www.revistaft.com.br

Revista FT | ISSN 1678-0817 | v. 30, n. 157, 2026 | DOI: 10.69849/tatdsn13

# Desenvolvimento de um sistema embarcado capaz de mensurar desgaste de componentes em máquinas de estamparia

Development of an embedder system capable of measuring component wear in stamping machines

Arthur da Costa Holanda[1]
Daniel Nascimento[2]
Joanderson Luan da Silva Linhares[3]
Maycon Alexsander da Silva[4]
Mozart Lima do Nascimento[5]
Filipe de Carvalho Pinto Raulino [6]
Ivanilson França Vieira [7]
João Moreno Vilas Boas de Souza Silva [8]

## Resumo

Na indústria, a manutenção do maquinário tem tomado cada vez mais protagonismo dentro das empresas, sendo de suma importância para evitar paradas inesperadas que podem causar perdas na produtividade. Buscando minorar as paradas inesperadas, o presente trabalho busca fazer a detecção antecipada de falhas nos equipamentos. Este trabalho foi realizado em caráter experimental e apresenta um protótipo de uma solução em manutenção preditiva em máquinas de *Silk Screen* industriais. O objetivo é identificar desgastes prematuros em determinados componentes adotando ferramentas da Indústria 4.0 para um monitoramento online do equipamento. O sistema desenvolvido é conectado em rede com servidor local, onde as anomalias podem ser acompanhadas em qualquer terminal alocado no setor de manutenção da fábrica.

[1] Discente do Curso de Formação Inicial e Continuada (Curso FIC) em Residência Tecnológica em Software Embarcado, na modalidade a distância. e-mail: holandaarthur46@gmail.com
[2] Discente do Curso de Formação Inicial e Continuada (Curso FIC) em Residência Tecnológica em Software Embarcado, na modalidade a distância. e-mail: dxnielalves@gmail.com
[3] Discente do Curso de Formação Inicial e Continuada (Curso FIC) em Residência Tecnológica em Software Embarcado, na modalidade a distância. e-mail: joanderson.luan@gmail.com
[4] Discente do Curso de Formação Inicial e Continuada (Curso FIC) em Residência Tecnológica em Software Embarcado, na modalidade a distância. e-mail: mayconalexm@gmail.com
[5] Discente do Curso de Formação Inicial e Continuada (Curso FIC) em Residência Tecnológica em Software Embarcado, na modalidade a distância. e-mail: aluno.mzt.lima@gmail.com
[6] Docente do Curso de Formação Inicial e Continuada (Curso FIC) em Residência Tecnológica em Software Embarcado, na modalidade a distância. e-mail: filipe.raulino@ifrn.edu.br
[7] Docente do Curso de Formação Inicial e Continuada (Curso FIC) em Residência Tecnológica em Software Embarcado, na modalidade a distância. e-mail: ivanilson.junior@ifrn.edu.br
[8] Docente do Curso de Formação Inicial e Continuada (Curso FIC) em Residência Tecnológica em Software Embarcado, na modalidade a distância. e-mail: joao.vilasboas@ifrn.edu.br

**Palavras-chave:** IoT; Manutenção Preditiva; Análise; Monitoramento; Indústria 4.0.

**Abstract**

In the industrial sector, machinery maintenance has increasingly assumed a prominent role within companies, being of paramount importance to prevent unexpected downtime that may lead to productivity losses. In an effort to minimize unforeseen interruptions, the present study aims to perform the early detection of equipment failures. This work was conducted on an experimental basis and presents a prototype solution for predictive maintenance in industrial Silk Screen machines. The objective is to identify premature wear in specific components through the adoption of Industry 4.0 tools for real-time online equipment monitoring. The developed system is network-connected to a local server, allowing anomalies to be monitored from any terminal allocated within the factory maintenance sector.

**Keywords:** IoT; Predictive Maintenance; Analysis; Monitoring; Industry 4.0.

## 1 INTRODUÇÃO

O objetivo da manutenção é sempre maximizar a disponibilidade dos equipamentos, seja nas suas condições originais ou, ao menos, nas condições necessárias para operação. A manutenção preditiva tem como finalidade definir os parâmetros a serem medidos e determinar os limites a serem tolerados. Assim, com essas informações, são tomadas ações para antecipar possíveis falhas mais graves, minimizando o tempo de parada de um equipamento ou planta e diminuindo o custo do reparo (Baldissarelli, 2019).

