamentos, mensagens, estados e suas associações (*bindings*). Eles são responsáveis pela lógica de aplicação específica da rede Mesh.

3.7 Protocolos de comunicação Utilizados

3.7.1 I2C (Inter-Integrated Circuit)

O **I2C**(Inter-Integrated Circuit) é um protocolo de comunicação serial síncrona amplamente utilizado para conectar periféricos de baixa velocidade a microcontroladores. Desenvolvido pela Philips na década de 1980, ele se destaca por sua simplicidade de implementação, utilizando apenas duas linhas: uma para dados (SDA) e outra para clock (SCL). Este protocolo permite a comunicação entre múltiplos dispositivos conectados ao mesmo barramento, possibilitando que um único microcontrolador controle diversos sensores, memórias ou displays [**21**].

Aplicação no Projeto: Foi utilizado na comunicação entre o sensor de temperatura MEMS, o I2C é uma escolha ideal por consumir poucos pinos do microcontrolador, ser eficiente em termos de energia e suportar comunicação com múltiplos dispositivos no mesmo barramento .

3.7.2 SPI (Serial Peripheral Interface)

O **SPI** O SPI (Serial Peripheral Interface) é outro protocolo de comunicação serial síncrona, porém, diferente do I²C, o SPI utiliza quatro linhas para comunicação: MOSI (Master Out Slave In), MISO (Master In Slave Out), SCK (Serial Clock) e SS (Slave Select). Este protocolo é amplamente usado para comunicação de alta velocidade entre microcontroladores e periféricos, sendo preferido em aplicações que exigem transferências rápidas de dados e uma maior flexibilidade na comunicação ponto a ponto [**22**].

Aplicação no Projeto: Como a leitura do sensor de aceleração, necessita de altas taxas de frequencia de amostragem,foi selecionado o protcolo SPI para a comunicação.

3.7.3 J-Link (Programador/Gravador)

O **J-Link** é uma interface de depuração e programação amplamente utilizada em sistemas embarcados, especialmente para microcontroladores baseados na arquitetura ARM, como a série Cortex-M. Fabricado pela SEGGER, o J-Link oferece suporte tanto para a interface SWD (Serial Wire Debug) quanto para JTAG, permitindo carregar firmware no microcontrolador, monitorar a execução do código e realizar depuração em tempo real. Esta ferramenta é valorizada pela sua eficiência e robustez em tarefas de desenvolvimento e depuração [**23**]

Aplicação no Projeto: Essa ferramenta é essencial para o desenvolvimento de sistemas embarcados, permitindo gravar o código de forma rápida e confiável, além de oferecer recursos avançados de depuração, como monitoramento de registros, breakpoints e análise em tempo real, facilitando o processo de desenvolvimento e diagnóstico de problemas no sistema embarcado.

3.7.4 UART (Universal Asynchronous Receiver-Transmitter)

A **UART** (Universal Asynchronous Receiver-Transmitter) é um dos protocolos de comunicação mais utilizados em dispositivos para transferência de dados de forma serial. A comunicação serial ocorre bit a bit através de uma linha ou fio, sendo que, em uma comunicação bidirecional, são necessárias duas linhas para transmissão e recepção de dados. No contexto de sistemas embarcados, microcontroladores e computadores, o UART desempenha

um papel fundamental na comunicação entre dispositivos, devido à sua simplicidade e ao uso reduzido de fios, o que minimiza os custos de implementação [**24**].

O funcionamento do UART se baseia em duas linhas principais: **TX** (Transmitter) para enviar dados e **RX** (Receiver) para recebê-los. Diferentemente de outros protocolos, como SPI ou I2C, a UART é assíncrona, o que significa que não necessita de um clock compartilhado entre os dispositivos. Em vez disso, os dispositivos se sincronizam por meio de bits de início e parada, comumente conhecidos como start bit e stop bit, permitindo a transmissão correta dos dados. Entre os bits de início e de parada, os dados são transmitidos geralmente em pacotes de 8 bits, mas podem variar dependendo da configuração [**24**].

Um dos benefícios da comunicação UART é sua adaptabilidade a diferentes tipos de protocolos seriais, permitindo a comunicação entre diferentes tipos de dispositivos. No entanto, apesar de sua ampla utilização, o protocolo UART não é otimizado em todos os casos. A implementação inadequada de frames de comunicação pode levar a problemas de sincronização e perda de dados. Por isso, é fundamental que o protocolo seja configurado corretamente dentro do microcontrolador para garantir uma troca de dados eficiente e sem falhas [**24**].

Aplicação no Projeto:A comunicação UART foi utilizada para enviar e receber dados de depuração, logs e comandos.

3.7.5 USB (Universal Serial Bus)

O **USB** (Universal Serial Bus) é um protocolo de comunicação amplamente adotado para conectar dispositivos periféricos a computadores e outros dispositivos centrais. A especificação USB, publicada inicialmente em 1996, evoluiu ao longo dos anos para atender às demandas crescentes por maior largura de banda e conectores mais eficientes. A versão 2.0 da especificação USB foi publicada em 2000, introduzindo melhorias significativas em termos de desempenho e capacidade de transmissão de dados [**23**].

A arquitetura USB é baseada em um modelo mestre-escravo, onde o computador (host) controla a comunicação, iniciando as transações de dados. O USB opera com quatro linhas principais: VBUS (alimentação), GND (terra), D+ e D- (dados). Essas linhas permitem a comunicação diferencial e bidirecional, proporcionando uma comunicação eficiente e confiável [**25**].

Aplicação no Projeto: Na programação e depuração, o USB permite que o J-Link seja reconhecido pelo computador, facilitando a gravação do firmware e a depuração. Além disso, o USB é utilizado como meio de comunicação serial, através de adaptadores USB-UART, que convertem os sinais UART do microcontrolador para o padrão USB, permitindo que o PC receba os dados, visualize logs e interaja diretamente com o firmware em execução. A alta taxa de transferência e a ampla compatibilidade tornam o USB uma escolha confiável para conexões de curto alcance em sistemas embarcados.

3.7.6 Comunicação NFC

A **Near Field Communication** (NFC) é uma tecnologia de comunicação sem fio de curto alcance projetada para facilitar a troca de dados de maneira rápida e segura entre dispositivos. Operando na frequência de 13,56 MHz, a NFC utiliza modulação por amplitude (AM) para transmitir informações em distâncias de até 10 cm. Embora compartilhe fundamentos tecnológicos com o RFID (Radio-Frequency Identification), a NFC se diferencia por oferecer comunicação bidirecional, permitindo que dispositivos atuem tanto como transmissores quanto como receptores. Além disso, a NFC suporta múltiplos modos de operação, ampliando sua versatilidade em aplicações como pagamentos por aproximação, autenticação e transferência de arquivos [**25**].

1. **Modos de Operação:**
    - **Leitura/Gravação:** Permite que um dispositivo leia e escreva dados em etiquetas NFC.
    - **Modo Tag:** Um dispositivo NFC se comporta como um cartão de crédito ou identificação.
    - **Peer-to-Peer:** Dispositivos NFC trocam dados diretamente entre si.
2. **Segurança e Interoperabilidade:** Devido ao curto alcance, NFC é intrinsecamente segura. O protocolo suporta criptografia e autenticação para maior proteção dos dados. A interoperabilidade é garantida por padrões que permitem a comunicação entre dispositivos de diferentes fabricantes.[**26**]

3.7.7 Protocolo NDEF

O **NFC Forum** é responsável pela definição e desenvolvimento de protocolos relacionados à comunicação por NFC (Near Field Communication). Entre esses protocolos, destaca-se o **NFC Data Exchange Format (NDEF)** e seu tipo específico, o **NDEF Text** [**26**].

O NDEF é um formato padronizado para a troca de dados entre dispositivos NFC. Ele organiza as informações em mensagens compostas por registros, cada um contendo um cabeçalho e um corpo de dados. O tipo de registro **NDEF Text**, desenvolvido pelo NFC Forum, é utilizado para armazenar e transmitir texto entre dispositivos NFC. Este formato permite a comunicação eficiente e interoperável de informações textuais [**25**].

Além disso, o NFC Forum define várias classes de dispositivos NFC, cada uma com características e protocolos específicos. O protocolo NDEF Text é compatível com essas classes, permitindo que dispositivos de diferentes categorias, como leitores, tags e dispositivos móveis, troquem informações de forma eficiente

Aplicação no Projeto : Para o projeto, a comunicação NFC com o protocolo NDEF Text foi escolhida para gravar informações dos sensores na memória. Essa abordagem permite uma integração eficaz e uma comunicação prática para o monitoramento e configuração do sensor no sistema.

3.8 Microcontroladores e Programação de Software Embarcado

Os microcontroladores desempenham um papel crucial em sistemas embarcados, fornecendo uma plataforma de processamento dedicada para controlar e coordenar as operações de dispositivos eletrônicos. Desde suas origens até os dias de hoje, eles têm sido a espinha dorsal de uma ampla gama de aplicações, desde os primeiros sistemas embarcados em eletrodomésticos até os mais avançados sistemas de controle em aeronaves e veículos espaciais [**27**].

A arquitetura ARM (Advanced RISC Machine) tem desempenhado um papel significativo na evolução dos microcontroladores. Desenvolvida inicialmente pela Acorn Computers na década de 1980, a arquitetura ARM é baseada em uma abordagem RISC (Reduced Instruction Set Computing), que se concentra na simplicidade e eficiência das instruções do

processador [**28**]. A arquitetura ARM é caracterizada por um conjunto reduzido de instruções altamente otimizadas, resultando em um desempenho eficiente e um consumo de energia mais baixo em comparação com arquiteturas mais complexas. Isso permite que os microcontroladores ARM ofereçam um equilíbrio entre desempenho, consumo de energia e custo, tornando-os uma escolha popular em uma variedade de aplicações [**27**].

Os microcontroladores geralmente incluem uma variedade de recursos integrados para facilitar o desenvolvimento de sistemas embarcados. Estes recursos podem incluir:

- **1. Unidade Central de Processamento (CPU):** Responsável pela execução de instruções e operações de processamento.
- **2. Memória:** RAM (Random Access Memory) para armazenamento temporário de dados e ROM/Flash para armazenamento de código de programa.
- **3. Periféricos de Entrada/Saída (GPIO):** Interfaces para conexão com dispositivos externos, como sensores, atuadores e dispositivos de comunicação.
- **4. Interfaces de Comunicação:** UART (Universal Asynchronous Receiver-Transmitter), I2C (Inter-Integrated Circuit), SPI (Serial Peripheral Interface) e USB (Universal Serial Bus) para comunicação com outros dispositivos.
- **5. Temporizadores e Contadores:** Para operações de temporização e geração de pulsos.
- **6. Conversores Analógico-Digital (ADC) e Digital-Analógico (DAC):** Para converter sinais analógicos em digitais e vice-versa.
- **7. Watchdog Timer:** Para monitorar o funcionamento do sistema e reiniciá-lo em caso de falhas.
- **8. Interfaces de Periféricos Específicos:** Como PWM (Pulse Width Modulation) para controle de motores e geradores de sinais.

Além disso, os microcontroladores ARM geralmente oferecem recursos avançados de economia de energia, como modos de baixo consumo e capacidade de desligar periféricos não utilizados para prolongar a vida útil da bateria em dispositivos alimentados por energia [**28**].

A programação de software embarcado refere-se ao desenvolvimento de programas e algoritmos que são executados diretamente em um microcontrolador ou em sistemas embarcados semelhantes. Este tipo de software é projetado para atender a requisitos específicos de hardware e oferecer um desempenho otimizado para uma aplicação específica. Linguagens de programação comuns para desenvolvimento de software embarcado incluem C e Assembly [**27**].

Aplicação no Projeto: A compreensão dos microcontroladores, da arquitetura ARM e da programação de software embarcado é uma otima solução para o desenvolvimento de sistemas IoT. No contexto deste projeto, será utilizado o microcontrolador nRF52840 da Nordic Semiconductor, que aproveita os recursos avançados da arquitetura ARM e oferece uma solução eficiente em termos de energia para aplicações de IoT. A programação de software embarcado será realizada utilizando linguagens de programação como C e utilizando as ferramentas de desenvolvimento fornecidas pela Nordic Semiconductor para desenvolvimento, depuração e simulação do código. Isso garantirá um desenvolvimento eficiente e robusto do sistema de monitoramento de temperatura com rede mesh Bluetooth.

4 Metodologia

Este projeto foi desenvolvido seguindo uma abordagem sistemática de engenharia de produto, organizada em etapas interdependentes. Essas etapas englobaram o desenvolvimento de hardware, firmware, machine learning embarcado, case mecânico e análise de mercado. Cada fase foi planejada e executada com cuidado, assegurando a integração harmônica entre os componentes eletrônicos, o software embarcado e a estrutura mecânica que abriga o dispositivo.

Para viabilizar a prática deste projeto, utilizou-se inicialmente uma placa de sensor de temperatura e aceleração, equipada com um microcontrolador Bluetooth, fornecida pela empresa Pullup Soluções em Sistemas Eletrônicos. Embora esta placa não atendesse completamente aos requisitos do projeto final, ela foi crucial como referência para o desenvolvimento do novo protótipo, permitindo identificar e corrigir limitações importantes. A placa demonstrou-se ideal para os testes, pois utiliza um sensor de temperatura , aceleração e um microcontrolador da mesma família da Nordic Semiconductor planejado para o projeto,mas com menor capacidade de memória, o que facilitou a validação das funcionalidades propostas. A contribuição da Pullup foi essencial para viabilizar financeiramente a demonstração prática do projeto, uma vez que o desenvolvimento completo do hardware não era possível neste momento devido a restrições orçamentárias.

4.1 Desenvolvimento do Hardware

O desenvolvimento do hardware seguiu uma metodologia estruturada, iniciando com o levantamento de requisitos. Nessa fase, foram estabelecidas as especificações críticas do sensor, como baixo consumo de energia, tamanho compacto e compatibilidade com comunicação via rede Bluetooth Mesh. Durante a análise da placa fornecida pela Pullup, foram identificadas limitações importantes:

- **Baixa taxa de amostragem no sensor de aceleração**, inadequada para análise de vibra-

ção.

- **Microcontrolador com memória insuficiente**, inviabilizando a implementação simultânea de rede Mesh Bluetooth e algoritmos de machine learning.

Com base nessas limitações, foi decidido substituir esses componentes no novo projeto. Utilizando o software **Altium Designer** (versão estudantil), foram realizados o esquemático e o layout da PCB, respeitando requisitos de design, como espaçamento de trilhas, controle de impedância e eficiência energética. Foi utilizado a ferramenta do Altium Designer que verifica falhas no projeto que é chamada de DRC (Design Rule Check) foram conduzidas para validar o design antes da geração de arquivos de produção. No entanto, por limitações financeiras, a nova placa projetada não foi fabricada.

4.2 Desenvolvimento de Firmware

O desenvolvimento do firmware foi planejado e implementado de forma estruturada, considerando as necessidades específicas do projeto e as limitações do hardware disponível. Foram desenvolvidos quatro firmwares distintos, cada um com funcionalidades específicas para atender aos requisitos de comunicação Bluetooth Mesh, coleta de dados e treinamento de modelos de machine learning. A implementação utilizou o *Nordic SDK Mesh* na IDE *Segger Embedded Studio*, com programação em linguagem *C*.

Firmware para Sensor (Bluetooth Mesh - Servidor)

O primeiro firmware foi projetado para os sensores que coletam informações de temperatura e aceleração. Esses sensores utilizam uma rede *Bluetooth Mesh* para transmitir os dados capturados. Foi implementado um modelo proprietário chamado **Sensor Model Servidor**, que:

- Coleta dados de sensores MEMS via comunicação *I2C*.
- Transmite os dados capturados, incluindo valores de temperatura e aceleração, para outros dispositivos na rede Mesh.

- Implementa uma máquina de estados para gerenciar os ciclos de coleta, processamento e envio de dados, otimizando o consumo de energia.

Adicionalmente, este firmware incorpora o modelo **Light Switch**, uma aplicação *Bluetooth Mesh* padrão adaptada para este projeto, permitindo ligar e desligar LEDs no sensor remotamente. Este modelo foi utilizado como teste funcional da comunicação na rede Mesh.

Firmware para Gateway (Bluetooth Mesh - Cliente)

O segundo firmware foi desenvolvido para o gateway, que atua como ponto central de comunicação entre os sensores e o computador. Este firmware, baseado no modelo proprietário **Sensor Model Cliente**, possui as seguintes características:

- Recebe dados transmitidos pelos sensores na rede Mesh, incluindo informações de temperatura e aceleração.
- Realiza o roteamento dos dados para o computador via USB, onde são exibidos no terminal.
- Implementa lógica de recepção e validação de pacotes, garantindo que os dados provenientes do *Sensor Model Servidor* sejam processados corretamente.

Firmware para Treinamento de Machine Learning (Servidor)

O terceiro firmware foi desenvolvido para sensores que capturam dados brutos, sem utilizar a funcionalidade Mesh, e transmitem diretamente para o gateway em um formato compatível com o *NanoEdge AI Studio*. As principais características incluem:

- Coleta de dados de vibração e temperatura em cenários normais e anômalos.
- Transmissão dos dados em um formato padronizado para treinamento de modelos de machine learning.
- Simulação de diferentes condições, como desbalanceamento e obstrução de fluxo, para capturar padrões representativos para treinamento.

Firmware para Gateway de Treinamento de Machine Learning (Cliente)

O quarto firmware foi desenvolvido para o gateway que recebe os dados transmitidos pelo firmware servidor de treinamento. Este firmware processa e armazena os dados no formato compatível com o *NanoEdge AI Studio*, permitindo seu uso direto no treinamento dos modelos. Suas características incluem:

- Recepção de dados brutos transmitidos pelo sensor de treinamento.
- Formatação e encaminhamento dos dados ao computador via USB, para que sejam utilizados no software de treinamento de machine learning.
- Implementação de lógica para evitar perdas de pacotes e garantir a qualidade dos dados coletados.

Firmware para Execução do Modelo de Machine Learning

Por fim, o quinto firmware foi projetado para embarcar o modelo de machine learning treinado e validar sua funcionalidade em tempo real. Este firmware:

- Recebe os dados de vibração e temperatura do sensor MEMS.
- Processa os dados localmente utilizando a biblioteca gerada pelo *NanoEdge AI Studio*.
- Retorna os resultados de similaridade e temperatura processados diretamente ao gateway.

Devido às limitações de memória do microcontrolador utilizado, este firmware não inclui a funcionalidade de comunicação via rede Mesh. A integração com a rede Mesh será possível em versões futuras, desde que seja utilizado um microcontrolador com maior capacidade de RAM e Flash.

4.3 Machine Learning Embarcado

O desenvolvimento do sistema de machine learning embarcado foi planejado com base em um pipeline bem definido, utilizando o *NanoEdge AI Studio* para criação, ajuste e execução dos modelos. O objetivo principal foi desenvolver modelos capazes de identificar sinais

anômalos em cenários reais e simulados, respeitando as limitações de memória e processamento do hardware embarcado. Este pipeline abrangeu quatro etapas principais: coleta de dados, ajuste de dados, treinamento de modelos e validação empírica.

4.3.1 Pipeline do NanoEdge AI Studio

O processo de desenvolvimento com o *NanoEdge AI Studio* pode ser descrito pelas seguintes etapas fundamentais:

1. Coleta e pré-processamento dos dados: Os dados brutos de vibração e temperatura foram capturados em diferentes cenários utilizando sensores MEMS, como ilustrado nas Figuras 6 e 7.

2. Envio dos dados ao *NanoEdge AI Studio*: Os dados foram transmitidos pelo firmware do sensor (Seção 4.2) ao gateway conectado ao computador, onde o software NanoEdge gerenciou a importação e análise inicial.

3. Seleção automática de modelos: O *NanoEdge AI Studio* realizou uma busca automática para encontrar o modelo de machine learning mais adequado às características dos dados e às restrições de hardware. Essa seleção foi baseada no benchmark apresentado na Figura 12.

4. Geração da biblioteca otimizada: Após a seleção do modelo, o software gerou uma biblioteca específica, pronta para ser embutida no firmware do dispositivo.