Nos últimos anos, algumas tecnologias vêm se tornando cada vez mais comuns em ambientes industriais, dentre elas podemos apontar a Internet das Coisas (IoT) e sistemas embarcados. Conforme Wolf (2017), de maneira geral, um sistema embarcado é caracterizado como qualquer dispositivo que incorpora um computador programável, embora seu propósito não seja ser um computador de propósito geral. Sistemas embarcados como sistemas eletrônicos microprocessados cuja função específica, uma vez programada, geralmente não pode ser alterada (Almeida; Moraes; Seraphim, 2021).

A Internet das Coisas, por sua vez, nada mais é que uma extensão da Internet atual, que proporciona aos objetos do dia a dia (quaisquer que sejam), mas com capacidade computacional e de comunicação, se conectarem à Internet (Santos, 2016). Portanto, o uso destas tecnologias pode assumir protagonismos no setor de manutenção em indústrias.

Nesse sentido, o desafio proposto à equipe do Desafio 01 é evitar o desgaste prematuro da longarina, um componente situado na máquina de estamparia (Carrossel), cuja importação

é de alto valor e demorada, de modo a reduzir os custos e o tempo de inatividade com a manutenção corretiva. Com isso, esse desafio torna-se de grande relevância dentro do processo de produtividade na companhia.

Para fins de organização e validação técnica, o desenvolvimento deste projeto foi estruturado em duas etapas distintas: a Fase 1, focada na prototipagem e validação de conceito em ambiente controlado (bancada); e a Fase 2, destinada à implementação industrial com componentes robustos (chão de fábrica).

Ao longo deste documento serão detalhados quais componentes serão adotados, o que são, sua aplicação no protótipo na Fase 1, bem como o comportamento geral do protótipo.

## 1.2 OBJETIVO GERAL

Desenvolver um sistema de monitoramento preditivo para o carrossel de impressão, que é a máquina responsável pelo processo de *Silk Screen* na empresa, de acordo com o modelo de manutenção preditiva.

## 1.3 OBJETIVO ESPECÍFICO

- Modelagem do Sistema e Levantamento de Requisitos
    - Elicitar e documentar requisitos funcionais e não funcionais do sistema, priorizando funcionalidades críticas como leitura de sinais, cálculo de desgaste e geração de alertas.

- Prova de Conceito (Fase 1 – Ambiente Controlado)
    - Selecionar e integrar componentes de prototipagem (ESP32, sensor VL53L0X, Orange Pi) para validar a arquitetura proposta.
    - Implementar o fluxo completo de dados:
      a. Aquisição de sinais de movimento da cabeça de impressão.
      b. Transmissão via protocolo MQTT para um servidor local.
      c. Armazenamento em banco de dados MariaDB.
      d. Visualização em tempo real via dashboard.
    - Realizar testes de validação técnica, incluindo:
      a. Testes unitários e de integração da API.
      b. Verificação de latência e estabilidade da comunicação.

- Migração para Ambiente Industrial (Fase 2 – Implementação Robusta)

- Substituir os componentes de prototipagem por equipamentos industriais:
  a. CLP (Controlador Lógico Programável) no lugar do ESP32.
  b. Sensor de proximidade indutivo no lugar do sensor a laser.
- Adaptar a lógica de programação para linguagem Ladder (MasterTool IEC).
- Definir e testar protocolos de comunicação industrial (MQTT).
- Realizar testes em bancada e em campo para calibrar sensores e ajustar limiares de alerta.

- Validação e Escalabilidade
  - Coletar dados operacionais reais por um período mínimo de três semanas.
  - Ajustar os limiares de desgaste com base em análise estatística e eventos de manutenção registrados.
  - Homologar o sistema com as equipes de TI da fábrica, garantindo segurança, backup e integração à rede industrial.
  - Documentar a solução para replicação em outras máquinas do parque fabril.

## 2 FUNDAMENTAÇÃO TEÓRICA

A manutenção, tradicionalmente, é dividida de acordo com a forma de programação e o objetivo das tarefas a serem executadas (Siqueira, 2014).

As manutenções são divididas em duas classes, a manutenção programada e não programada, na qual as atividades são designadas obedecendo critérios de tempo e condições pré-definidas e executadas em função da necessidade, respectivamente.

**Tabela 1 – Tipos de manutenção**

| | |
|---|---|
| ***Corretiva*** | Visa corrigir uma falha ou o desempenho menor do que esperado de um equipamento dentro do processo de produção de acordo com o que foi definido em seu projeto. Pode ser subdividida em programada e não programada [ABNT, 1994]. |
| ***Preventiva*** | Tem por objetivo reduzir a probabilidade de quebra, evitar a falha ou queda no desempenho, obedecendo a um plano previamente elaborado, baseado em intervalos definidos de tempo [ABNT, 1994]. |
| ***Preditiva*** | É baseada na tentativa de definir o estado futuro de um equipamento ou sistema, por meio dos dados coletados ao longo do tempo por uma instrumentação específica, verificando e analisando a tendência de variáveis do equipamento [ABNT, 1994]. |
| ***Detectiva*** | Realizada em sistemas de proteção, comando e controle, buscando detectar falhas ocultas ou não perceptíveis ao pessoal de operação e manutenção [ABNT, 1994]. |

## 2.2 MICROCONTROLADORES

Microcontroladores são pequenos sistemas computacionais que englobam em um microchip, interfaces de entrada e saída, analógicas e digitais, e periféricos como memória RAM, memória FLASH, interfaces de comunicação serial, conversores analógicos/digitais (ADC) e temporizadores/contadores. Seu propósito consiste em executar tarefas específicas a partir de programas escritos para eles chamados de firmware, armazenados em sua memória (Chase, 2025).

Os microcontroladores inicialmente se resumiam a interfaces de entrada e saída (I/O) e foram agregando, a cada nova versão ou produto, memória RAM, memória EPROM para programas e dados e circuito de oscilador (clock), interfaces de comunicação (serial, USB), interfaces de rede, Ethernet, WiFi e Bluetooth (Oliveira, 2017).

Ainda de acordo com Oliveira (2017), as aplicações de microcontroladores, inicialmente incipientes e voltadas para a indústria, começaram a se propagar nas diversas áreas de atividade humana, essas funcionalidades agregadas aos microcontroladores inicialmente se resumiam a interfaces de entrada e saída (I/O) e foram agregando, a cada nova versão ou produto, memória RAM, memória EPROM para programas e dados e circuito de oscilador (clock), interfaces de comunicação. Assim, um microcontrolador com todas essas funcionalidades integradas permite desenvolver inúmeras aplicações necessitando de pouca energia.

**Tabela 2 – Microcontroladores utilizados na Fase 1**

| | |
|---|---|
| ***ESP*** | A Espressif Systems é uma empresa de semicondutores estabelecida na China em 2008. Focados no desenvolvimento de soluções de comunicação sem fio de ponta, AIoT (Artificial intelligence of things) de baixa potência. [Lima Filho, 2023]. A programação do NodeMCU pode ser realizada através da linguagem LUA, por meio de softwares como:, LuaLoader, ESPlorer e NodeMCU PyFlasher e também é possível utilizar a IDE do Arduino para programar essa placa, o que pode vir a ser uma grande vantagem para quem já está familiarizado esse ambiente desenvolvimento [Espressif,2023]. |
| ***Raspberry Pi*** | A plataforma Raspberry Pi está surgiu de um grupo de cientistas da computação da Universidade de Cambridge liderados por Eben Upton no ano de 2005, com o objetivo de produzir um microcomputador barato, voltado ao ensino de programação a candidatos a uma vaga no curso de Ciência da Computação de Cambridge.<br>O Raspberry Pi pode ser usado como um computador, central multimídia, servidor, ou para a programação de diversas linguagens, entre elas o Python, C/C++, Java, e por possuir portas GPIO é possível que seja usado como componente de projetos de sistemas embarcados [Barbosa, 2017]. |

| | |
|---|---|
| ***Labrador*** | A Labrador é uma Single-Board Computer desenvolvida pela Caninos Loucos — um projeto totalmente brasileiro. Ela é composta por duas placas: a Labrador Core Board, que fornece todo o poder de processamento e memória típicos de um computador moderno, e a Labrador Base Board, que expande as opções de periféricos e comunicações, oferecendo uma grande variedade de conectores.<br>Esse conjunto modular e de fabricação nacional foi projetado para aplicações de Internet das Coisas (IoT), combinando desempenho, versatilidade e a robustez de um hardware concebido no Brasil. |