4.3.2 Coleta de Dados

Os dados foram coletados a partir de uma ventoinha equipada com sensores MEMS de vibração, posicionados em diversas configurações. Os cenários simulados para captura de dados incluíram:

Figura 6: ventoinha com sensor acoplado na vertical

Figura 7: ventoinha com sensor acoplado na horizontal

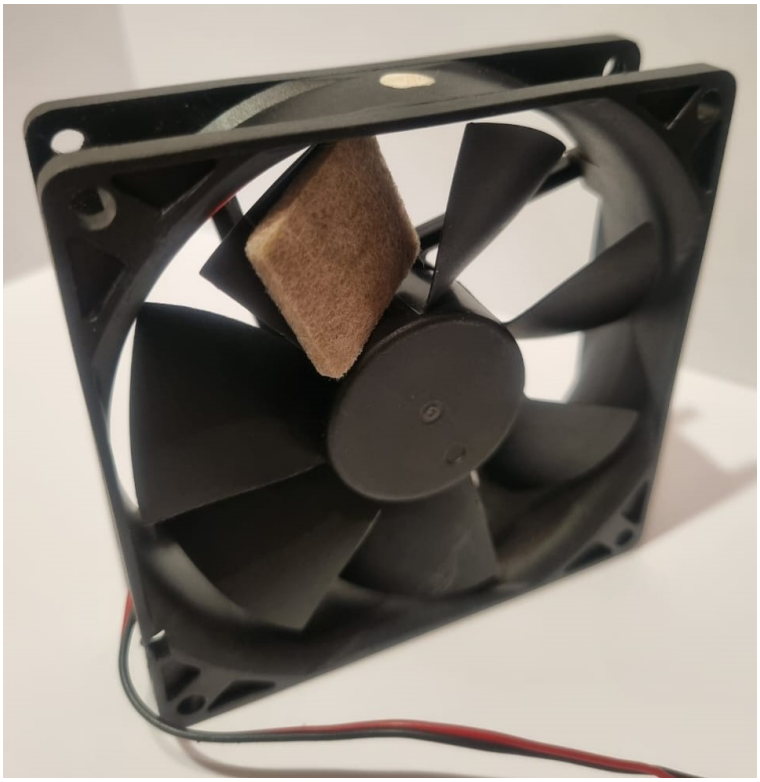


Figura 8: ventoinha com sensor acoplado na vertical com desbalnaceamento

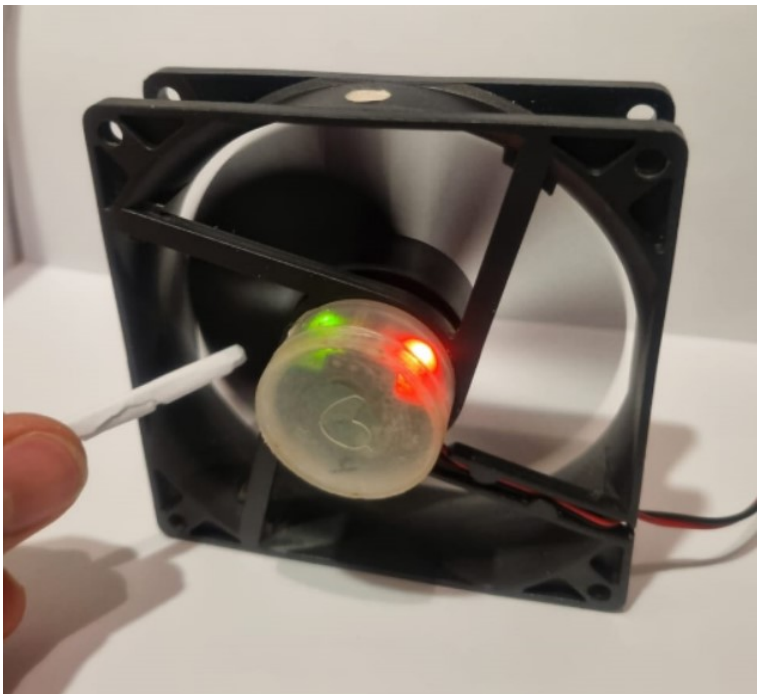


Figura 9: ventoinha com sensor acoplado na vertical com interferência

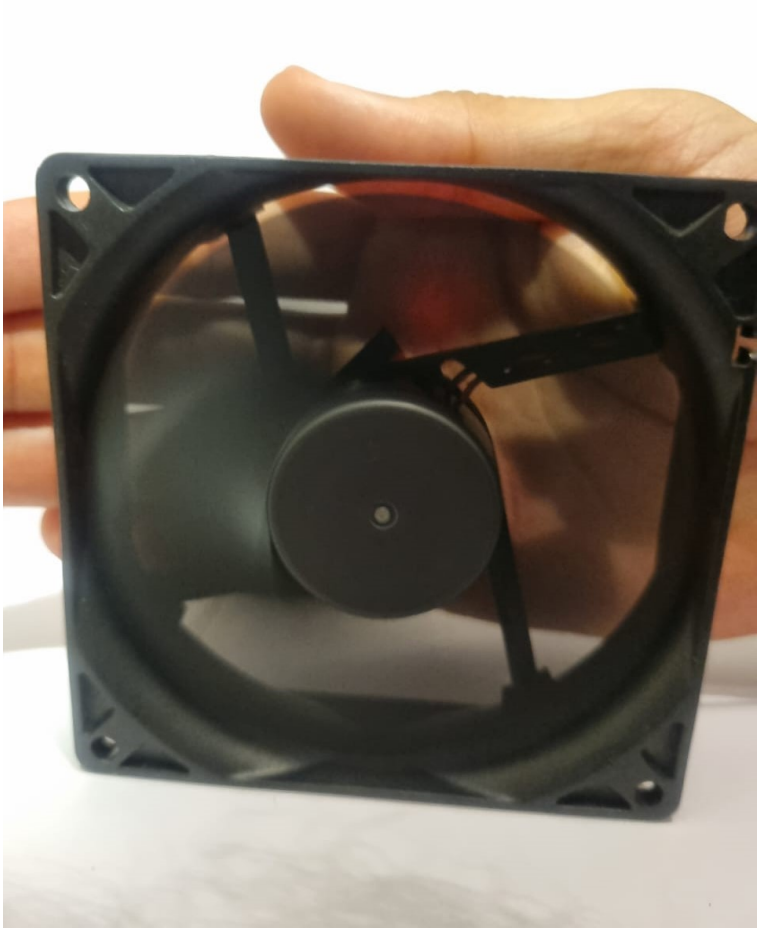


Figura 10: ventoinha com sensor acoplado na vertical com fluxo de ar obstruído

Os dados capturados foram registrados e segmentados em janelas temporais para posterior processamento no *NanoEdge AI Studio*. Essas janelas permitiram analisar variações nas vibrações ao longo do tempo, facilitando a detecção de padrões anômalos.

4.3.3 Ajuste de Dados

O ajuste dos dados foi uma etapa essencial para garantir que estivessem no formato correto e otimizados para o treinamento dos modelos de machine learning. Durante esta fase, foram aplicadas as seguintes técnicas:

- Janelamento de dados: Diferentes tamanhos de janelas temporais foram testados, para otimizar a análise de vibrações e equilibrar precisão e consumo de memória.
- Normalização: Os dados foram padronizados para garantir que estivessem em escalas

adequadas para o processamento pelo *NanoEdge AI Studio*, minimizando viés de escala nos resultados como por exemplo configuração de taxa de amostragem e numero de bits dos dados.

- Análise de desempenho no hardware: Após os ajustes, verificou-se que os dados processados podiam ser utilizados em tempo real no microcontrolador, sem exceder os limites de RAM e Flash disponíveis.

- Adequação do gateway para o nano edge receber os dados pela porta COM: Foi utilizado o firware descrito na (Seção 4.2) conforme ilustra a Figura 11

Figura 11: nano edge recebendo dados de aceleração pela porta COM

4.3.4 Treinamento de Modelos

O treinamento dos modelos foi realizado no *NanoEdge AI Studio* utilizando os dados coletados nos cenários simulados. Durante o processo:

- Diferentes algoritmos de machine learning sugeridos pelo software foram testados para encontrar o modelo que melhor atendesse às restrições do dispositivo embarcado.
- O software executou benchmarks automáticos (Figura 12) para avaliar a precisão, consumo de memória e tempo de execução de cada modelo.

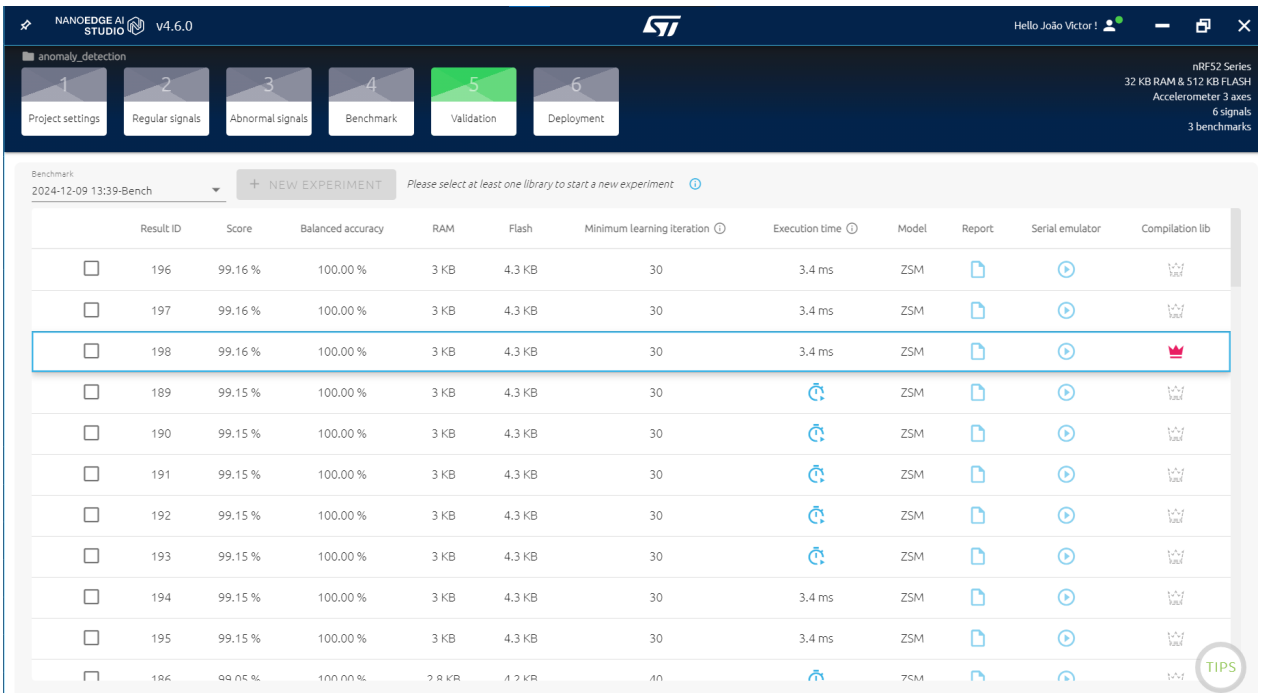


Figura 12: benchmark gerado pelo nano edge

- Após a escolha do modelo ideal, foi gerada uma biblioteca otimizada para ser integrada ao firmware do dispositivo.

4.3.5 Validação Empírica

A validação foi realizada primeiramente no simulador do *NanoEdge AI Studio* com a comunicação com a porta COM do computador conforme pode ser visto na imagem abaixo:

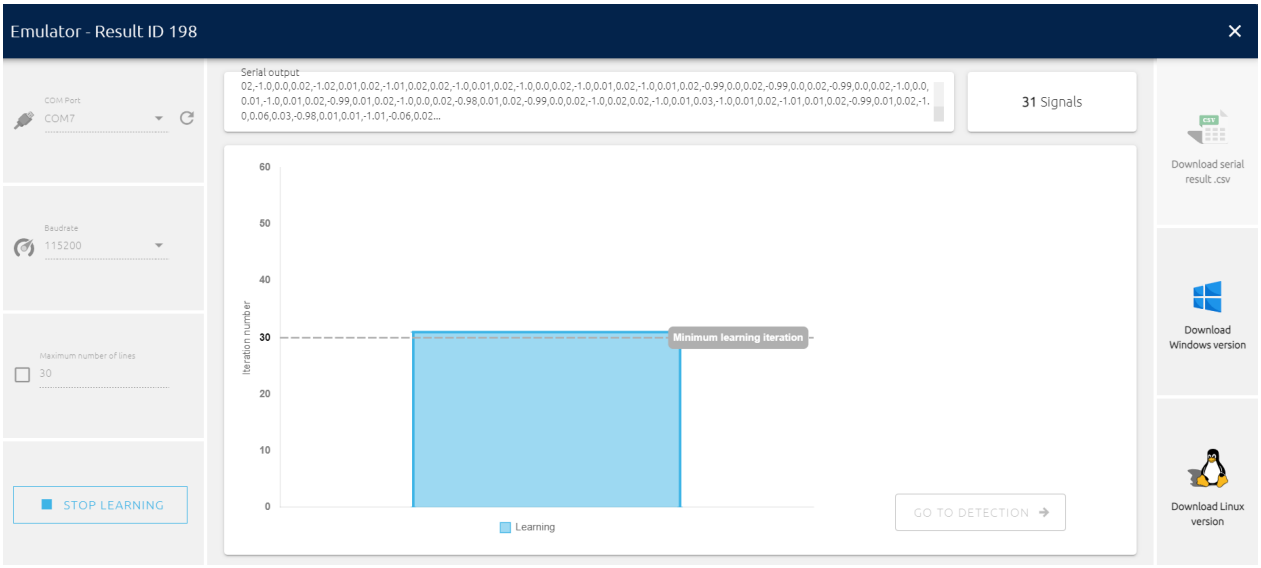


Figura 13: Emulador de modelo do nano edge com sinais de acelereação recebidos pela porta COM

Quanto no hardware embarcado utilizando o *Firmware para Execução do Modelo de Machine Learning*(Seção 4.2) e o Gateway bluetooth recebendo os dados de aceleração dos eixos X , Y , Z e os dados calulados pelo microcontrolador do modelo de similaridade do sinal em porcentagem..

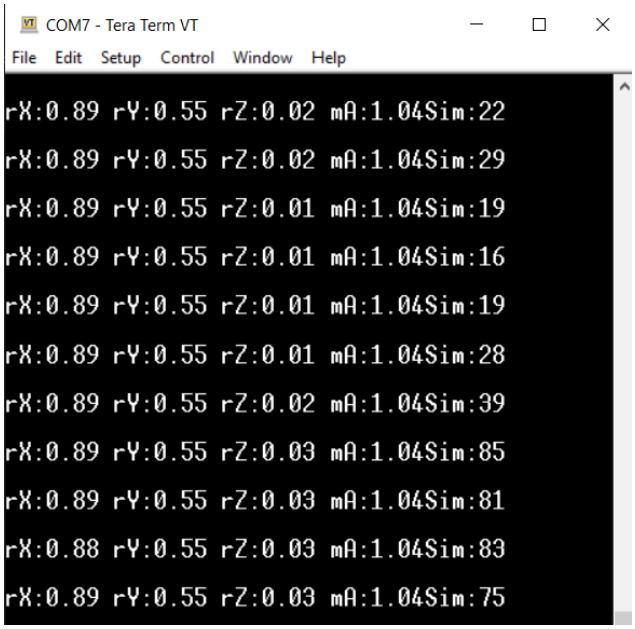


Figura 14: Gateway recebendo os dados com modelo embarcado

As principais etapas de validação incluíram:

- Avaliação da taxa de acerto: A porcentagem de acertos e erros na detecção de anomalias foi medida para cada cenário.
- Teste de detecção em tempo real: O dispositivo foi operado continuamente para verificar sua capacidade de identificar anomalias durante o funcionamento.
- Medição de latência: O tempo necessário para processar os dados e gerar uma decisão foi registrado.
- Estresse do sistema: Simulações de transições rápidas entre condições normais e anômalas foram realizadas para avaliar a estabilidade do sistema.

4.3.6 Embarcar o Modelo de Machine Learning do NanoEdge AI no Microcontrolador da Nordic

O *NanoEdge AI Studio* é uma ferramenta desenvolvida pela STMicroelectronics para criar modelos otimizados de aprendizado de máquina em sistemas embarcados. Embora seja projetado para suportar primariamente os microcontroladores da própria ST, também oferece compatibilidade com microcontroladores de outros fabricantes, como Nordic e Texas Instruments. É importante observar as condições de uso da licença, e neste projeto foi utilizada a licença de estudante.

Como o *NanoEdge AI Studio* gera bibliotecas pré-compiladas no formato `.a`, é fundamental selecionar corretamente o microcontrolador-alvo, respeitando as restrições de memória RAM e Flash configuradas durante a configuração do projeto no software. No caso deste projeto, foi utilizado o microcontrolador **nRF52832** da Nordic, o que demandou algumas adaptações específicas para integrar a biblioteca na IDE *Segger Embedded Studio*.

4.3.7 Passos para Embarcar o Modelo de Machine Learning no nRF52832

1. Adicionando a Biblioteca ao Projeto

- No *Segger Embedded Studio*, clique com o botão direito no projeto no painel *Project Explorer*.

- Selecione a opção *Add Existing File....*
- Navegue até o diretório onde a biblioteca `.a` gerada pelo *NanoEdge AI Studio*, selecione-a e clique em *Open* para adicioná-la ao projeto.

2. Configurando o Linker

- Clique novamente com o botão direito no projeto e selecione *Options....*
- No menu de opções, vá até *Linker* > *Libraries*:
  - Em *Additional Libraries*, insira o nome da biblioteca sem o prefixo `lib` e sem a extensão `.a`. Por exemplo, para o arquivo `libneai.a`, insira apenas `neai`.
  - Em *Library Search Directories*, adicione o caminho do diretório onde a biblioteca `.a` está localizada.

3. Configurando o Arquivo `flash_placement.xml`

Para garantir que os recursos de memória Flash e RAM sejam adequadamente alocados para o modelo *NanoEdge AI*, é necessário editar o arquivo `flash_placement.xml`. Os valores exatos de consumo de memória devem ser obtidos no arquivo de metadados gerado pelo *NanoEdge AI Studio*, que descreve o espaço necessário para o código e os dados da biblioteca.

- **Seção de Código da Biblioteca (`.neai`):** Adicione a seguinte entrada à memória Flash no arquivo `flash_placement.xml`:

  ```
  <ProgramSection alignment="4" load="Yes" name=".neai" size="0x0C00" /> <!-
  - 3 KB -->
  ```

  Esta configuração reserva 3 KB de Flash para o código da biblioteca *NanoEdge AI*. No entanto, o tamanho exato deve ser ajustado com base no valor indicado no arquivo de metadados gerado pelo software.

- **Seção de Dados da Biblioteca (`.neai_data`):** Adicione a seguinte entrada à memória RAM:

```
<ProgramSection alignment="4" load="No" name=".neai_data" size="0x099A" /> <!-
- 2.4 KB -->
```

Aqui, 2.4 KB de RAM são reservados para os dados dinâmicos e buffers utilizados pela biblioteca *NanoEdge AI*. Assim como na seção de código, o valor exato deve ser configurado conforme especificado no arquivo de metadados.

4. Compilação e Verificação

- Após configurar os arquivos e parâmetros, compile o projeto no *Segger Embedded Studio*.
- Caso ocorram erros de alocação de memória, o compilador fornecerá informações para ajustes nos limites configurados.

4.4 Desenvolvimento de Case Mecânico

O case mecânico foi projetado para proteger os componentes eletrônicos de condições adversas, como impactos, variações de temperatura e exposição a umidade. O processo de design incluiu:

- Uso do **Fusion 360 CAD** para modelagem 3D.
- Prototipagem em **impressora 3D** com resina SLA, permitindo ajustes e iterações rápidas.
- Considerações para um invólucro compacto, resistente e adequado à dissipação térmica.

A solução final permitiu a integração eficiente dos sensores e proteção em ambientes industriais.

4.5 Levantamento de Mercado e Custo do Projeto

Uma análise detalhada de mercado foi realizada para avaliar a viabilidade econômica do projeto. Foram levantados custos dos componentes, comparados fornecedores e identificadas

soluções similares no mercado. Além disso, foi feita uma estimativa de custo para produção em larga escala, considerando fabricação de PCB, montagem e testes.

Essa etapa foi essencial para garantir que o projeto pudesse competir com outras soluções de mercado e oferecer um custo-benefício atrativo.