## 2.3 SENSORES

São dispositivos que mudam seu comportamento sob a ação de uma grandeza física, podendo fornecer diretamente ou indiretamente um sinal que indica esta grandeza. Quando operam diretamente, convertendo uma forma de energia neutra, são chamados transdutores. Os de operação indireta alteram suas propriedades, como a resistência, a capacitância ou a indutância, sob ação de uma grandeza, de forma mais ou menos proporcional (Costa, 2003).

**Tabela 3 – Tipos de sensores**

| | |
|---|---|
| ***Sensores Ativos*** | São aqueles que emitem um tipo de pulso de energia no ambiente e realizam uma estimativa através da energia retornada até eles, como por exemplo os sensores de medida de distância como Sonares e lasers [ALBERTO, 2023]. |
| ***Sensores Passivos*** | Diferentes dos ativos, estes realizam as estimativas através das informações obtidas nas leituras externas disponíveis no ambiente, como por exemplo sensores de câmera, temperatura, calor, umidade, giroscópios e acelerômetros [ALBERTO, 2023]. |

**Tabela 4 – Sensores utilizados no projeto**

| | |
|---|---|
| ***Sensor a laser VL53L0X*** | O sensor de distância VL53L0X é um módulo eletrônico com enorme precisão, sendo capaz de fazer medições de distâncias com mínima margem de erro se comparado a outros sensores existentes. |
| ***Sensor de proximidade indutivo ME12-04BPSZC0S*** | Se caracteriza por detectar objetos mais ferrosos. Funciona em tensão 10-30Vdc, detectando objetos a 4mm de distância. |

## 2.4 CONTROLADOR LÓGICO PROGRAMÁVEL (CLP)

É um dispositivo eletrônico com hardware e software que permite comandar aplicações industriais. Dentre os seus componentes estão o processador, a memória, a fonte de alimentação, os módulos de entrada e de saída e os dispositivos de programação. Softwares específicos desenvolvidos pelos usuários permitem que o CLP possa ser utilizado em aplicações para automação, controle e monitoramento de processos e máquinas de diferentes tipos e complexidades. Ele também é mais resistente que um computador convencional e suporta condições extremas como alteração de temperaturas e ambientes agressivos (Altus, 2025).

## 2.5 ALIMENTAÇÃO

O Suporte Case Porta 2 Pilhas é um dispositivo comumente utilizado em projetos de robótica e tem por finalidade promover a alimentação do dispositivo ou circuito. O Módulo Regulador de Tensão LM2596 trabalha como um conversor DC-DC e é capaz de reduzir uma carga de até 3A. A tensão de saída pode ser ajustada entre 1,5 a 35V, tendo como entrada 3,2 a 40V e pode ser aplicado em circuitos onde a saída de um sensor é superior a 5V. Esses dois componentes foram adotados apenas na Fase 1 do projeto.

## 2.6 COMUNICAÇÃO

MQTT é um protocolo de mensagens padrão da OASIS para a Internet das Coisas (IoT). Ele foi projetado como um transporte de mensagens de publicação/assinatura extremamente leve, ideal para conectar dispositivos remotos com uma pequena quantidade de código e largura de banda de rede mínima.