4.6 Testes e Avaliação

Os testes foram realizados com a placa fornecida pela Pullup. Esses incluíram:

- **Validação dos sensores**: Precisão e resposta dos sensores de temperatura e aceleração foram avaliadas em diferentes condições.
- **Rede Mesh Bluetooth**: Foram testadas estabilidade, confiabilidade e latência na transmissão de dados.
- **Modelo de Machine Learning**: Um setup simplificado foi usado para simular falhas em rolamentos de motores industriais, validando o modelo de detecção de anomalias gerado.

5 Arquitetura do Projeto

A arquitetura do projeto é composta por uma rede de sensores Bluetooth de temperatura e aceleração, integrados a uma rede mesh. Esses sensores são responsáveis por monitorar condições ambientais e enviar os dados coletados através da rede Bluetooth Mesh, garantindo cobertura e confiabilidade mesmo em ambientes com obstáculos ou longas distâncias. Cada sensor é equipado com módulos de comunicação Bluetooth de baixo consumo, permitindo o envio eficiente de informações coletadas para um gateway central. Esse gateway, um dongle USB conectado a um computador, é o ponto de interface entre os sensores e o sistema de monitoramento central. Ele recebe os dados da rede mesh e transmite para um terminal no computador. O terminal exibe as informações em tempo real, e o sistema pode ser facilmente expandido para incluir um banco de dados que armazene os dados coletados. Em uma fase futura, essas informações podem ser enviadas para um serviço em nuvem, possibilitando a visualização e análise remota, além de garantir histórico e monitoramento contínuo dos dados de temperatura e aceleração.

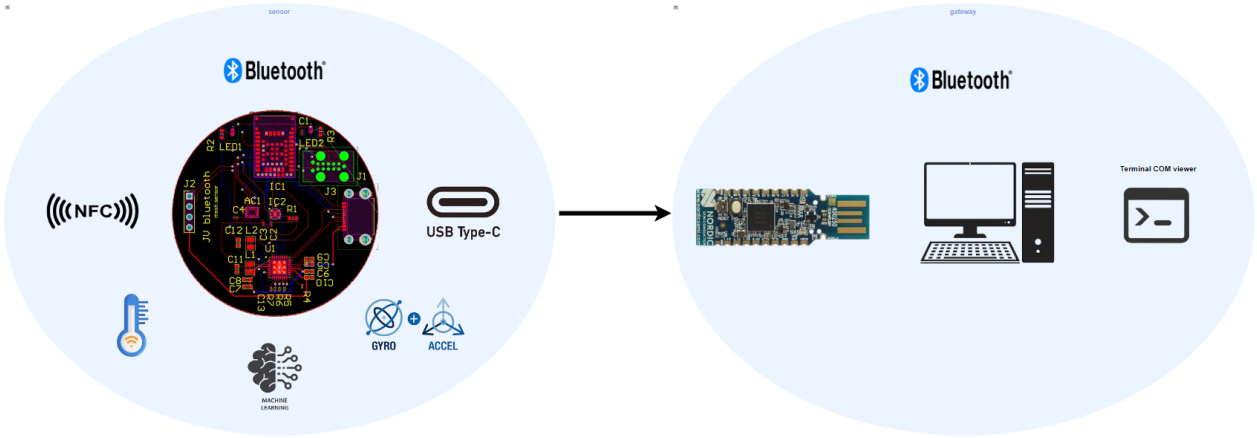


Figura 15: Arquitetura do projeto

A arquitetura modular do sistema permite flexibilidade e escalabilidade, facilitando a integração de novos sensores ou a expansão para múltiplos gateways, conforme a necessidade do projeto.

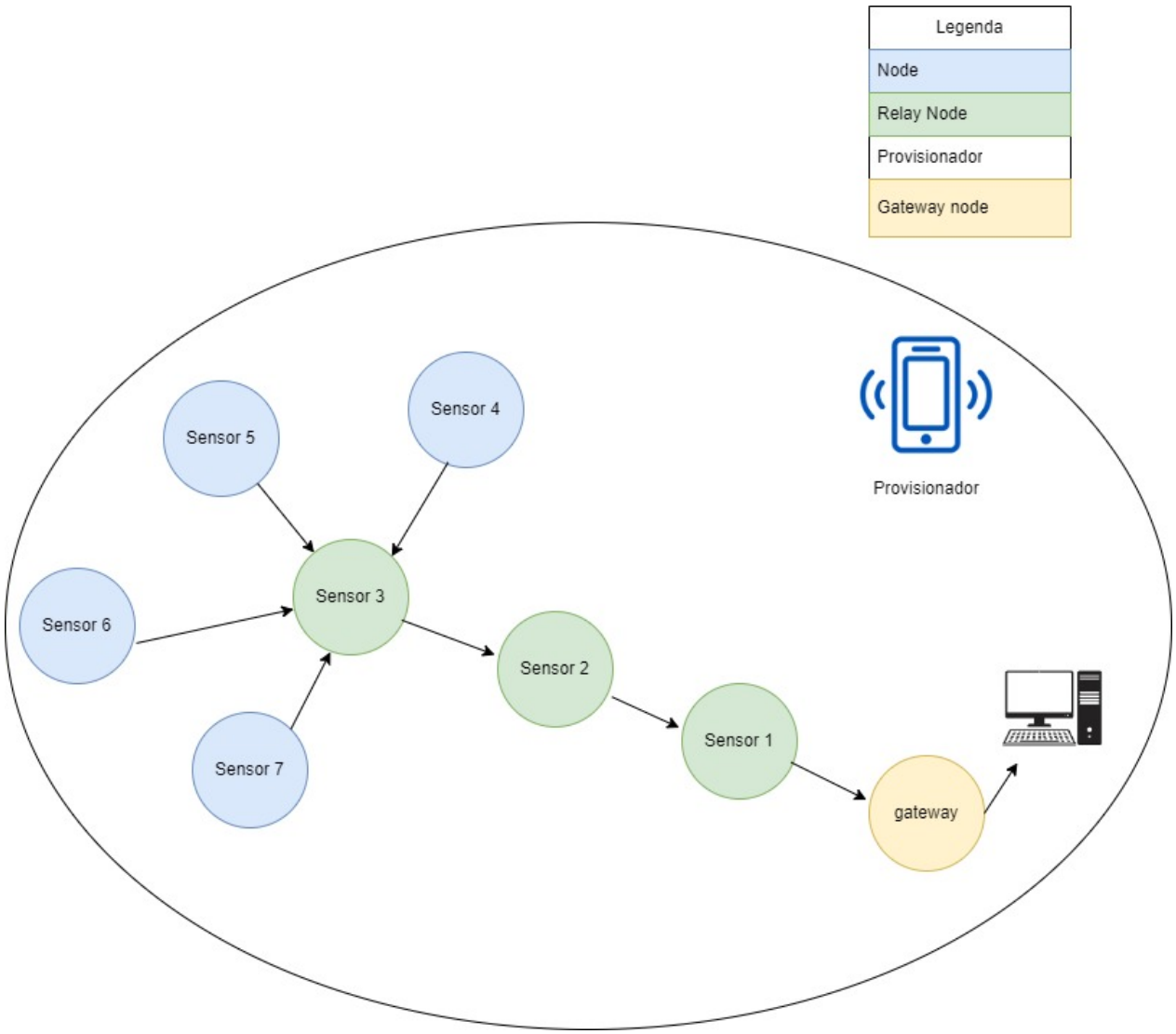


Figura 16: Diagrama de rede

Este diagrama ilustra uma rede Mesh Bluetooth composta por vários sensores e um gateway, com um celular atuando como o **provisionador**. A função principal do provisionador é configurar a rede, distribuindo as chaves e definindo o papel de cada nó dentro da rede Mesh.

- **Nó Relay (Verde)**: Os sensores representados em verde (Sensor 3 e Sensor 2) possuem a característica de **Relay** ativada. Isso significa que, além de processarem suas próprias informações, esses sensores retransmitem dados de outros sensores, ampliando o alcance da rede. A função de relay é configurada pelo **provisionador**, que é o celular. Os nós relay são cruciais para garantir que as informações de sensores mais distantes, como o **Sensor 5**, cheguem ao **gateway**.

- **Nó Normal (Azul)**: Os sensores representados em azul (Sensor 5, Sensor 6, Sensor 7 e Sensor 4) são nós que não retransmitem dados, mas enviam diretamente para os sensores Relay (como o Sensor 3). Eles são configurados pelo provisionador para atuar apenas como sensores.
- **Gateway (Amarelo)**: O **gateway** coleta todas as informações transmitidas pelos sensores através da rede mesh. Ele está conectado a um computador, para onde encaminha os dados capturados pelos sensores. No diagrama, ele é mostrado como o destino final das informações da rede.
- **Provisionador (Celular)**: O provisionador é o dispositivo responsável por **iniciar** e **configurar** toda a rede Mesh. No caso deste projeto, o celular é o provisionador, que se comunica via Bluetooth GATT com todos os dispositivos. O GATT (Generic Attribute Profile) de todos os dispositivos está habilitado para garantir essa comunicação direta entre o provisionador e os sensores, permitindo a configuração inicial da rede e a administração de novos dispositivos.

GATT (Generic Attribute Profile):

Todos os dispositivos têm o GATT habilitado para garantir que o **celular provisionador** possa se conectar a eles durante o processo de configuração. O GATT é uma característica essencial para que o celular possa estabelecer e gerenciar a comunicação com cada dispositivo individualmente, especialmente durante o provisionamento.

Fluxo de Dados:

- Os sensores coletam as informações (ex.: aceleração, temperatura).
- Sensores sem função relay (azul) enviam os dados aos sensores relay (verde).
- Os sensores relay retransmitem os dados para o **gateway**, que está conectado ao computador.
- O provisionador (celular) administra e pode alterar as configurações dos sensores conforme necessário.

6 Projeto de desenvolvimento de produto

6.1 Projeto de Hardware

O desenvolvimento do hardware para o projeto de monitoramento via rede Mesh Bluetooth envolveu a criação de uma placa customizada para o sensor, incorporando diversas melhorias em relação à placa fornecida pela Pullup Soluções, além da utilização de uma placa comercial para o gateway. As alterações realizadas na placa do sensor buscaram atender às exigências de compactação, eficiência energética, comunicação avançada e capacidade de processamento, garantindo um desempenho superior para aplicações industriais.

Os detalhes do esquemático e do layout do hardware desenvolvido podem ser visualizados na seção de apêndices, nas Figuras 33, 34, 35 e 36.

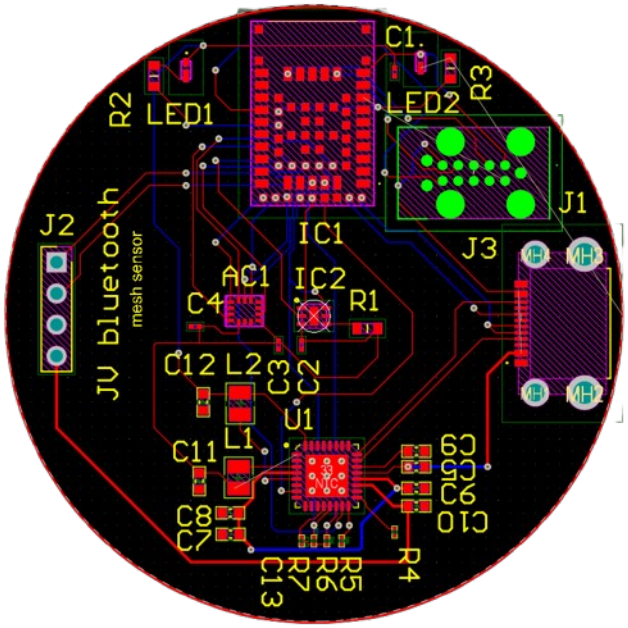


Figura 17: layout PCB em 2D

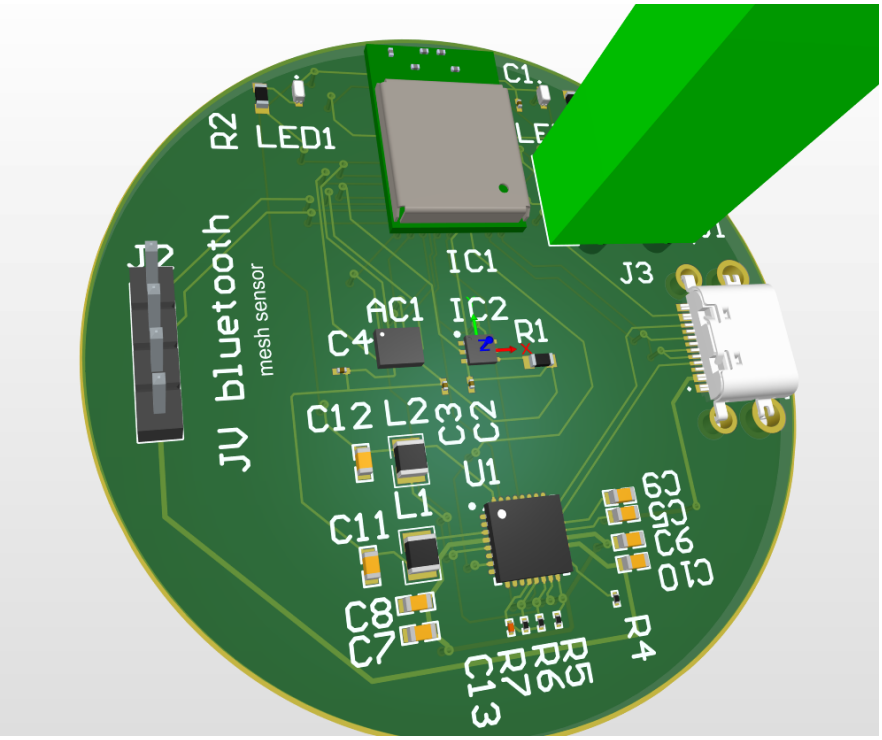


Figura 18: layout PCB em 3D

6.1.1 Alterações no Hardware do Sensor

As principais modificações realizadas entre a placa inicial da Pullup e o design final do projeto incluem:

- **Substituição do Acelerômetro**: O acelerômetro *LIS2DH12* foi substituído pelo *IIS3DWBTR*, que oferece uma taxa de amostragem significativamente superior (26,7 kHz contra 1,344 kHz), essencial para capturar vibrações de alta frequência em aplicações industriais. Essa diferença permite uma análise mais precisa das condições de vibração e amplia o escopo de detecção de anomalias.

- **Atualização do Microcontrolador**: O *nRF52832* foi substituído pelo *nRF52840*, mantendo a compatibilidade com a família de microcontroladores Nordic Semiconductor. A principal vantagem dessa troca foi o aumento da memória RAM (de 64 KB para 256 KB), permitindo armazenar mais amostras e rodar algoritmos mais complexos de *Machine Learning*, sem alterar significativamente a base de programação.

- **Adição de um Gerenciador de Energia**: Foi incluído o *nPM1300*, um *Power Management Integrated Circuit* (PMIC), que permite a utilização de uma bateria recarregável, gerencia a alimentação dos componentes do sistema e fornece informações precisas sobre o estado de carga da bateria. Essa modificação melhora significativamente a autonomia e a confiabilidade do dispositivo em campo.
- **Incorporação de uma Antena NFC Flexível**: Uma antena NFC flexível foi adicionada para possibilitar comunicação em curta distância, substituindo a antena rígida utilizada na versão anterior. A antena rígida apresentou problemas de fabricação e baixa eficiência. A nova antena flexível facilita o diagnóstico e a configuração do sensor sem a necessidade de desmontar o invólucro à prova d'água, aumentando a praticidade de manutenção em ambientes industriais.

Sensor Customizado:

1. **Microcontrolador nRF52840**:[**29**]

   O **nRF52840** é um SoC avançado da Nordic Semiconductor, que oferece suporte completo ao Bluetooth 5 e ao Bluetooth Mesh. Ele conta com um processador ARM Cortex-M4 de 32 bits com FPU (unidade de ponto flutuante), operando a 64 MHz, o que proporciona o desempenho necessário para executar tarefas complexas como a coleta e processamento de dados de sensores e a execução de inferência de modelos de machine learning.

   - **Memória Flash**: 1 MB
   - **RAM**: 256 KB
   - **Periféricos**: suporte a diversas interfaces, incluindo I2C, SPI e UART, além de módulos dedicados para gerenciamento de energia, que ajudam a manter o consumo de energia em níveis extremamente baixos.
   - **Função no Projeto**: O nRF52840 é responsável por gerenciar a rede mesh Bluetooth, realizar o processamento dos dados do sensor, implementar as rotinas de machine learning embarcado e coordenar a comunicação com outros dispositivos da rede.

2. **Sensor de Temperatura STTS22HTR**:

   O **STTS22HTR** [**30**].é um sensor de temperatura digital de alta precisão da STMicroelectronics. Ele foi escolhido por sua baixa corrente de operação, precisão de ±0.5°C e pela capacidade de gerar interrupções com base em thresholds programados (limites de temperatura). Isso permite que o dispositivo entre em modos de economia de energia e somente acorde quando necessário

   - **Faixa de Medição**: -40°C a +125°C
   - **Resolução**: 0.5°C
   - **Funções Especiais**: Suporte a interrupções por threshold, permitindo a detecção instantânea de condições críticas de temperatura sem a necessidade de leitura constante.
   - **Função no Projeto**: O STTS22HTR monitora a temperatura ambiente, gerando dados que podem ser usados para detectar variações de temperatura e, em casos extremos, acionar alertas por meio da rede mesh. Suas interrupções por threshold ajudam a reduzir o consumo de energia ao permitir leituras intermitentes.

3. **Acelerômetro IIS3DWBTR**: [**14**]

   - **Taxa de Amostragem**: Até 26,7 kHz;
   - **Faixa de Medição (g)**: ±2 g, ±4 g, ±8 g e ±16 g;
   - **Capacidade de FIFO**: Até 4096 níveis de amostragem;
   - **Resolução**: 16 bits.

   Função no Projeto: Captura vibrações de alta frequência para análise detalhada e suporte a algoritmos de detecção de anomalias.

4. **Power Management IC nPM1300**: [**32**]

   O **nPM1300** é um PMIC (Power Management Integrated Circuit) multifuncional da Nordic Semiconductor, projetado para fornecer alimentação eficiente e gerenciar as necessidades energéticas de dispositivos de baixo consumo.

- **Reguladores**: Possui reguladores de tensão ajustáveis de 1.8V, 3V e 5V, garantindo uma alimentação estável e eficiente para diferentes componentes do sistema.
- **Carregador de Bateria**: Inclui um carregador de bateria de lítio-íon/polímero.
- **Battery Gauge**: Fornece informações precisas sobre o estado de carga da bateria, auxiliando no gerenciamento de energia do dispositivo.
- **Função no Projeto**: O nPM1300 gerencia a alimentação de todos os componentes, garantindo que o sistema opere de maneira eficiente e prolongando a vida útil da bateria. Ele também realiza o carregamento da bateria e monitora o nível de carga, o que é crucial para garantir o funcionamento contínuo do dispositivo em campo.

5. **Antena NFC PCB**:

   A antena NFC foi incluída no design para fornecer uma interface de comunicação sem fio em curta distância, especialmente útil para a leitura e programação do dispositivo quando ele está dentro de um invólucro à prova d'água. **Função no Projeto**: Permite a comunicação NFC, possibilitando que os dados de diagnóstico e configuração sejam acessados sem a necessidade de abrir o invólucro, mantendo a integridade do dispositivo em ambientes hostis.

6. **Coin Cell Lithium Battery 3.7V 500mAh**:

   A bateria de 500mAh foi escolhida para fornecer uma longa duração de operação ao dispositivo.

   Função no Projeto: A bateria alimenta o sensor por longos períodos, possibilitando a operação autônoma em redes mesh Bluetooth sem a necessidade de recargas frequentes.

- **Capacidade da bateria**: 500 mAh (miliampère-hora).
- **Tensão da bateria**: 3.7V.
- **Tensão de operação do dispositivo**: 3.0V.