## 2.7 SOFTWARE

**Tabela 5 – Softwares utilizados na Fase 1**

| | |
|---|---|
| ***Arduino IDE*** | O software de programação utilizado para enviar e gravar o código no ESP32 foi o Arduino IDE, um software de código aberto que auxilia desenvolvedores na escrita e no upload de códigos em placas Arduino.<br>Sua interface é responsiva e moderna, contando com recursos de gerenciamento de placas e bibliotecas, monitor serial, ferramentas de edição e compilação de código, além de suporte às linguagens C e C++.<br>Apesar de ser pensado para placas Arduino, o microcontrolador ESP32 pode ser programado no Arduino IDE mediante a utilização de uma biblioteca |

| | |
|---|---|
| | fornecida pela própria Espressif.(RODRIGUES, 2024) |
| ***MariaDB*** | O MariaDB é um sistema de gerenciamento de banco de dados relacional de código aberto, ele permite gerenciar os relacionamentos predefinidos entre dados, no qual os dados são organizados como um conjunto de tabelas, colunas e linhas.(MARIADB.ORG, 2025) |
| ***Mosquitto*** | É um broker de mensagens de código aberto (licenciado sob EPL/EDL) que implementa as versões do protocolo MQTT.<br>O Mosquitto é leve e adequado para uso em todos os tipos de dispositivos, desde computadores de placa única de baixo consumo até servidores completos.(MOSQUITTO.ORG, 2025) |
| ***Vue.js e Ecossistema*** | Framework JavaScript progressivo utilizado para o desenvolvimento da interface de usuário (Dashboard).<br>A implementação adotou a arquitetura de Aplicação de Página Única (SPA), utilizando TypeScript para tipagem estática e Pinia para gerenciamento de estado, garantindo reatividade na visualização dos dados em tempo real. |

## 2.8 TRABALHOS RELACIONADOS

A integração entre manutenção preditiva e tecnologias da Indústria 4.0 tem sido amplamente estudada na academia e aplicada na indústria. Um trabalho relevante nesse contexto é o de Souza *et al*. (2022), que investigou a *"Utilização das tecnologias da indústria 4.0 na manutenção preditiva através do monitoramento de equipamentos e instalações"*. Os autores realizaram um estudo de caso dividido em duas fases:

- Validação em empresa parceira: Analisaram o funcionamento de um sistema preditivo comercial (fornecido pela empresa Semeq) em uma estação de tratamento de água. O sistema utilizava sensores sem fio para monitorar vibração e temperatura, com transmissão de dados via Bluetooth para um *gateway* e posterior envio à nuvem. Um *dashboard* (My Semeq APP) permitia a visualização dos dados e alertas.
- Implementação em ambiente universitário: Instalaram o mesmo sistema em motores de bombas de resfriamento do Centro Universitário. O estudo demonstrou benefícios, como a otimização das atividades da equipe de manutenção e o aumento da confiabilidade do sistema, detectando precocemente uma falha por ressonância em rolamentos.

Este trabalho relacionado estabelece um precedente claro para a viabilidade de sistemas de monitoramento baseados em IoT e sensoriamento remoto para manutenção preditiva, validando o conceito em um ambiente real e não industrial.

Embora a solução apresentada por Souza *et al*. (2022) seja robusta e comercialmente disponível, ela representa uma abordagem verticalizada e proprietária. Sua arquitetura é

fechada, dependente de sensores, *gateways* e software específicos de um único fornecedor (Semeq). Isso pode implicar em um alto custo de implantação e escalabilidade, dificuldade de personalização para problemas específicos de diferentes máquinas e dependência do fornecedor.

## 2.9 ESTADO DA ARTE

O presente trabalho se posiciona de forma complementar e inovadora no estado da arte ao propor uma solução horizontal, modular e de código aberto. Nossa abordagem, detalhada nas Fases 1 e 2, justifica-se pelos seguintes diferenciais:

- Foco em um problema específico: Enquanto o trabalho relacionado monitora parâmetros gerais de saúde em motores rotativos, nosso sistema foi concebido para resolver um problema específico de desgaste por deslocamento linear em máquinas de estamparia (pastilhas e casquilhos da longarina). Desenvolvemos uma métrica direta correlacionada ao desgaste do componente, o que pode gerar alertas mais precisos e acionáveis para a equipe de manutenção.
- Arquitetura aberta e de baixo custo (Fase 1 - Prova de Conceito): Utilizamos componentes de prototipagem amplamente disponíveis e de baixo custo (ESP32, sensor VL53L0X, Orange Pi/Labrador), protocolos abertos (MQTT) e desenvolvemos nosso próprio *backend* (API Node.js) e *frontend* (Dashboard Vue.js). Isso demonstra que é possível construir um sistema funcional de monitoramento preditivo sem depender de soluções caras e fechadas.
- Caminho claramente definido para a industrialização (Fase 2 - Implementação Robusta): Nosso projeto avança além da prova de conceito ao detalhar um plano de migração para componentes industriais. A substituição do ESP32 por um CLP e do sensor a laser por um sensor de proximidade indutivo robusto demonstra uma preocupação com confiabilidade, segurança e integração ao parque fabril existente.
- Desenvolvimento de software personalizado e testado: Desenvolvemos uma API e um Dashboard sob medida para a necessidade do problema. A implementação de testes unitários, análise estática de código garante a qualidade e a manutenibilidade do *software*, um aspecto crítico muitas vezes negligenciado em protótipos.