Calculo de consumo da bateria

Utilizando o simulador de consumo do microcontrolador da nordic fig [41] temos um consumo aproximado de 117ua

A energia total disponível da bateria, em miliwatt-hora (mWh), é calculada como:

$$\text{Energia total (mWh)} = \text{Capacidade da bateria (mAh)} \times \text{Tensão da bateria (V)}$$

Substituindo os valores:

$$500\,\text{mAh} \times 3.7\,\text{V} = 1850\,\text{mWh}$$

Potência consumida pelo dispositivo:

A potência consumida pelo dispositivo, em miliwatts (mW), é calculada como:

$$\text{Potência (mW)} = \left(\frac{\text{Consumo médio (µA)}}{1000}\right) \times \text{Tensão do dispositivo (V)}$$

Substituindo os valores:

$$\left(\frac{117\,\mu\text{A}}{1000}\right) \times 3.0\,\text{V} = 0.351\,\text{mW}$$

Duração da bateria:

A duração da bateria, em horas, é calculada dividindo a energia total disponível pela potência consumida pelo dispositivo:

$$\text{Duração (horas)} = \frac{\text{Energia total (mWh)}}{\text{Potência consumida (mW)}}$$

Substituindo os valores:

$$\frac{1850\,\text{mWh}}{0.351\,\text{mW}} = 5270.66\,\text{horas}$$

Duração em dias:

Finalmente, a duração da bateria em dias pode ser obtida dividindo o resultado anterior por 24:

$$\frac{5270.66\,\text{horas}}{24} = 219.61\,\text{dias}$$

Gateway

No projeto de monitoramento via rede mesh Bluetooth, o **gateway** é um componente crítico responsável por coletar os dados enviados pelos sensores e encaminhá-los para sistemas de processamento ou armazenamento centralizados. Para o desenvolvimento do gateway, foi escolhido o **dongle USB Nordic PCA10059**, uma solução compacta e eficiente para lidar com a comunicação de redes mesh Bluetooth e permitir a integração com um computador ou outro sistema via USB.

Componentes do Gateway:

Dongle USB Nordic PCA10059:

O **Nordic PCA10059** [**33**]: é uma placa de desenvolvimento baseada no microcontrolador **nRF52840** da Nordic Semiconductor, projetada para ser usada como um dongle USB, facilitando a criação de gateways Bluetooth Mesh com comunicação USB. Essa placa compacta contém o microcontrolador **nRF52840** , além de uma interface USB que permite conectá-la diretamente a um computador para monitoramento e controle da rede mesh.

Figura 19: dongle Noridc PCA10059

6.2 Projeto Mecânico

Para garantir a durabilidade do dispositivo, o projeto mecânico foi desenvolvido para ser **à prova d'água**, utilizando um **case de plástico** robusto e selado. O case possui as seguintes características:

1. **Design do Case**:

   O invólucro mecânico é uma **caixa redonda com tampa**, projetada para abrigar toda a eletrônica do sensor. A escolha desse formato visa facilitar o fechamento hermético, protegendo os componentes internos contra poeira e água, com classificação IP67. O uso de uma tampa de rosca ou vedação com anel de borracha (O-ring) garante a resistência à entrada de líquidos.

2. **Montagem e Fixação**:

   O case foi projetado com um **parafuso M8** integrado na parte inferior, permitindo que o sensor seja **rosqueado e fixado em qualquer superfície** a ser monitorada, como estruturas metálicas, paredes ou máquinas industriais. A fixação robusta ajuda a garantir que o sensor permaneça estável durante o monitoramento de vibrações e temperaturas, essencial para obter dados precisos e consistentes.

3. **Material de Prototipagem:**

   Como protótipo, o case será fabricado em uma impressora 3D utilizando o material **resina SLA**, conhecido por sua alta precisão e acabamento superficial adequado para testes iniciais. A resina oferece boa resistência mecânica e precisão de detalhes, garantindo que o case acomode perfeitamente os componentes eletrônicos e tenha a vedação necessária.

4. **Estudo de Materiais para o Projeto Final:**

   Para o projeto final, será necessário utilizar um material mais resistente, capaz de suportar as condições adversas esperadas em ambientes industriais e externos. Isso inclui exposição prolongada ao sol, raios ultravioleta, chuva e pequenos impactos. Será

necessário realizar um estudo detalhado de materiais, buscando opções que combinem alta resistência mecânica, durabilidade contra intempéries e capacidade de manter a integridade estrutural ao longo do tempo.

5. **Proteção Eletrônica**:

   A caixa abriga os componentes eletrônicos de forma segura, garantindo que o microcontrolador, sensores e outros circuitos estejam protegidos contra danos físicos, corrosão e outros fatores ambientais. A disposição interna foi projetada para minimizar o movimento dos componentes, garantindo que as vibrações detectadas pelo sensor de aceleração sejam fiéis ao ambiente externo e não geradas por deslocamentos internos dos componentes

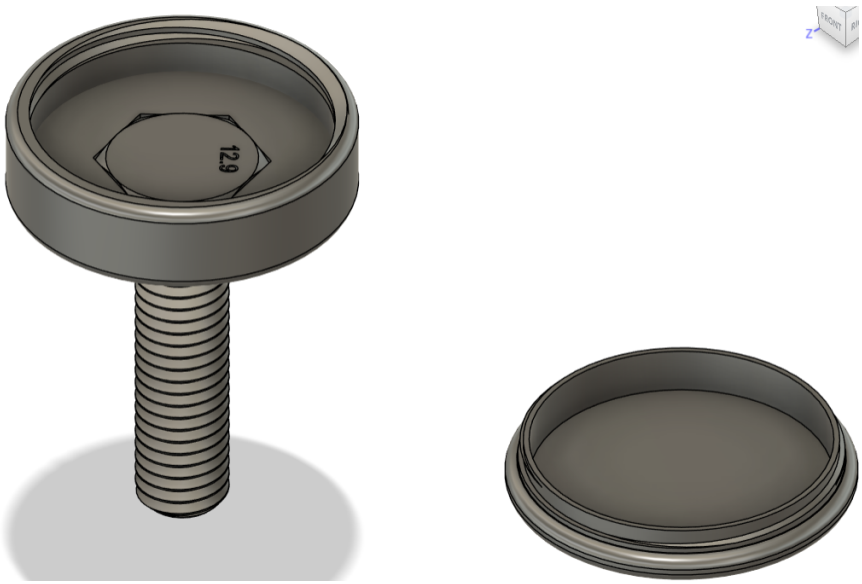


Figura 20: Case mecanico 3D

6.3 Projeto de Software

O desenvolvimento do firmware foi realizado em linguagem C, utilizando o SDK v17.1.0 . Este SDK proporciona um ambiente de desenvolvimento abrangente para dispositivos da série nRF5, oferecendo uma ampla seleção de drivers, bibliotecas, exemplos de periféricos e protocolos de rádio proprietários. Além disso, o **nRF5 SDK para Mesh v5.0.0** foi utilizado

para implementar a comunicação Bluetooth® Mesh. Este SDK permite que os dispositivos aproveitem os recursos da rede Mesh ao rodar nos chips da série nRF5 da Nordic Semiconductor.

Além disso, foi desenvolvido um firmware alternativo para o sensor, com a finalidade de transmitir os dados de aceleração via Bluetooth sem a utilização da rede Mesh. Essa abordagem foi necessária para viabilizar o treinamento e a validação do modelo de Machine Learning, uma vez que o microcontrolador utilizado não possuía memória suficiente para suportar simultaneamente a rede Mesh e os algoritmos de Machine Learning.

A arquitetura e a máquina de estados permaneceram essencialmente inalteradas, com a única modificação sendo a desativação da funcionalidade de rede Mesh para permitir o foco exclusivo na transmissão dos dados de aceleração

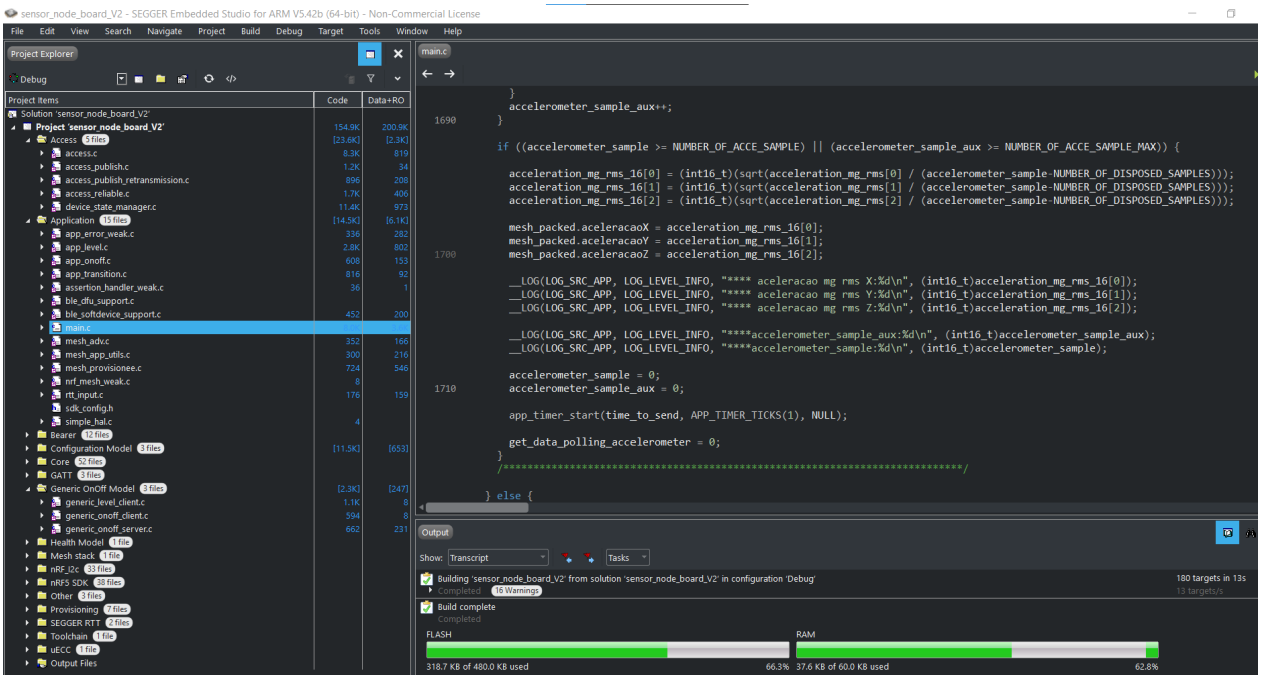


Figura 21: IDE SEGGER Embedded studio

Máquina de Estados do sensor

A máquina de estados foi projetada com foco em **eficiência energética** e **uso otimizado dos recursos** de processamento de memória. A principal prioridade do sistema é garantir que o sensor opere com o menor consumo de energia possível, especialmente durante longos períodos de operação em campo.

Inicialização do Sistema

1. **Configuração Inicial**: O programa inicia configurando os clocks, timers e GPIOs do microcontrolador. Logo após, o firmware verifica se há alguma chave de provisionamento da rede Mesh armazenada na memória flash do microcontrolador. **Sem Provisionamento**: Caso as chaves não estejam presentes, o sensor entra automaticamente no **modo de provisionamento**, habilitando o Bluetooth no modo GATT. Nesse estado, o sensor faz o **advertisement**, permitindo que um dispositivo provisionador se conecte e envie as informações da rede Mesh.

2. **Provisionamento Bluetooth Mesh**: Para o provisionamento da rede Mesh, foi utilizado o aplicativo **nRF Mesh**, disponibilizado gratuitamente pela Nordic Semiconductor. Através do aplicativo, o dispositivo se conecta ao sensor via Bluetooth, onde as chaves de acesso e o grupo ao qual o sensor pertence são configurados. Após essa etapa, as informações são gravadas na memória flash do microcontrolador.

3. **Inicialização Pós-Provisionamento**: Após o provisionamento, o sensor inicializa os outros periféricos, configurando os pinos de I2C e ativando os sensores de temperatura e aceleração.

Verificação de Estado Após a inicialização do hardware, o sistema verifica o estado atual do sensor:

- **Modo de Medição ou Estoque**: Através da leitura da memória NFC, o firmware determina se o dispositivo está em modo de medição ou em modo de estoque (economia de energia).
  - **Modo de Estoque**: Caso o sensor esteja no modo de economia de energia, ele entra em **ultra low power mode**, minimizando o consumo.
  - **Modo de Medição**: No modo de medição, o sistema configura completamente os sensores e periféricos, inicializando a comunicação I2C e capturando dados dos sensores.

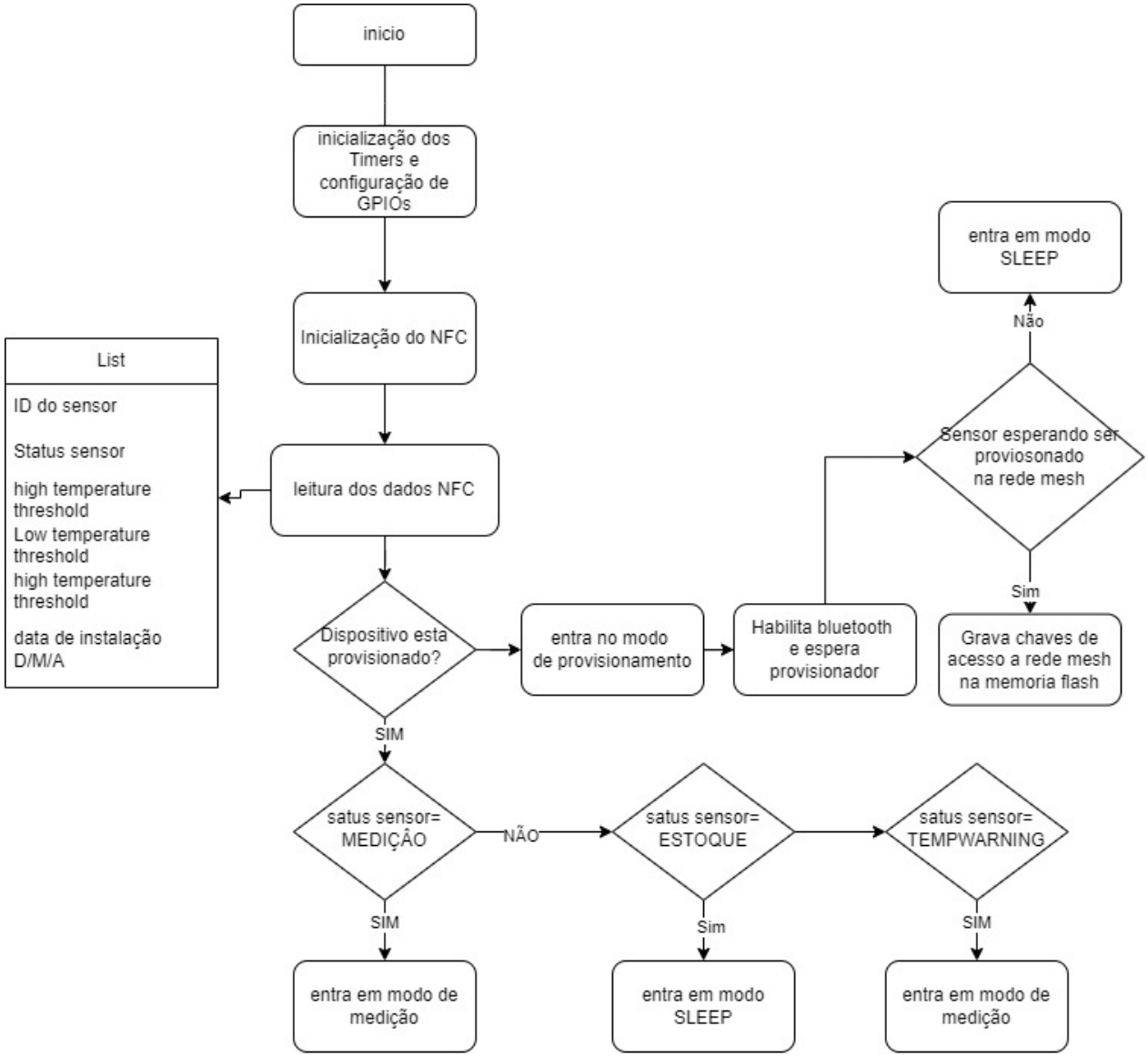


Figura 22: Diagrama de estados do sensor em provisionamento

Medições e Processamento

1. **Captura de Dados**: O firmware captura as leituras de temperatura e, no caso do sensor de aceleração, coleta **100 amostras RMS**. Após essa coleta, o sistema processa os dados de aceleração utilizando a função de **Machine Learning embarcada** no acelerômetro para realizar a predição de falhas, retornando o percentual de similaridade calculado.

2. **Envio de Dados**: Após as medições, o sensor envia as informações processadas através da rede Bluetooth Mesh. O envio é tentado até 5 vezes, e em caso de falha no envio

(sem ACK), o sensor entra em modo **low energy** para economizar energia.

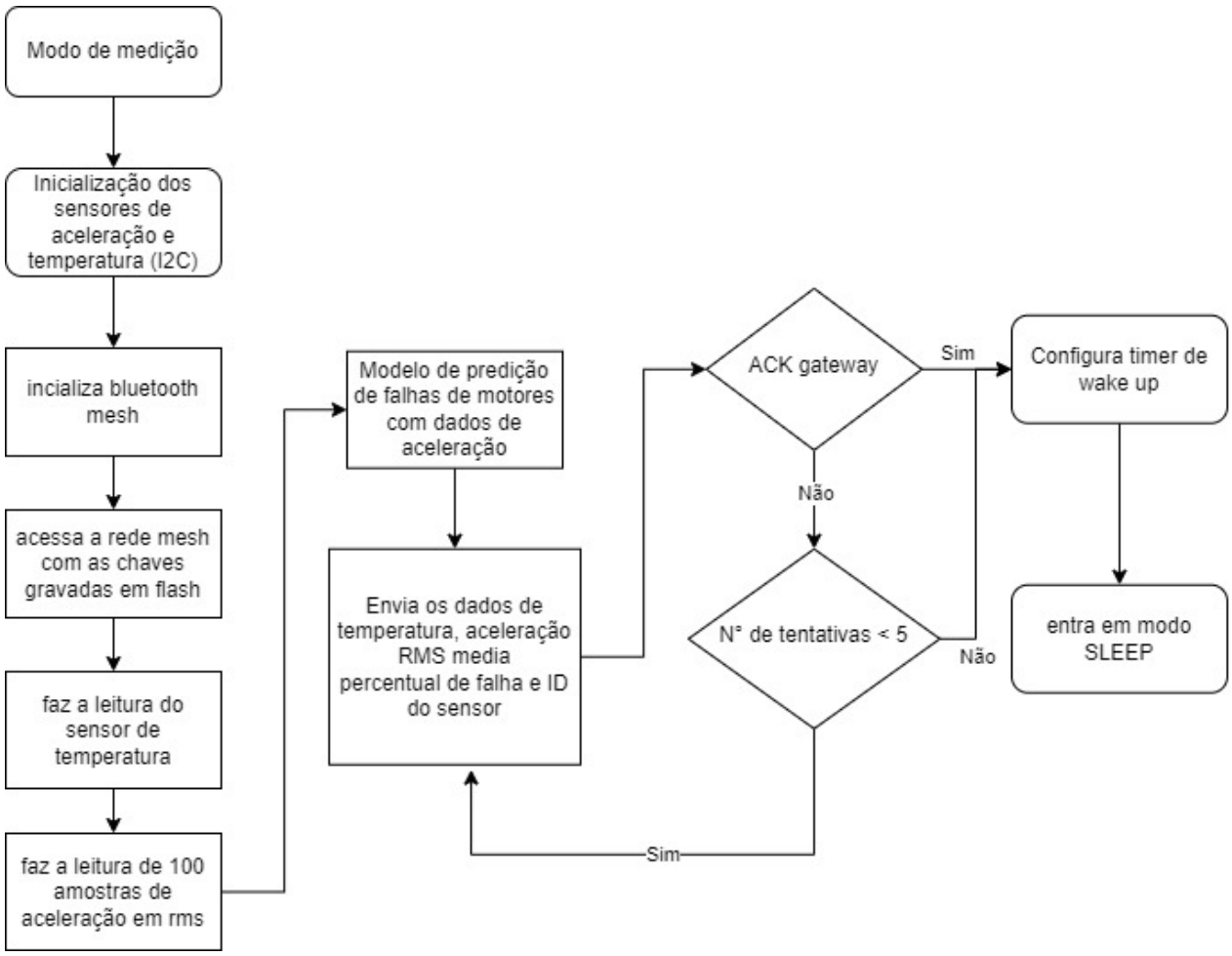


Figura 23: Diagrama de estados do sensor em medição

Máquina de Estados do Gateway

O gateway do projeto, implementado utilizando o **dongle USB Nordic PCA10059**, desempenha a função de coordenar a comunicação da rede Bluetooth Mesh e transmitir os dados coletados dos sensores para o computador via USB. A máquina de estados do gateway foi projetada para operar de maneira eficiente, garantindo a inicialização da rede Mesh e a coleta contínua de dados dos sensores, com envio confiável ao computador.

Estados do Gateway

1. **Inicialização**:

- O sistema inicia configurando os periféricos do dongle, incluindo os **clocks**, **timers**, **GPIOs**, e a **interface USB**. A seguir, o firmware verifica se há chaves de provisionamento ou grupos Mesh armazenados na memória flash.
- Caso essas informações estejam presentes, o gateway configura o stack Bluetooth Mesh, entrando diretamente no estado de operação da rede. Se não houver chaves, o gateway inicia um novo processo de **provisionamento** para se integrar à rede Mesh.

2. **Provisionamento da Rede Mesh**:

   - No modo de provisionamento, o gateway anuncia sua presença na rede Bluetooth Mesh, permitindo que outros dispositivos Mesh (sensores) se conectem.
   - Após o processo de provisionamento, que envolve a configuração de grupos e endereços, as informações são armazenadas na memória flash, e o gateway passa para o estado de operação.

3. **Operação na Rede Mesh**:

   - No estado de operação, o gateway entra em **modo de escuta** contínua, monitorando as mensagens transmitidas pelos sensores da rede Mesh. Ele fica atento a pacotes de dados vindos dos sensores de temperatura e aceleração.
   - Ao receber um pacote de dados de um sensor, o gateway realiza a verificação de integridade e ACK (Acknowledgement) para garantir que os dados foram recebidos corretamente. Se houver falhas, solicite retransmissão dos dados.

4. **Envio de Dados via USB**:

   - Após o recebimento correto dos dados de um sensor, o gateway utiliza a interface USB para transmitir as informações ao computador. O envio é feito em tempo real, utilizando a porta serial virtual configurada no dongle.
   - O gateway continua no ciclo de escuta e envio, aguardando novos pacotes de dados dos sensores e garantindo que as informações sejam passadas para o computador sem perda.

5. **Gerenciamento de Erros e Reconfiguração**:

   - Se o gateway detectar falhas recorrentes na rede Mesh (por exemplo, pacotes não confirmados ou dispositivos não acessíveis), ele pode reiniciar a conexão com a rede e tentar reconfigurar os dispositivos conectados.
   - Em casos extremos, o gateway pode reiniciar o processo de provisionamento para garantir a continuidade da comunicação.

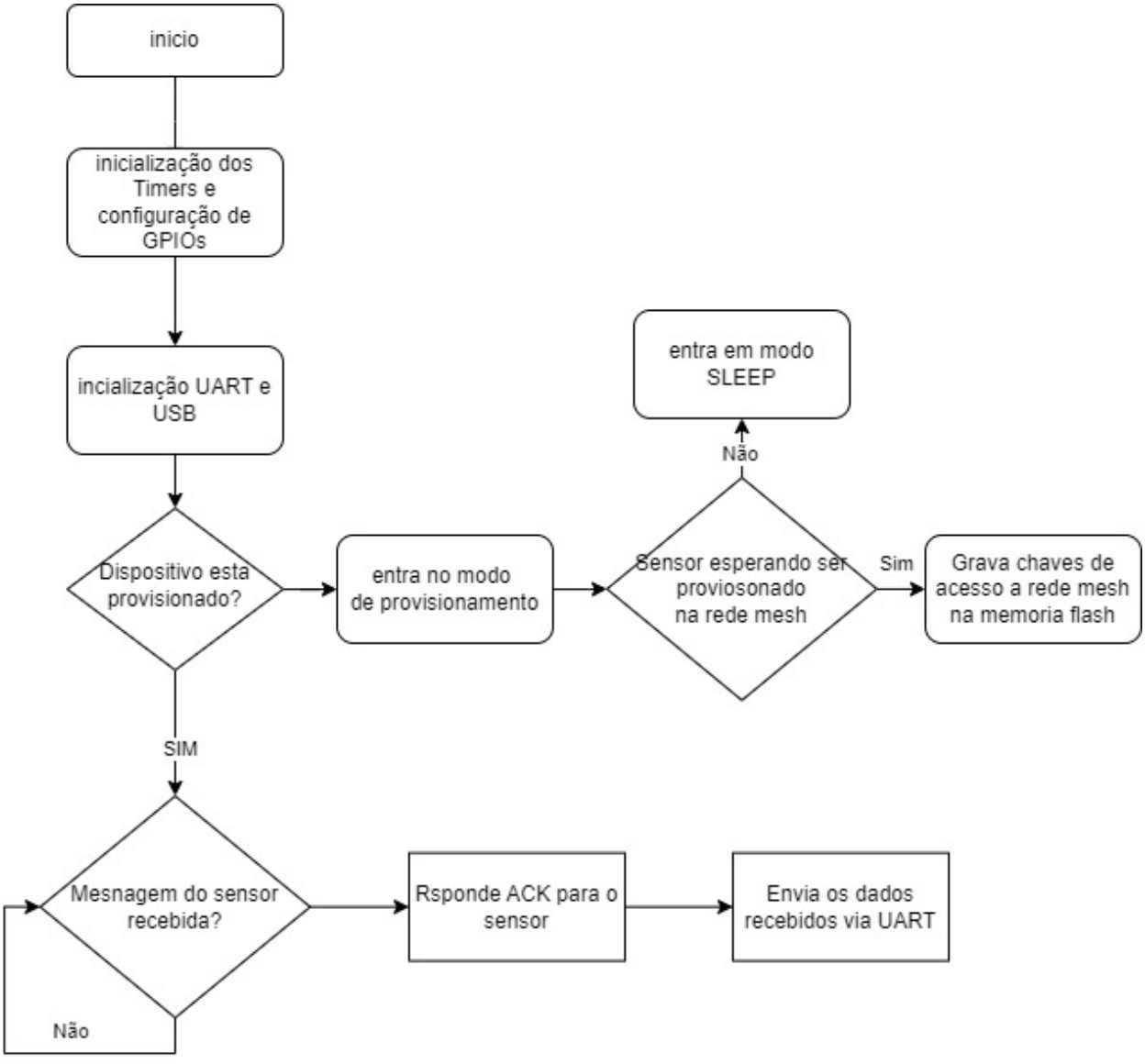


Figura 24: Diagrama de estados do gateway

6.3.1 Desenvolvimento do Projeto: Machine Learning

No desenvolvimento do modelo de detecção de anomalias baseado em *Machine Learning*, a escolha e especificação do hardware desempenharam um papel essencial. Durante os testes iniciais, foi utilizado o acelerômetro **LIS2DH12**, integrado à placa disponibilizada pela Pullup Soluções. Esse dispositivo possui uma taxa máxima de amostragem de **1.344 Hz** [**34**]. Para o projeto final, foi especificado o acelerômetro **IIS3DWBTR**, capaz de operar com uma taxa de amostragem significativamente maior, de até **26,7 kHz** [**14**], permitindo capturar vibrações em frequências mais altas, típicas de aplicações industriais avançadas.

Nos testes com o **LIS2DH12**, foram realizadas simplificações para atender às limitações do hardware. Foi estabelecida uma frequência de análise de **500 Hz** e um tamanho de buffer ajustado às restrições de memória, mantendo a funcionalidade dentro das capacidades disponíveis e capturando vibrações características relevantes.

Premissas de Amostragem e Especificação Técnica

Para monitorar vibrações em uma frequência máxima de **500 Hz**, as seguintes premissas foram adotadas com base no **Teorema de Nyquist**: - O **Teorema de Nyquist** estabelece que a frequência de amostragem deve ser, no mínimo, o dobro da frequência máxima do sinal a ser analisado. Assim:

$$f_s \geq 2 \cdot f_{\text{máx}} = 2 \cdot 500 = 1.000\,\text{Hz}.$$

- A frequência de amostragem de **1.344 Hz** é suficiente, pois excede o requisito mínimo e permite capturar vibrações de até **672 Hz**, cobrindo as frequências de interesse.

Definição do Modelo de Amostragem

Para capturar e analisar o comportamento vibracional do sistema, foi decidido que a análise cobriria **100 ciclos de uma vibração de 500 Hz**. Essa abordagem garante a captura de padrões repetitivos suficientes para identificar possíveis anomalias. Os cálculos realizados para determinar o número de amostras e o consumo de memória foram os seguintes:

1. **Período de um ciclo de 500 Hz:** A frequência de 500 Hz significa que cada ciclo dura:

$$T_{\text{ciclo}} = \frac{1}{500} = 0,002\,\text{s}\,(2\,\text{ms}).$$

2. **Tempo total para 100 ciclos:** O tempo necessário para capturar 100 ciclos completos é:

$$T_{100\text{ ciclos}} = 100 \cdot T_{\text{ciclo}} = 100 \cdot 0,002 = 0,2\,\text{s}\,(200\,\text{ms}).$$

3. **Número de amostras necessárias:** Com uma frequência de amostragem de **1.344 Hz**, o número de amostras para capturar 200 ms de sinal é:

$$N_{100\text{ ciclos}} = T_{100\text{ ciclos}} \cdot f_s = 0,2 \cdot 1.344 = 268,8.$$

   Arredondando para o próximo número inteiro:

$$N_{100\text{ ciclos}} = 269.$$

4. **Ajuste para três eixos (X, Y e Z):** Para capturar dados nos três eixos do acelerômetro:

$$N_{\text{total}} = N_{100\text{ ciclos}} \cdot 3 = 269 \cdot 3 = 807\,\text{amostras}.$$

5. **Consumo de memória:** Representando cada amostra como um número de ponto flutuante (*float*, 4 bytes por amostra), o consumo total de memória é:

$$\text{Memória total} = N_{\text{total}} \cdot 4 = 807 \cdot 4 = 3.228\,\text{bytes}\,(3,2\,\text{kB}).$$

Devido à restrição de memória do microcontrolador utilizado, o tamanho do buffer foi ajustado para capturar exatamente **100 ciclos de 500 Hz**, cobrindo **200 ms** de vibração. O tamanho de buffer definido para os três eixos foi de **807 amostras**, correspondendo a um consumo de **3,2 kB de memória**. Essa configuração permitiu detectar padrões vibracionais e identificar anomalias em testes controlados, mesmo com as limitações de hardware.

Para aplicações mais complexas ou frequências mais altas, o acelerômetro IIS3DWBTR, com sua capacidade de amostragem significativamente maior, será necessário para permitir maior resolução e uma análise mais ampla de frequências.

Configuração do Setup de Testes

Para validar o modelo, foi montado um setup simplificado utilizando uma ventoinha de motor DC. O sensor foi fixado no suporte do eixo para captar variações de vibração. As anomalias simuladas incluíram:

- Adição de um peso às hélices para causar **desbalanceamento**.
- Introdução de interferência mecânica.
- Restrição do fluxo de ar.

Esse sistema simplificado foi escolhido por ser mais viável do que realizar testes em motores trifásicos industriais, que exigiriam maior infraestrutura. Ainda assim, o setup foi suficiente para demonstrar a capacidade do modelo em identificar variações anômalas nos padrões de vibração.

Análise de Sinais e Eficiência do Modelo

A análise de sinais foi baseada nos princípios de processamento digital, considerando as seguintes etapas:

1. **Captura de Amostras**: As vibrações foram medidas nos três eixos (X, Y e Z), resultando em 807 amostras por ciclo.
2. **Preprocessamento**: Os dados capturados foram normalizados e analisados no domínio do tempo, ajustados para as limitações do microcontrolador.
3. **Treinamento e Inferência Local**: O modelo de aprendizado foi treinado para reconhecer padrões normais de vibração, e o processamento local mostrou-se eficiente, mesmo com ciclos de análise reduzidos. A detecção de anomalias ocorreu em tempo real, com desempenho satisfatório.

Etapas do Treinamento

O treinamento do modelo seguiu três etapas fundamentais:

1. **Project Settings**: Configurações gerais do projeto, como a seleção do processador (nRF52840) e a alocação de memória RAM e Flash, garantindo que o modelo se ajustasse aos recursos disponíveis no hardware embarcado.

2. **Inserção de Sinais**: Os dados de aceleração foram categorizados em **regular signals** (sinais normais) e **abnormal signals** (sinais anômalos). Esses dados, extraídos dos datasets, permitiram ao modelo reconhecer padrões normais e identificar anomalias com precisão.

Figura 25: Entrada de sinais de vibração regular do sistema

Figura 26: Entrada de sinais de vibração irregular do sistema

Benchmarking: A etapa de benchmarking avaliou o desempenho dos algoritmos treinados, medindo a capacidade do modelo de identificar falhas e classificar as anomalias corretamente.

Figura 27: Informações do modelo Gerado

Validação do Modelo

Após o treinamento, foi realizada a **validação** do modelo, um processo crítico para garantir a eficácia da solução embarcada. Nesta fase, o modelo foi submetido a novos dados, que não faziam parte do conjunto de treinamento, para verificar a sua capacidade de generalização. Foram avaliados:

- **Taxa de acerto e erro** do modelo em identificar falhas corretamente.

A validação foi fundamental para assegurar que o modelo fosse eficaz em diferentes cenários e que sua implementação não comprometesse a autonomia do sensor.

Deployment no Firmware

Com o modelo validado, a fase de **deployment** (implantação) foi realizada, onde o algoritmo de Machine Learning foi integrado ao firmware do sensor sem rede mesh utilizando o **NanoEdge AI Studio**. Esse processo envolveu:

- **Otimização da memória**: O modelo foi ajustado para utilizar eficientemente a memória RAM e Flash disponíveis no microcontrolador.

- **Integração com os sensores**: O modelo foi acoplado à captura de dados em tempo real dos sensores de aceleração, permitindo a predição de falhas diretamente no dispositivo.

Essa abordagem permitiu que o sensor operasse de forma autônoma e realizasse diagnósticos locais, sem a necessidade de processamentos externos, garantindo maior eficiência no monitoramento em tempo real.

6.4 Levantamento de Custo do Projeto de TCC

Este tópico aborda o levantamento de custos estimados para a fabricação e montagem dos sensores Bluetooth projetados. O levantamento de custos foi baseado na tabela de **BOM** (Bill of Materials), levando em consideração diferentes quantidades de produção: 1 unidade, 100 unidades, 1000 unidades e 5000 ou mais unidades.

Abaixo, apresentamos uma análise detalhada dos principais componentes utilizados no projeto, com seus respectivos preços conforme a quantidade produzida:

Tabela 1: Tabela de Custo por Quantidade de Produção

Componente	**1 Unid (USD)**	**100 Unid (USD)**	**1000 Unid (USD)**	**5000+ Unid (USD)**
IIS3DWBTR	5,38	3,92	3,08	2,75
100nF	0,44	0,136	0,088	0,056
1.0µF	0,20	0,052	0,034	0,022
10µF	1,00	0,315	0,265	0,155
2.2µF	0,21	0,072	0,046	0,027
NINA-B306-00B	14,35	11,13	10,46	10,46
STTS22HTR	1,93	1,31	0,947	0,799
2.2µH	0,36	0,186	0,156	0,154
1K Resistor	0,51	0,096	0,078	0,045
10K Resistor	0,20	0,016	0,014	0,006
nPM1300-QEAA	3,98	2,93	2,10	1,92
Bateria (batt)	7,70	6,54	6,54	6,54
PCB	1,00	0,61	0,36	0,28
Total (aproximado)	**39,35**	**28,33**	**25,01**	**23,90**

Análise de Custo

1. **1 Unidade**: O custo de produção de uma unidade do sensor Bluetooth é estimado em aproximadamente **39,35 USD**. Este valor é significativamente mais alto devido à baixa

escala de produção e à compra de componentes em pequenas quantidades.

2. **100 Unidades**: Ao aumentar a produção para 100 unidades, o custo unitário cai para **28,33 USD**, uma redução de aproximadamente 28% comparado à produção de uma única unidade.

3. **1000 Unidades**: Para uma produção em larga escala (1000 unidades), o custo unitário reduz para **25,01 USD**. Neste cenário, o preço de componentes críticos, como o módulo **NINA-B306-00B** e o **IIS3DWB**, é consideravelmente reduzido.

4. **5000+ Unidades**: Para produções maiores, acima de 5000 unidades, o custo unitário atinge um valor ainda mais baixo, de **23,90 USD**. Nesse nível de produção, os componentes são adquiridos em grande volume, resultando em maior economia e otimizando o custo total do projeto.

7 Resultados

Os testes realizados demonstraram a capacidade de comunicação dos sensores de temperatura e aceleração por meio de uma rede Mesh Bluetooth, que conectava os dispositivos a um gateway integrado ao computador via USB. Durante os experimentos, as informações enviadas pelos sensores foram recebidas no terminal do computador, indicando a funcionalidade do sistema.

7.1 Análise do funcionamento da rede Mesh

Para validar a performance da rede Mesh, foi realizado um teste utilizando três sensores, dispostos a uma distância inicial de **1 metro** entre cada um e, gradativamente, aumentando essa distância até **15 metros**. O experimento foi projetado para avaliar o comportamento da rede em um ambiente fechado, com paredes e outros obstáculos interferindo na propagação das ondas eletromagnéticas. Durante os testes, foram enviadas **10 mensagens** em cada variação de distância e contabilizadas as mensagens recebidas para analisar a **taxa de perda de pacotes**, um indicador importante da confiabilidade da comunicação em redes sem fio.

A **Tabela 2** apresenta os resultados obtidos. Observa-se que, até **10 metros**, a transmissão ocorreu de forma relativamente estável, com perdas mínimas ou moderadas. No entanto, a partir dessa distância, as perdas aumentaram significativamente devido à atenuação do sinal e aos obstáculos físicos presentes no ambiente. Em 14 metros, por exemplo, nenhuma mensagem foi recebida, indicando a falha completa na comunicação.

Os resultados indicam que a **taxa de perda de pacotes** aumentou progressivamente com a distância. Esse fenômeno pode ser explicado pela atenuação do sinal causada pelas paredes e outros materiais do ambiente, que geraram efeitos de **reflexão, refração e absorção** das ondas eletromagnéticas. Em ambientes fechados, esses fatores reduzem o alcance efetivo da comunicação. Já em ambientes abertos, sem a presença de barreiras físicas significativas, é possível obter resultados diferentes, com maior alcance de transmissão e menores taxas de perda.

Distância (m)	MSG Enviadas	MSG Recebidas	MSG Perdidas	% de Perda de Dados
1	10	10	0	0.0%
2	10	10	0	0.0%
3	10	10	0	0.0%
4	10	10	0	0.0%
5	10	9	1	10.0%
6	10	8	2	20.0%
7	10	8	2	20.0%
8	10	7	3	30.0%
9	10	5	5	50.0%
10	10	4	6	60.0%
11	10	3	7	70.0%
12	10	2	8	80.0%
13	10	1	9	90.0%
14	10	0	10	100.0%
15	10	0	10	100.0%

Tabela 2: Resultados do teste de perda de pacotes em rede Bluetooth Mesh para diferentes distâncias.

Figura 28: Sensor de temperatura e aceleração pullup.

Para conduzir este teste, utilizou-se o modelo de aplicação **Light Switch** da rede Mesh Bluetooth, descrito no Capítulo 4.2 deste projeto. Esse modelo foi essencial para estabelecer o funcionamento básico da comunicação entre os sensores e demonstrar o roteamento de pacotes. Como ilustrado na **Figura 28**, os três sensores foram posicionados próximos para

inicializar o teste antes de aumentar as distâncias.

Além disso, foi verificado que, mesmo quando o terceiro sensor não tinha comunicação direta com o gateway, os dados coletados por ele foram corretamente retransmitidos pelos outros sensores da rede até o terminal do computador. Este comportamento confirmou o funcionamento do **roteamento automático de dados**, um dos principais recursos da tecnologia Mesh Bluetooth. A análise da taxa de perda de pacotes demonstrou a robustez da rede para distâncias menores e a necessidade de otimizações em ambientes com alta interferência para garantir a eficiência da comunicação.