Em síntese, este trabalho se alinha ao estado da arte ao confirmar a importância da IoT e do sensoriamento para manutenção preditiva, possui uma abordagem alternativa e evolutiva. Ele parte de uma prova de conceito acessível e replicável, utilizando tecnologias abertas, e traça um caminho prático e detalhado para sua transformação em uma solução robusta e específica para um problema de desgaste mecânico.

Enquanto o trabalho relacionado valida o *conceito* de monitoramento remoto, nosso trabalho avança no *método* de desenvolvimento e na aplicação específica para estamparia, podendo ser adaptado para outras máquinas e tipos de desgaste.

# 3 METODOLOGIA

Projetos podem se beneficiar com o uso de algum tipo de modelagem, pois modelos auxiliam, para ter uma visão mais abrangente, do funcionamento de um sistema. Tornando possível desenvolver os projetos, de forma mais rápida e correta, pois, quanto mais complexo for o sistema, maior será a probabilidade de ocorrência de erros, caso tenha sido elaborado sem uma modelagem prévia (Chwif *et al*., 2010).

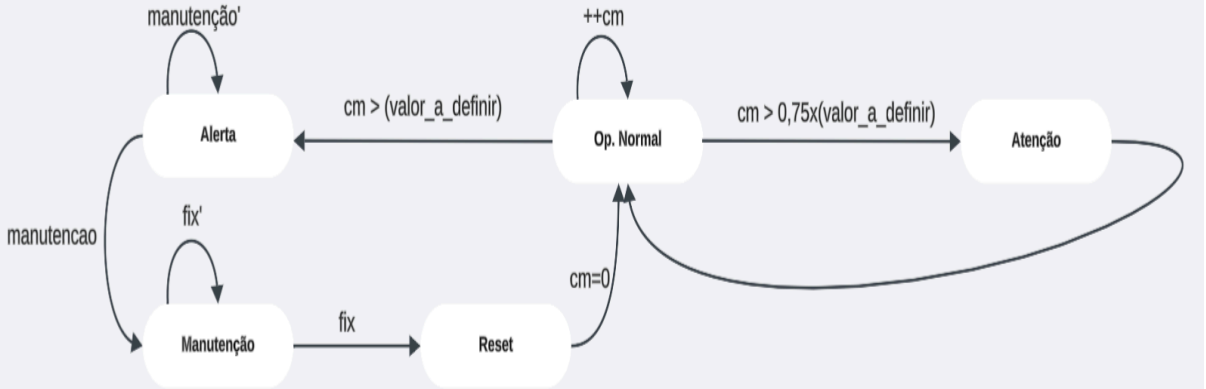
Nessa direção, o projeto baseou-se na criação de um diagrama de estados finitos. Posteriormente, realizou-se a coleta dos Requisitos Funcionais e Requisitos Não Funcionais que o projeto deveria possuir. Neste momento, os residentes se dedicaram ao estudo da problemática.

A partir dessas modelagens, iniciou-se a Fase 1, que consistiu na seleção de componentes para a elaboração do protótipo. Esses itens foram concedidos pelo Laboratório de Pesquisa Allyson Amilcar Angelus (LAICA) da instituição de ensino.

Com a Fase 1 concluída, o passo seguinte foi o desenvolvimento do protótipo com componentes mais robustos, componentes esses que são amplamente empregados em soluções industriais e que foram fornecidos pela Guararapes Confecções para o desenvolvimento do protótipo.