7.2 Análise dos modelos de machine learning

No que tange ao **Machine Learning**, os modelos foram treinados empiricamente e demonstraram capacidade de identificar sinais anômalos em diferentes cenários simulados, como:

- Ventoinha com sensor acoplado na posição vertical (Figura 6).
- Ventoinha com sensor acoplado na posição horizontal (Figura 7).
- Ventoinha com sensor acoplado na posição vertical, mas com desbalanceamento simulado (Figura 8).
- Ventoinha com sensor acoplado na posição vertical, submetida a interferências externas (Figura 9).
- Ventoinha com sensor acoplado na posição vertical, com o fluxo de ar obstruído (Figura 10).

Análise dos Sinais Capturados da Ventoinha para Treinamento de Machine Learning

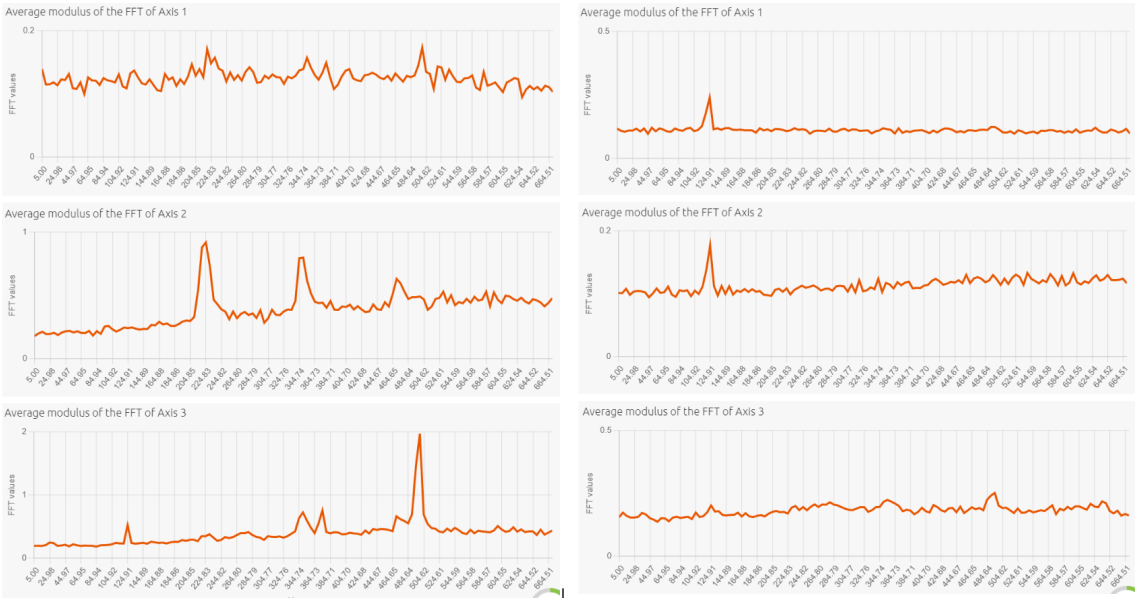
A análise dos sinais vibracionais da ventoinha foi realizada com base na Transformada Rápida de Fourier (FFT), com o objetivo de identificar padrões frequenciais que possam ser utilizados no treinamento de modelos de machine learning voltados para o monitoramento preditivo e a detecção de falhas.

Os testes consistiram na coleta de dados da ventoinha em diferentes condições operacionais, explorando variações na orientação (posição horizontal e vertical) e no estado mecânico (sinal equilibrado e com desbalanceamento). Essas análises permitem construir um conjunto de dados rico em características, que foram usados para ensinar ao modelo como distinguir entre situações normais e anômalas, como o desbalanceamento.

Nos gráficos apresentados, comparam-se as diferenças entre as condições analisadas, destacando como fatores físicos, como gravidade e ressonâncias estruturais, influenciam o comportamento vibracional do sistema.

Análise dos Dados na Posição Horizontal e Vertical

Os gráficos da FFT para a ventoinha rodando em diferentes condições, como na **posição horizontal** (lado direito) e na **posição vertical** (lado esquerdo), mostram diferenças significativas. Essas diferenças resultam de mudanças nas forças físicas atuando sobre o sistema devido à orientação e são importantes para identificar padrões vibracionais específicos que podem ser utilizados no treinamento do modelo.



(a) Ventoinha na posição vertical (b) Ventoinha na posição horizontal

Figura 29: Comparação entre ventoinha em posição vertical e horizontal.

Ventoinha na Posição Horizontal

Quando a ventoinha opera com o eixo de rotação paralelo ao solo, observa-se um comportamento mais estável devido à distribuição uniforme do peso das pás ao longo do eixo.

Principais características observadas:

- Os sinais apresentam menores amplitudes em todos os eixos, com energia vibracional concentrada na frequência fundamental (124 Hz).
- Os harmônicos têm amplitudes muito reduzidas, indicando estabilidade e ausência de ressonâncias significativas.
- O espectro é limpo e bem definido, com baixo nível de ruído, refletindo um funcionamento equilibrado.
- A distribuição uniforme do peso das pás e o atrito mais homogêneo nos rolamentos contribuem para a redução das vibrações.

Ventoinha na Posição Vertical

Na posição vertical, o eixo de rotação está perpendicular ao solo, o que gera forças gravitacionais desiguais e desbalanceamento dinâmico, causando maiores amplitudes em frequências específicas.

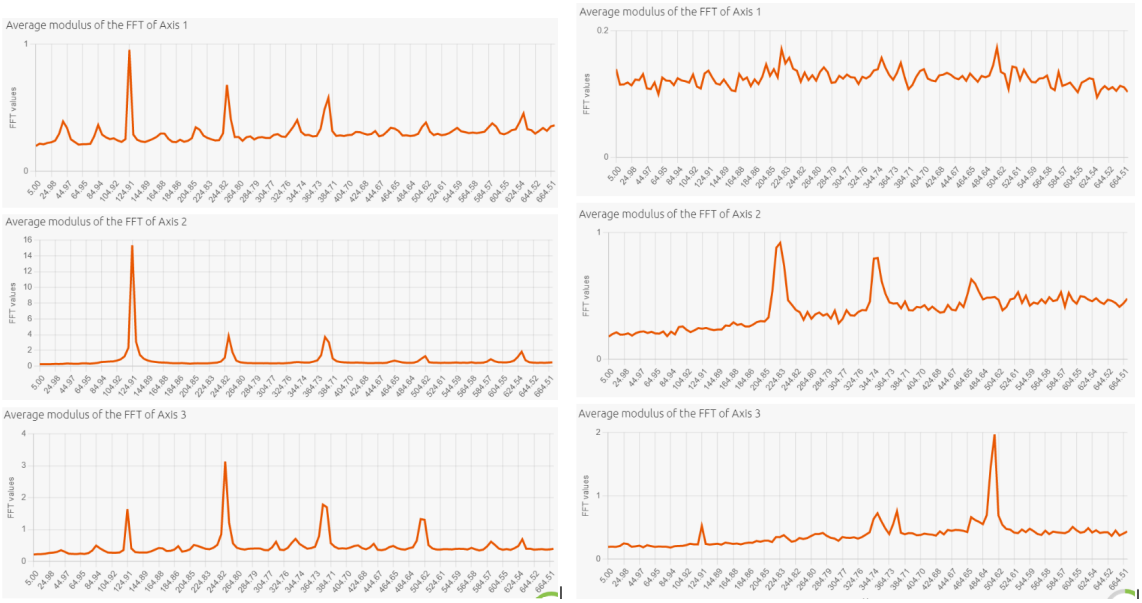
Principais características observadas:

- Apresenta amplitudes significativamente maiores, especialmente nos harmônicos (264 Hz e 384 Hz), indicando maior impacto de ressonâncias e desbalanceamento.
- O espectro é mais ruidoso, com picos adicionais distribuídos por frequências secundárias.
- O eixo de rotação perpendicular ao solo intensifica as vibrações devido à força gravitacional desbalanceada, que aumenta o atrito nos rolamentos e a pressão sobre as pás.
- O eixo 2 é o mais sensível às alterações, apresentando os maiores picos de amplitude.

A comparação entre as duas orientações evidencia que a posição horizontal proporciona maior estabilidade vibracional, com sinais mais limpos e amplitudes reduzidas. Em contraste, a posição vertical amplifica as vibrações, aumentando a energia em harmônicos e ressonâncias devido ao desbalanceamento dinâmico e à influência gravitacional. Essas diferenças fornecemram informações importantes para o treinamento do modelo,pois é necessário levar em consideração a posição da ventoinha para que o modelo aprenda a distinguir padrões de vibração em diferentes orientações.

Comparação Entre Sinal Normal e Sinal com Desbalanceamento

Abaixo estão os gráficos comparando o **sinal com desbalanceamento** (lado esquerdo) e o **sinal normal** (lado direito), ambos com a ventoinha na posição vertical.



(a) Sinal com desbalanceamento (b) Sinal em estado normal

Figura 30: Comparação entre sinais com desbalanceamento e em estado normal.

Sinal em Estado Normal

- Apresenta amplitudes reduzidas e bem distribuídas nos três eixos, com picos suaves e controlados.

- O pico dominante em **124 Hz** (frequência fundamental) possui baixa amplitude, refletindo estabilidade no sistema.
- Os harmônicos ($2f_0 = 248\,\text{Hz}$ e $3f_0 = 372\,\text{Hz}$) são pouco perceptíveis e possuem baixa energia.
- O espectro é limpo e com pouca presença de frequências secundárias, indicando funcionamento equilibrado e sem interferências mecânicas relevantes.

Sinal com Desbalanceamento

- Apresenta amplitudes significativamente maiores, com picos dominantes em **124 Hz** (frequência fundamental) e harmônicos ($2f_0 = 248\,\text{Hz}$ e $3f_0 = 372\,\text{Hz}$).
- No **Eixo 2**, o pico em **124 Hz** atinge valores extremamente altos (≈ 14), evidenciando vibração intensa.
- O espectro possui maior concentração de energia nas frequências harmônicas, com picos bem definidos.
- Vibrações adicionais e ruídos em frequências secundárias sugerem impacto de forças desbalanceadas, refletindo comportamento típico de falhas mecânicas.

O **sinal em estado normal** apresenta comportamento estável, com baixa energia vibracional e espectro equilibrado, indicando ausência de falhas no sistema. Já o **sinal com desbalanceamento** demonstra amplitudes elevadas e picos acentuados em frequências fundamentais e harmônicas, caracterizando vibrações excessivas causadas pela distribuição desigual de massas e forças dinâmicas.

Essa análise destaca o impacto do desbalanceamento no comportamento vibracional e é essencial para o treinamento de modelos de machine learning, permitindo a identificação automática de falhas em sistemas rotativos.

7.3 Validação do machine learning

A validação do modelo de machine learning foi realizada com base nos dados coletados em diferentes cenários, como sinais em estado normal e com desbalanceamento, além das orientações da ventoinha (horizontal e vertical) como pode ser visto nas (Figuras 6 e 7). O objetivo foi avaliar a capacidade do modelo em identificar padrões vibracionais normais e anômalos, simulando falhas mecânicas.

Os gráficos gerados mostram que, ao introduzir uma anomalia no sistema, o percentual de similaridade detectado pelos modelos treinados diminui significativamente. Essa redução reflete a capacidade do modelo de identificar condições anormais, indicando seu potencial para monitoramento e predição de falhas em motores industriais.

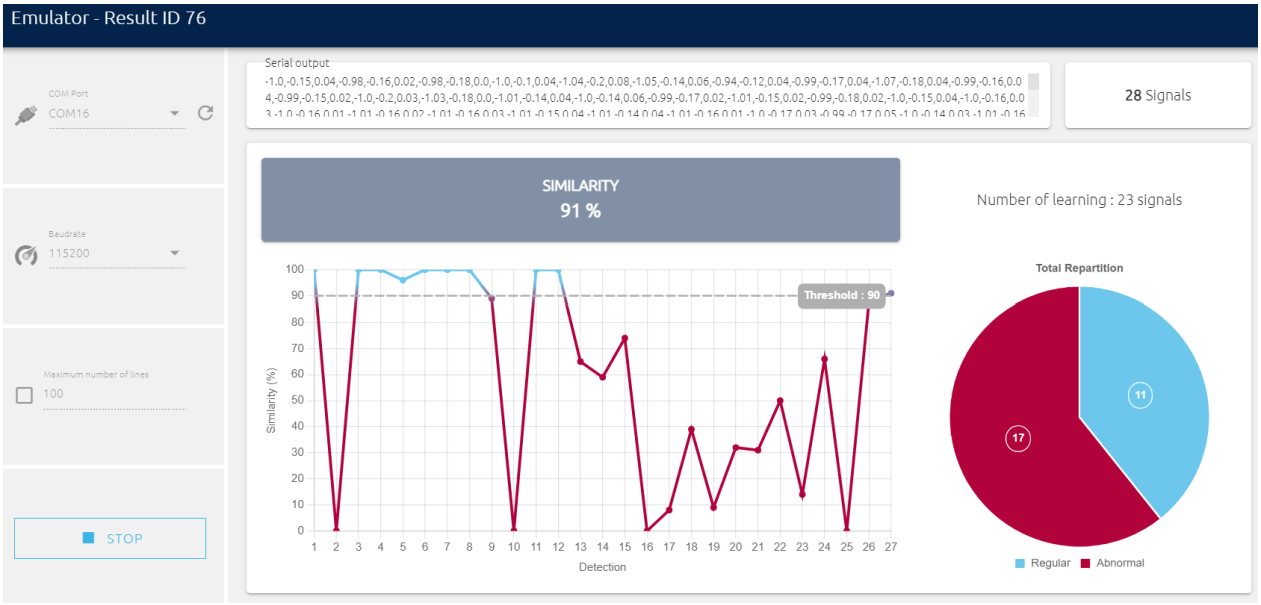


Figura 31: Similaridade do modelo durante os testes

A Figura 31 apresenta os resultados obtidos durante os testes utilizando um modelo de detecção de anomalias. Nesse experimento, o sensor foi colocado em diferentes condições, alternando entre estados normais e anômalos, para verificar se o modelo estava funcionando corretamente. O *NanoEdge AI Studio* gerou os percentuais de similaridade em tempo real durante os testes, acumulando as informações ao longo do tempo e consolidando os resultados no gráfico de pizza.

O gráfico de pizza representou de forma precisa o tempo em que o sensor permaneceu funcionando em condições normais e o período em que esteve em condições anômalas. Durante o teste, o modelo foi capaz de identificar corretamente as mudanças entre esses estados, como evidenciado pela evolução do gráfico de similaridade ao longo do tempo e pela distribuição proporcional apresentada no gráfico de pizza.

O gráfico principal exibe a evolução da similaridade capturada, com os valores em azul representando os momentos de funcionamento ”regular”, enquanto os valores em vermelho indicam as condições detectadas como ”anômalas”. Essa distinção demonstra que o modelo responde dinamicamente às mudanças nas condições do sensor, sendo eficaz na detecção de desvios em tempo real.

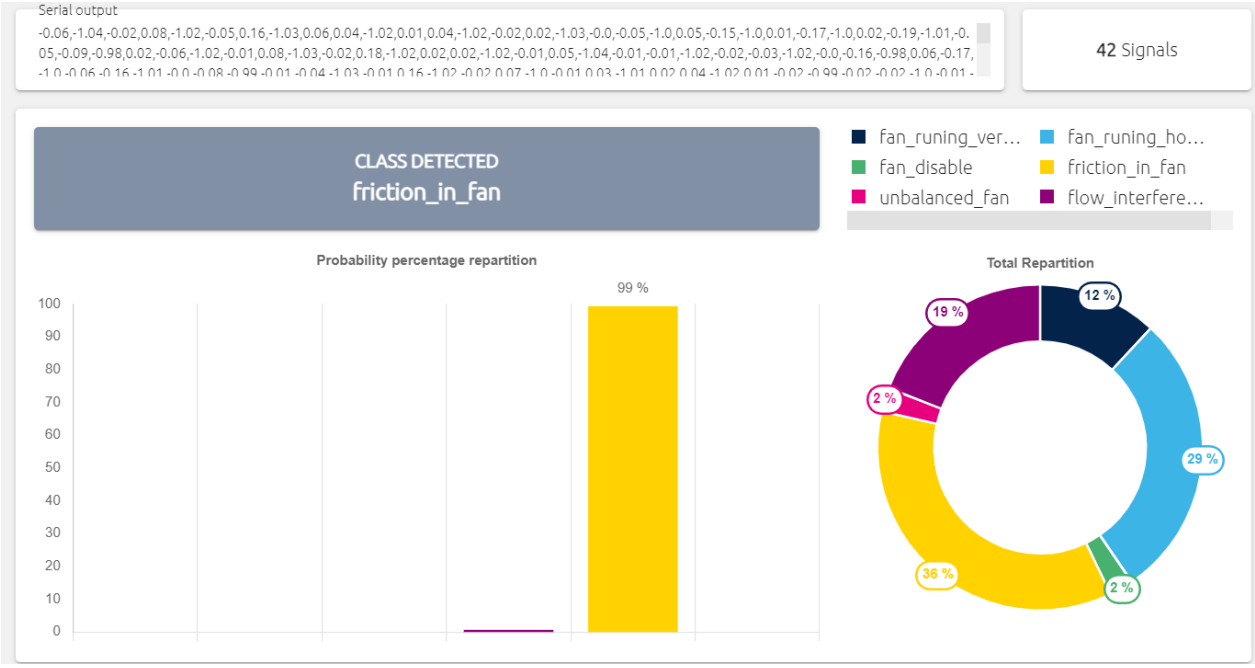


Figura 32: Similaridade do modelo durante os testes

A Figura 32 apresentada foi gerada pelo *NanoEdge AI Studio* durante os testes de assertividade do modelo de classificação, utilizando os dados coletados após o treinamento. Esses testes tiveram como objetivo verificar o desempenho do modelo na identificação correta das condições operacionais e anômalas simuladas.

Durante os experimentos, cada classe foi mantida por aproximadamente **1 minuto** para avaliar se o modelo estava funcionando adequadamente. Os resultados mostraram que

o modelo foi capaz de classificar corretamente as condições gerais, com destaque para a classe **friction_in_fan**, que apresentou uma alta probabilidade de acerto de **99%**, conforme ilustrado pela barra amarela no gráfico de probabilidade percentual. A repartição total dos sinais capturados também foi analisada, evidenciando que classes como **fan_running_vertical** e **fan_running_horizontal** foram detectadas em menor frequência.

No entanto, alguns desafios foram observados:

- **Dificuldade em diferenciar sinais anômalos**: Condições como **unbalanced_fan** e **flow_interference** apresentaram maior sobreposição nos resultados, o que indica uma menor assertividade na identificação dessas classes. Esse comportamento pode estar relacionado à semelhança nos padrões de vibração ou à insuficiência de dados representativos durante o treinamento.

- **Variabilidade limitada no treinamento**: Apesar do esforço em capturar cada classe por aproximadamente 1 minuto, o tempo de captura pode não ter sido suficiente para que o modelo aprendesse características mais robustas e distintivas entre as classes.

Esses resultados indicam que, embora o modelo esteja funcional e apresente bons resultados em algumas classes, como **friction_in_fan**, há espaço para melhorias em sua confiabilidade geral. Para aprimorar o desempenho do modelo, são sugeridos os seguintes ajustes:

- **Aumentar o tempo de captura dos dados de treinamento**: Um maior tempo de coleta para cada classe pode ampliar a diversidade dos sinais coletados, representando melhor as condições operacionais.

- **Adicionar variabilidade nos cenários de treinamento**: Variar as condições de teste, como velocidades, posicionamento do sensor ou outras influências externas, ajudará o modelo a aprender diferenças mais refinadas entre as classes.

- **Revisar os parâmetros de treinamento**: Ajustar parâmetros como o tamanho da janela de análise e a taxa de amostragem pode capturar detalhes adicionais que atualmente podem estar sendo perdidos.

Com essas melhorias, espera-se que o modelo seja capaz de diferenciar com maior precisão as classes atualmente mais ambíguas, tornando-o mais robusto e eficiente para aplicações práticas em tempo real.

8 Melhorias Encontradas

Entre as melhorias potenciais identificadas para o sistema, destacam-se as seguintes:

1. **Criação de um Banco de Dados Temporal:** Uma melhoria significativa seria a implementação de um banco de dados acessível via internet para armazenar e gerenciar as informações coletadas pelos sensores. Um banco de dados temporal poderia ser utilizado para este fim, com hospedagem em uma plataforma na nuvem. Essa abordagem proporcionaria maior organização, segurança e escalabilidade no gerenciamento dos dados.

2. **Desenvolvimento de Dashboards Visuais:** Para tornar a visualização dos dados mais eficiente e intuitiva, seria interessante implementar uma solução que permita a criação de dashboards interativos. Esses dashboards poderiam proporcionar o monitoramento contínuo das variáveis do sistema, facilitando a interpretação e a tomada de decisões por parte dos operadores.

3. **Treinamento com Datasets Públicos:** Como melhoria futura, o treinamento do modelo de Machine Learning pode ser aprimorado com o uso de datasets públicos amplamente reconhecidos. Esses conjuntos de dados possibilitariam um modelo mais robusto e generalizável para diferentes aplicações. Exemplos de datasets incluem:

    - **NASA Ames Research Center**[**27**]: Dados que abordam o desgaste gradual em rolamentos, ideais para monitoramento contínuo e predição de falhas incipientes.
    - **Case Western Reserve University**[**28**]: Conjunto de dados contendo diferentes tipos de falhas simuladas em rolamentos, como:
        - Danos nos elementos rolantes.
        - Danos na pista do anel interno.
        - Danos na pista do anel externo.

- **National Institute of Advanced Industrial Science and Technology (AIST)**[**29**]: Dados focados em condições anômalas, como:
  - Desbalanceamento.
  - Desalinhamento.

O uso desses datasets forneceria uma base confiável para melhorar a capacidade de detecção do modelo, tornando-o mais adaptado a uma ampla gama de cenários.

4. **Automatização do Sistema de Alerta:** Como parte das melhorias futuras, seria interessante desenvolver um sistema de alertas automáticos baseado nos dados coletados. Por exemplo, ao identificar um comportamento anômalo, o sistema poderia enviar notificações para os responsáveis pela manutenção, agilizando as intervenções e reduzindo os riscos de falha.

5. **Integração com Aplicativos Mobile:** Outro aprimoramento seria a criação de um aplicativo móvel que permitisse acompanhar os dados em tempo real e interagir com os sensores. Além disso, propõe-se o desenvolvimento de um aplicativo de provisionamento proprietário para a rede Mesh Bluetooth. Este aplicativo seria responsável por gerenciar as chaves de criptografia da rede de forma remota, permitindo a configuração e manutenção segura da rede Mesh sem a necessidade de dispositivos adicionais. Essa funcionalidade agregaria mobilidade, segurança e maior flexibilidade no uso do sistema.

9 Conclusão

O projeto apresentou resultados promissores quanto à comunicação dos sensores por meio da rede Mesh Bluetooth e à transmissão de dados para um terminal no computador. A integração dos sensores com o gateway foi realizada com sucesso, assim como o treinamento de modelos de Machine Learning para a identificação de condições anômalas em motores.

Embora dificuldades tenham surgido na integração entre os códigos de Machine Learning e a rede Mesh Bluetooth, a nova arquitetura do projeto conseguiu resolver essas limitações, consolidando uma solução funcional e eficiente.

Os resultados obtidos até o momento indicam que o sistema proposto está no caminho certo para atender aos objetivos de monitoramento de motores industriais e análise preditiva de falhas, alinhando-se às demandas da Indústria 4.0. Além disso, o projeto demonstrou que o sensor IoT pode atuar como um dispositivo de *edge computing*, promovendo economia no tráfego de dados e processamento local eficiente.

Por meio deste projeto, foi possível aplicar na prática os conhecimentos adquiridos durante o curso de Engenharia da Informação na Universidade Federal do ABC (UFABC). O projeto permitiu explorar conceitos avançados de sistemas IoT, redes Mesh Bluetooth e aprendizado de máquina, consolidando as competências técnicas desenvolvidas ao longo da formação.

Além disso, a experiência proporcionou um entendimento mais amplo sobre o impacto de tecnologias emergentes na transformação digital de ambientes industriais, preparando-me para os desafios e oportunidades no campo da engenharia e inovação tecnológica.

Referências Bibliográficas

[1] BAERT, M.; ROSSEY, J.; SHAHID, A.; HOEBEKE, J. The Bluetooth mesh standard: An overview and experimental evaluation. *Sensors*, v. 18, n. 2409, 2018.

[2] SINGH, A.; KUKREJA, S.; GANDOMANI, T. J. *Machine Learning for Edge Computing: Frameworks, Patterns and Best Practices*. Routledge, 2020.

[3] RED HAT. O que é edge machine learning? 2023. Disponível em: https://www.redhat.com/pt-br/topics/edge-computing/what-is-edge-machine-learning#:~:text=Uma%20plataforma%20est%C3%A1vel%20e%20comprovada,uma%20nuvem%20h%C3%ADbrida%20mais%20segura.&text=Uma%20plataforma%20de%20aplica%C3%A7%C3%B5es%20para,na%20infraestrutura%20de%20sua%20escolha.. Acesso em: [1 dez. 2024.].

[4] ZHANG, Y.; LI, Z.; WANG, Y. Edge Computing: A New Computing Paradigm for IoT. *Journal of Internet of Things*, v. 1, n. 1, p. 1-9, 2018.

[5] PRIVACY and Bluetooth Mesh. Disponível em: https://blog.nordicsemi.com/getconnected/privacy-and-bluetooth-mesh. Acesso em: [15 set. 2024].

[6] ORACLE. O que é Internet das Coisas (IoT)? Disponível em: https://www.oracle.com/br/internet-of-things/. Acesso em: [15 set. 2024].

[7] SINGH, A.; KUKREJA, S.; GANDOMANI, T. J. *Machine Learning for Edge Computing: Frameworks, Patterns and Best Practices*. Routledge, 2020.

[8] MCKINSEY COMPANY. Unlocking the potential of the Internet of Things. Disponível em:https://www.mckinsey.com/~/media/McKinsey/Industries/Technology%20Media%20and%20Telecommunications/High%20Tech/Our%20Insights/The%20Internet%20of%20Things%20The%20value%20of%20digitizing%20the%20physical%20world/Unlocking_the_potential_of_the_Internet_of_Things_Executive_summary.ashx.Acesso em: [15 set. 2024].

[9] IBM. The value of digitizing the physical world. Disponível em: https://www.ibm.com/downloads/cas/DA5LRXJ3. Acesso em: [15 set. 2024].

[10] IBM. Internet of Things (IoT): Riscos e desafios. Disponível em: https://www.ibm.com/br-pt/topics/internet-of-things. Acesso em: [15 set. 2024].

[11] IBM. What is Edge Computing? Disponível em: https://www.ibm.com/cloud/what-is-edge-computing. Acesso em: [15 set. 2024].

[12] CISCO. What is Edge Computing? Disponível em: https://www.cisco.com/c/en/us/solutions/computing/what-is-edge-computing.html. Acesso em: [Acesso em: 15 set. 2024].

[13] . *Industrial Artificial Intelligence: Technologies and Applications River Publishers Vermesan, O., Wotawa, F., Nava, M. D., & Debaillie, B. (Eds.). (2022)*

[14] Bishop, C. M. (2006), Pattern Recognition and Machine Learning, Springer, ISBN 978-0-387-31073-2 Acesso em: [Acesso em: 15 set. 2024].

[15] NanoEdge AI Studio. *Nano edge especificações*. https://wiki.st.com/stm32mcu/wiki/AI:NanoEdge_AI_Studio#Using_NanoEdge_AI_Studios. Acesso em: 1 dez. 2024.

[16] Girdhar, P. *Practical Machinery Vibration Analysis and Predictive Maintenance*. Newnes, 2004.

[17] Mobley, R. Keith. *An Introduction to Predictive Maintenance*. Butterworth-Heinemann, 2002.

[18] Tractian. *Análise de Vibração: Guia Completo*. https://tractian.com/blog/analise-de-vibracoes. Acesso em: 1 dez. 2024.

[19] The Bluetooth® Special Interest Group (SIG) em: https://www.mathworks.com/help/bluetooth/gs/bluetooth-protocol-stack.html. Acesso em: [Acesso em: 15 set. 2024].

[20] BLUETOOTH SIG. Core Specification Amended 5.4. Disponível em: https://www.bluetooth.com/specifications/specs/core-specification-amended-5-4/. Acesso em: 15 set. 2024.

[21] PHILIPS SEMICONDUCTORS. UM10204 I2C-Bus Specification and User Manual. Rev. 6. 2014. Disponível em: https://www.nxp.com/docs/en/user-guide/UM10204.pdf. Acesso em: 15 set. 2024.

[22] SEGGER. J-Link Debug Probes. 2024. Disponível em: https://www.segger.com/products/debug-probes/j-link/. Acesso em: 15 set. 2024.

[23] USB Implementers Forum. *Universal Serial Bus Specification, Revision 2.0*. 2000. Disponível em: https://www.usb.org/document-library/usb-20-specification. Acesso em: [data de acesso]. em:https://www.usb.org/sites/default/files/Embedded%20USB2%20Version%202.0%20Supplement%20to%20the%20USB%202.0%20Specification.pdf. Acesso em: [data de acesso].

[24] USB Implementers Forum. *Embedded USB2 Version 2.0: Supplement to the USB 2.0 Specification*. Disponível

[25] NFC Forum. *NFC Forum Specifications*. Disponível em: https://nfc-forum.org/build/specifications. Acesso em: [data de acesso].

[26] NFC Forum. *NFC Forum Technical Specifications*. Disponível em: https://nfc-forum.org/uploads/Certification-Files/NFC%20Forum-DevicesRequirements---3.2.00-20221129.pdf?_cchid=165dd7209e697944ee50cf60ebf2e840. Acesso em: [data de acesso].

[27] MORRIS, J. *Microcontrollers: From Assembly Language to C Using the PIC24 Family*. Cengage Learning, 2016 pg-587.

[28] FURBER, S. *ARM System-on-Chip Architecture*. 2nd ed. Addison-Wesley, 2016 pg21-25.

[29] nRF52840. *nRF52840 datasheet*. https://www.nordicsemi.com/-/media/Software-and-other-downloads/Product-Briefs/nRF52840-DK-product-brief.pdf. Acesso em: 1 dez. 2024.

[30] STTS22H . *STTS22H datasheet*. https://www.st.com/resource/en/datasheet/stts22h.pdf. Acesso em: 1 dez. 2024.

[31] *IIS3DWB: Ultra-wide bandwidth, low-noise, 3-axis digital output MEMS vibration sensor. Datasheet.* STMicroelectronics. Disponível em: https://www.st.com/resource/en/datasheet/iis3dwb.pdf. Acesso em: 1 dez. 2024.

[32] nPM1300. *nPM1300 datasheet*. https://br.mouser.com/datasheet/2/297/Nordic_nPM1300_PMIC_PB_v1_2-3402108.pdf. Acesso em: 1 dez. 2024.

[33] PCA10059 . *PCA10059 datasheet*. https://docs.nordicsemi.com/bundle/ncs-1.0.0/page/zephyr/boards/arm/nrf52840_pca10059/doc/index.html. Acesso em: 1 dez. 2024.

[34] *LIS2DH12: MEMS digital output motion sensor. Datasheet.* STMicroelectronics. Disponível em: https://www.st.com/resource/en/datasheet/lis2dh12.pdf. Acesso em: 1 dez. 2024.

[35] ST25DV nfc. *ST25DV datasheet*. https://www.st.com/resource/en/datasheet/st25dv04k.pdf. Acesso em: 1 dez. 2024.

[36] STTS22H . *STTS22H datasheet*. https://www.mouser.com/datasheet/2/297/nrf52840_soc_v3_0-2942478.pdf. Acesso em: 1 dez. 2024.

Appendices

Figura 33: Control Page Hardware

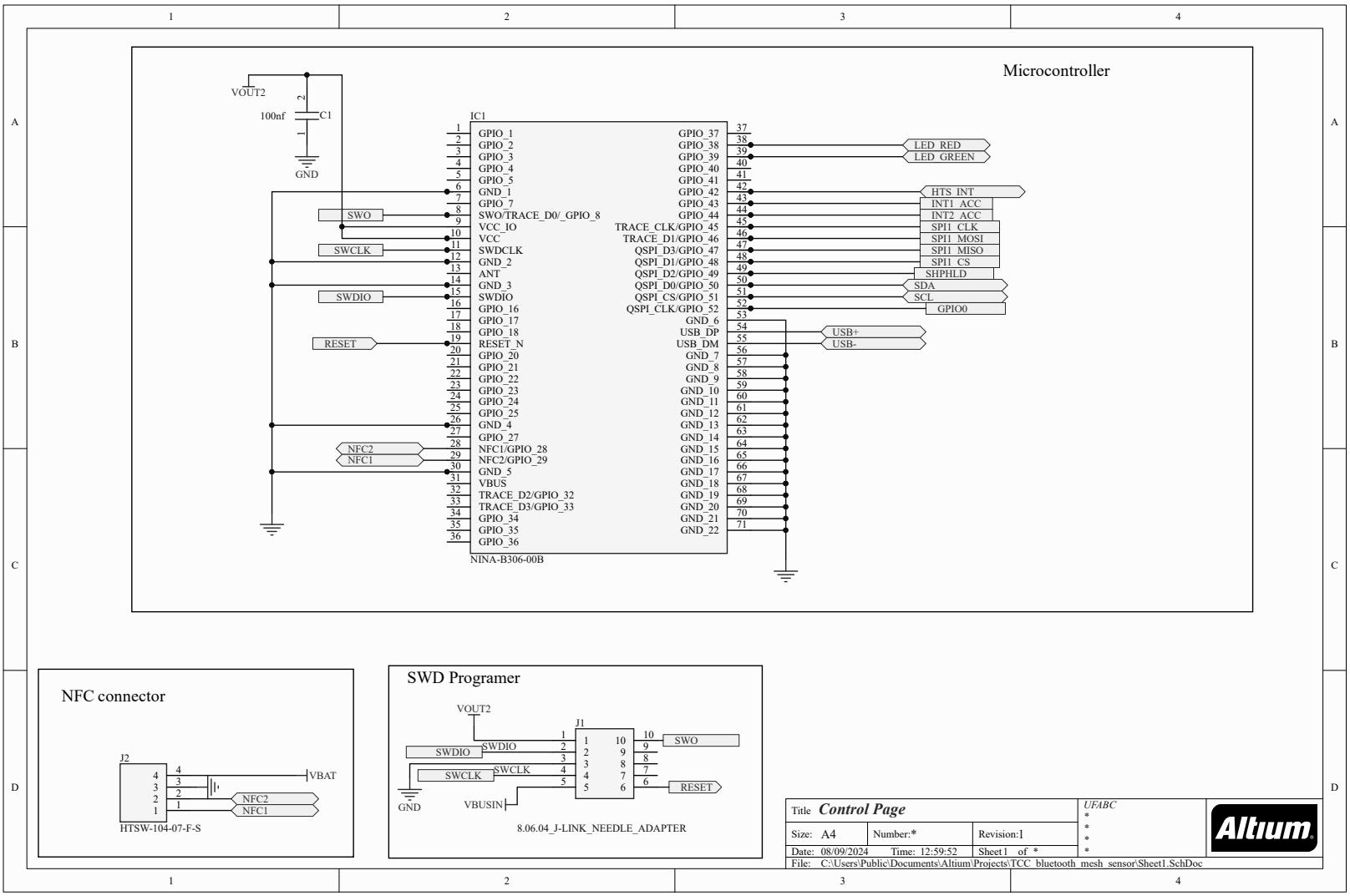


Figura 34: Peripherals Page Hardware

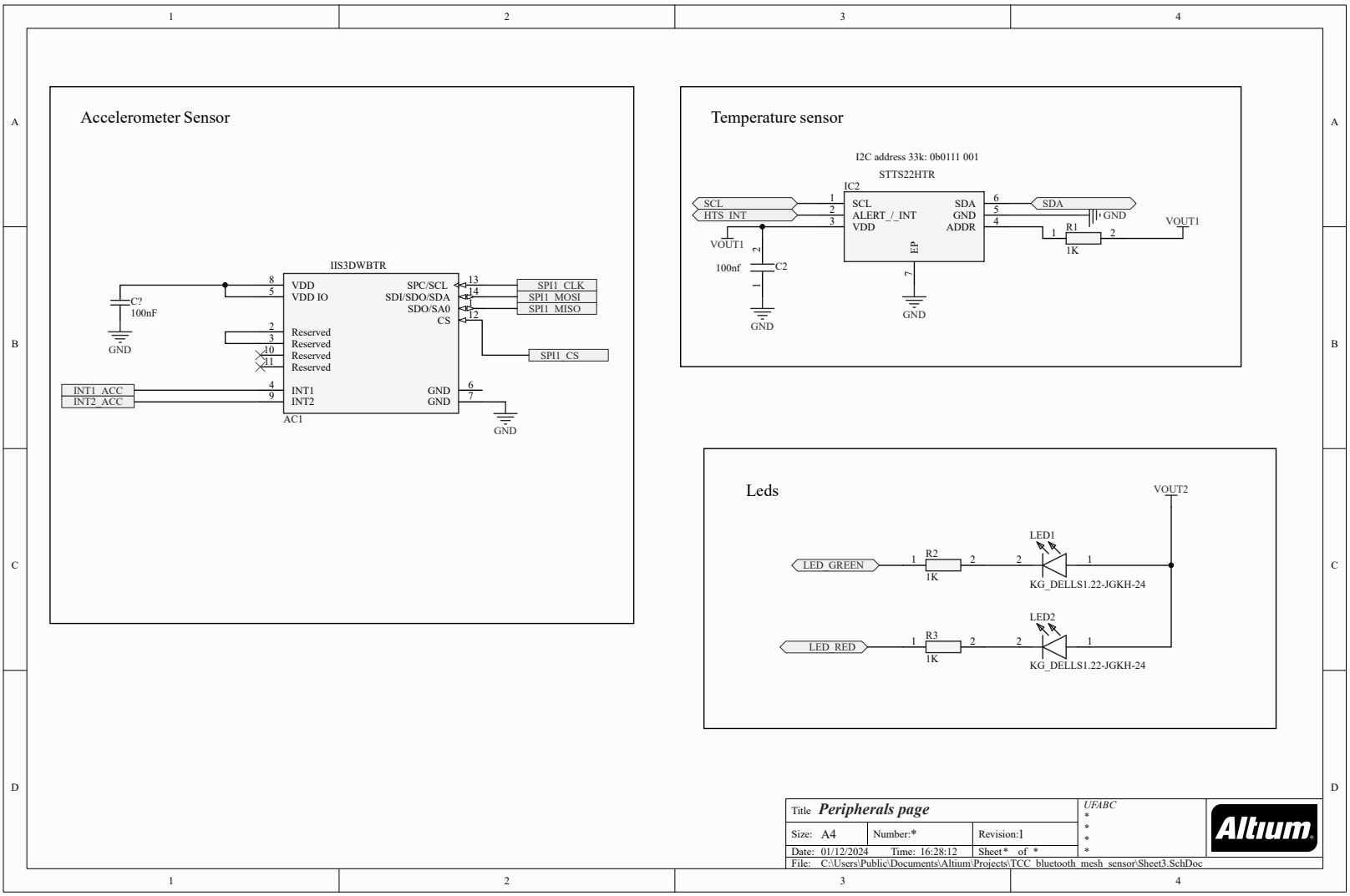


Figura 35: Power Management Page Hardware

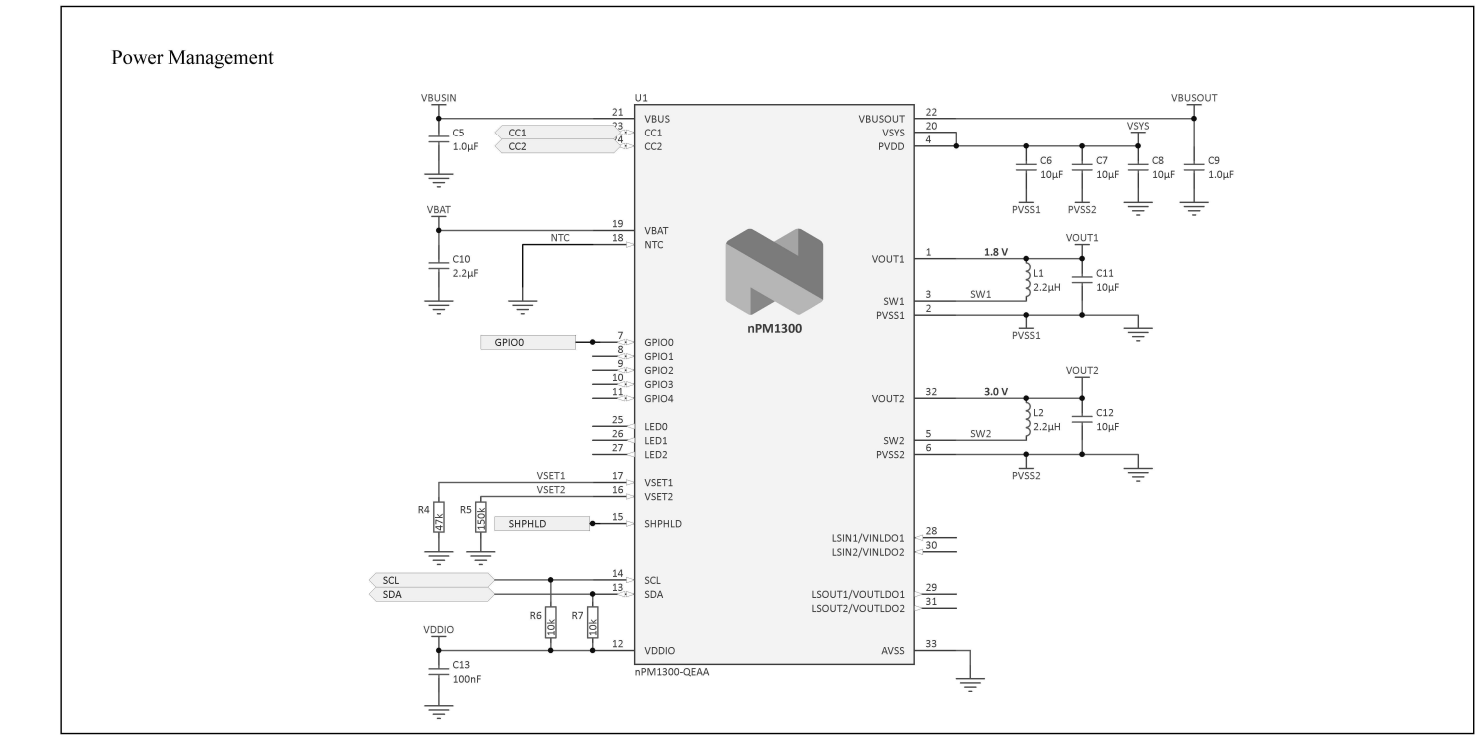


Figura 36: Layout Hardware

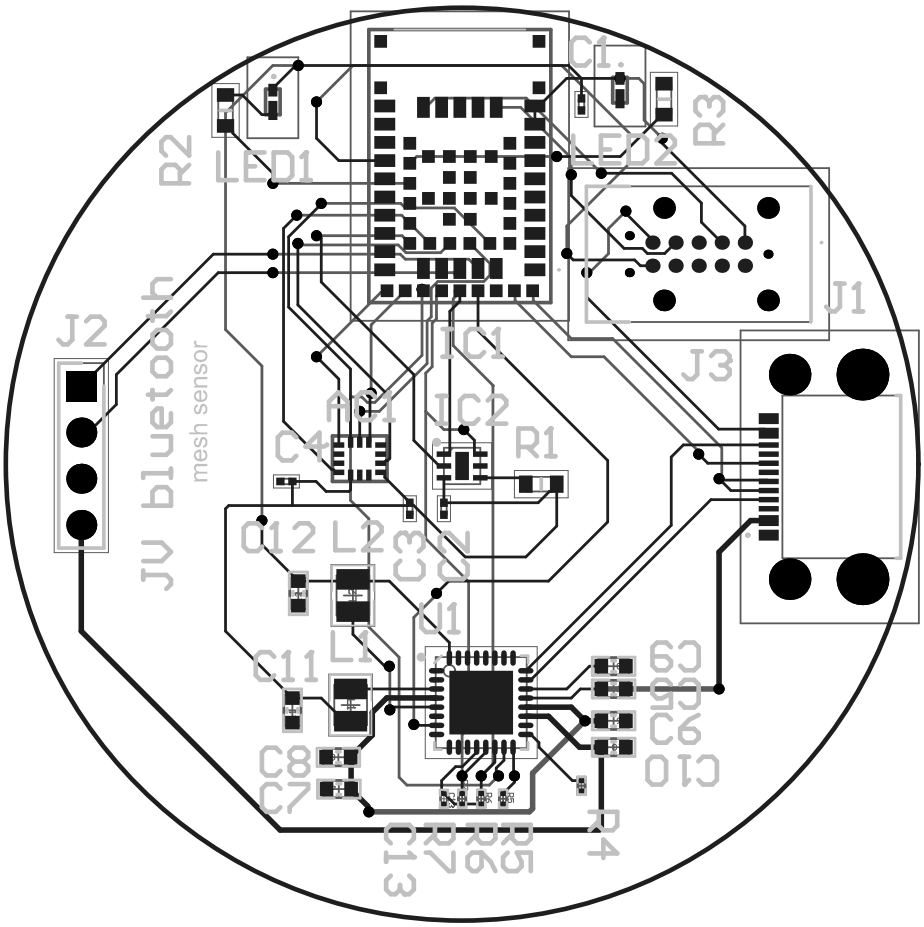


Figura 37: Case Sensor Drawing v4

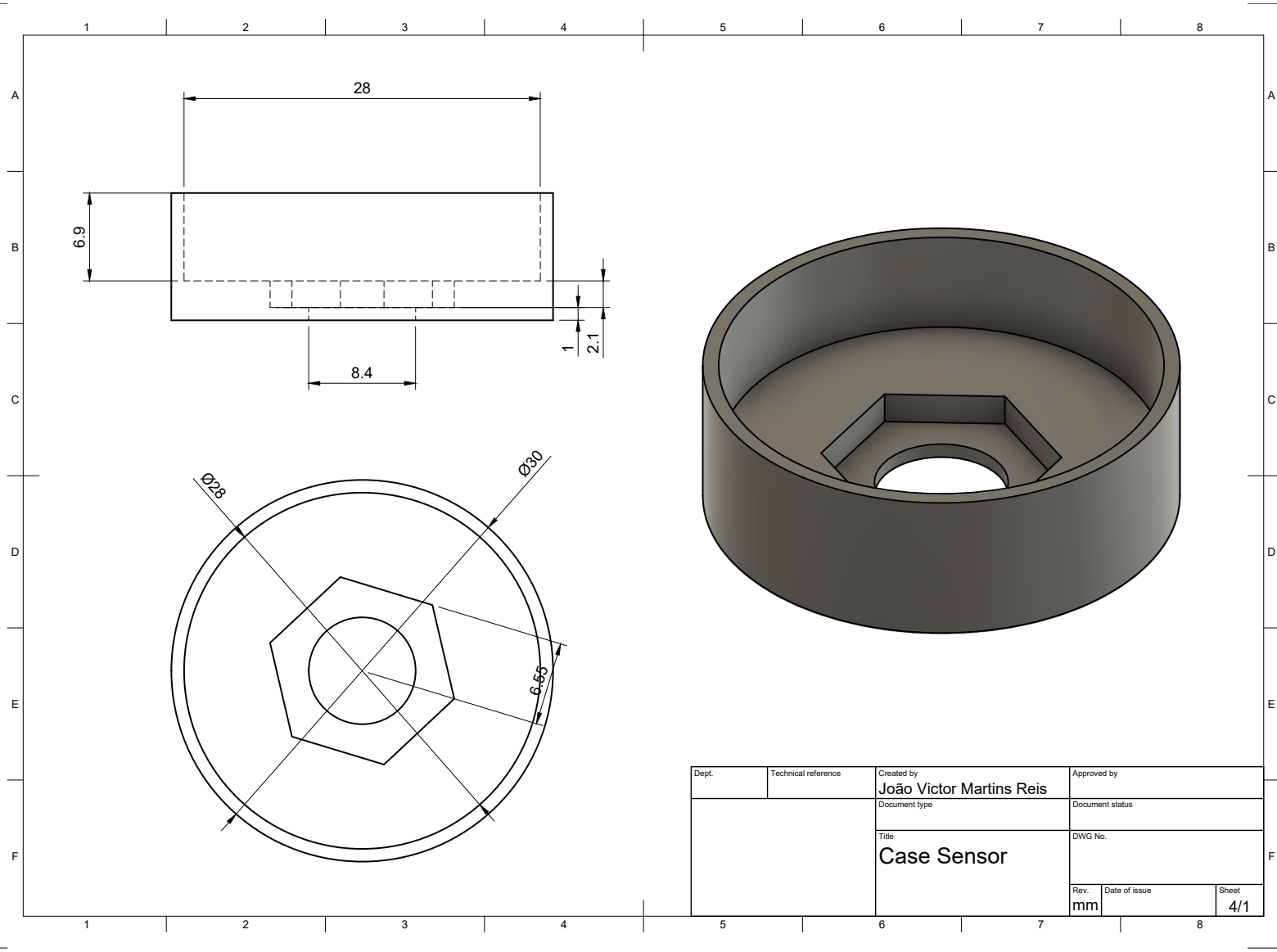
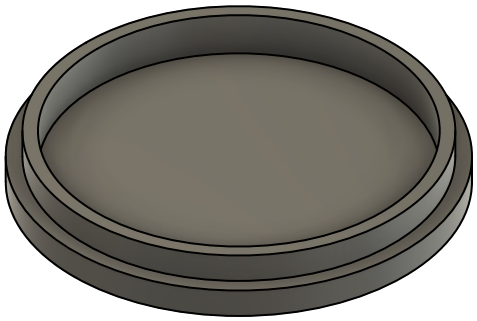
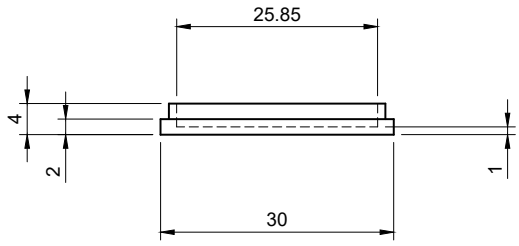
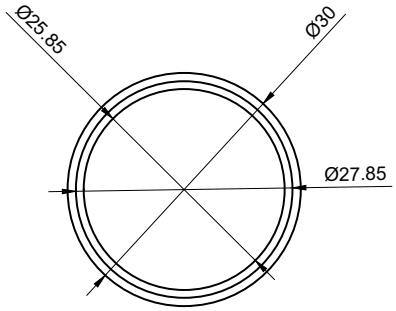


Figura 38: Tampa Encaixada Drawing v3

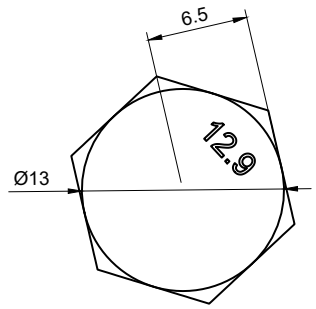


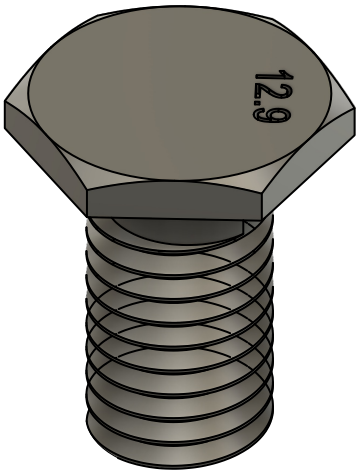


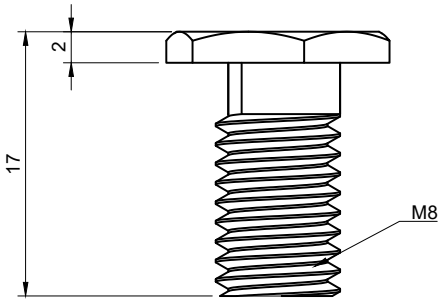


Dept.	Technical reference	Created by	Approved by
		João Victor Martins Reis	
		Document type	Document status
		Title: Case Tampa	DWG No.
			Rev.: mm / Date of issue / Sheet: 2/1

Figura 39: Parafuso v4

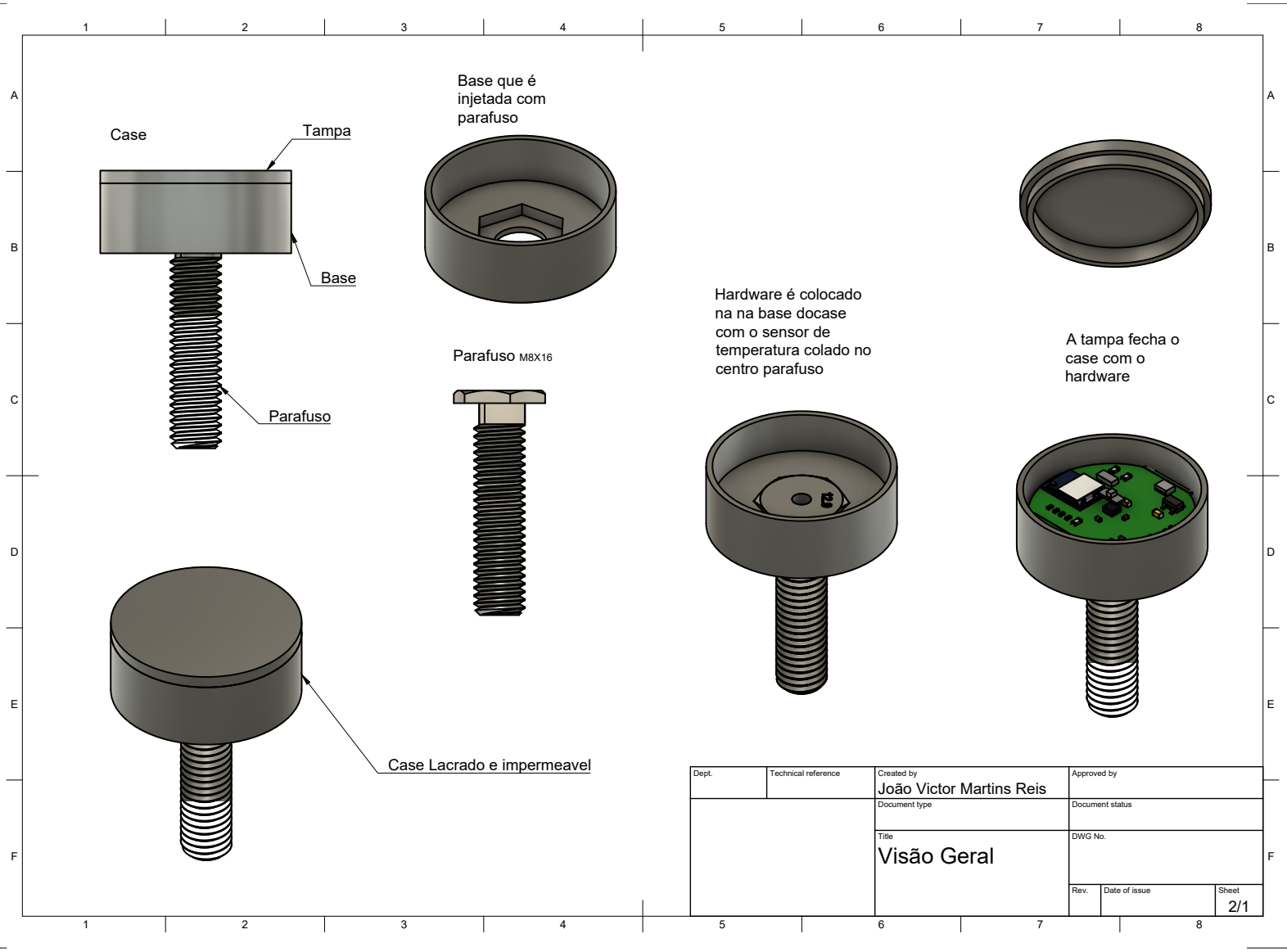






Dept.	Technical reference	Created by	Approved by
		João Victor Martins Reis	
		Document type	Document status
		Title: Parafuso	DWG No. / Scale 4:1
			Rev. mm / Date of issue / Sheet 11

Figura 40: Visão Geral v1



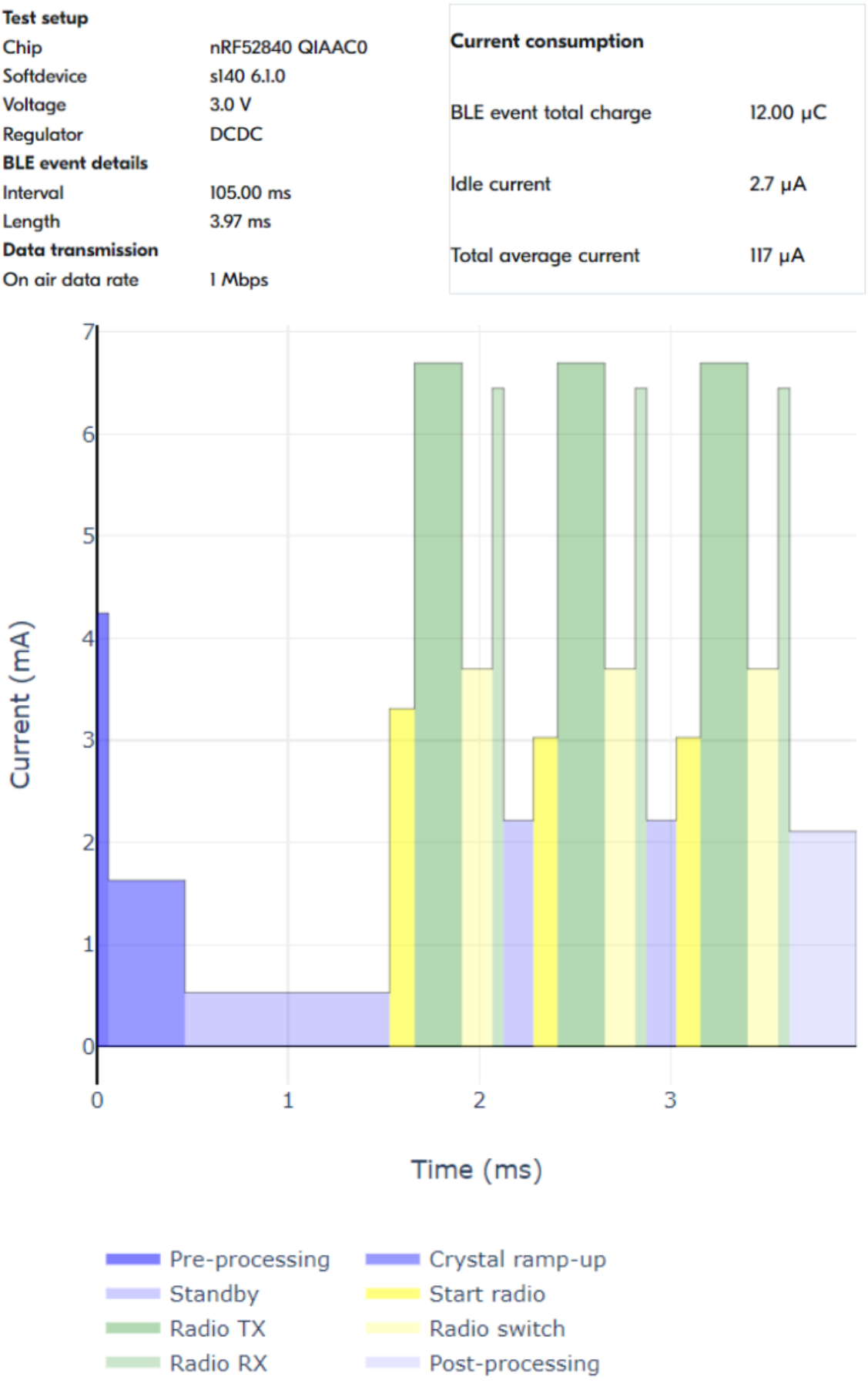


Figura 41: Nordic power consumption calculator