## 3.1 AUTÔMATO DO SISTEMA

**Figura 1 – Modelagem por meio de um autômato**




## 3.2 REQUISITOS FUNCIONAIS (RF)

**Tabela 6 – Requisitos Funcionais**

| ID | Requisitos Funcionais | Prioridade |
|---|---|---|
| **RF01** | O sistema deve ler os sinais digitais que indicam o avanço e o recuo da cabeça de impressão | Alta |
| **RF02** | O sistema deve ler o sinal analógico do potenciômetro que indica a velocidade. | Alta |
| **RF03** | O sistema deve calcular a distância percorrida a cada ciclo com base nos dados coletados (RF01) | Alta |
| **RF04** | O sistema deve armazenar a distância acumulada e o timestamp de cada ciclo. | Alta |
| **RF05** | O sistema deve disparar um alerta quando a distância acumulada atingir um limiar pré-configurado (ex: 80% da vida útil). | Alta |
| **RF06** | O sistema deve apresentar em um dashboard a distância total acumulada desde a última troca e o percentual de desgaste. | Média |

## 3.3 REQUISITOS NÃO FUNCIONAIS (RNF)

**Tabela 7 – Requisitos Não Funcionais**

| ID | Requisito Não Funcionais | Métrica |
|---|---|---|
| **RNF01** | **Desempenho:** O sistema deve processar e enviar os dados de cada ciclo em, no máximo, 5 segundos após o término do ciclo. | Latência < 5s |
| **RNF02** | **Confiabilidade:** O sistema deve operar continuamente, com uma disponibilidade mínima de 99.5%. | Uptime > 99.5% |
| **RNF03** | **Segurança:** A comunicação Wi-Fi deve utilizar o protocolo de segurança WPA2 ou superior. | Criptografia WPA2 |
| **RNF04** | **Usabilidade:** O dashboard deve ser acessível via navegador web padrão e exibir as informações de forma clara e objetiva. | Acesso via Chrome/Firefox |

## 3.4 COMPONENTES E ARQUITETURA DO SISTEMA NA FASE 1

Após visitas à empresa e conversas com a equipe de manutenção, foi realizado o levantamento de requisitos funcionais e não funcionais para iniciar a Fase 1 do projeto.

Esta seção apresenta detalhadamente os componentes de hardware utilizados na primeira fase do projeto, descrevendo sua função dentro da solução de monitoramento preditivo, bem como as ferramentas de software adotadas para processamento, transmissão e visualização dos dados.

A Fase 1 teve como principal objetivo validar a viabilidade técnica do conceito proposto por meio da criação de um protótipo funcional capaz de monitorar o movimento da cabeça de impressão na longarina, registrar ciclos de operação e estimar o desgaste dos componentes mecânicos.

Para isso, optou-se pelo uso de componentes eletrônicos de prototipagem, que oferecem flexibilidade e baixo custo, sem comprometer a representatividade dos resultados preliminares.

### 3.4.1 Hardware

**Tabela 8 – Componentes utilizados**

| | |
|---|---|
| ***ESP32*** | Tem a responsabilidade de coletar dados do sensor e/ou contato seco, processar esses dados e enviar via Wi-Fi. ESP32 lê o valor analógico do sensor e o converte em centímetros, assim como gerencia a conexão Wi-Fi e o protocolo de comunicação MQTT para envio dos dados semi-processados ao Orange Pi. |
| ***Orange Pi*** | Recebe os dados provenientes do ESP32 via Wi-Fi. O Orange Pi lê o dado encapsulado e enviado que foi enviado por meio do MQTT via Wi-Fi para o gateway (roteador). |
| ***Labrador*** | Este dispositivo atua como servidor. Contém o Mosquitto (broker), o servidor web (NGinx), o frontend (HTML) e a API (NodeJS). |
| ***Sensor VL53LOX*** | No protótipo proposto, esse dispositivo deve ser capaz de medir a distância total de deslocamento da cabeça de impressão. |
| ***Regulador Tensão LM2596*** | Terá a incumbência de ajustar a tensão de trabalho para o ESP32, fornecendo para ele uma tensão de 3,3V. |

### 3.4.2 Softwares e Tecnologias

**Tabela 9 – Softwares e tecnologias utilizadas**

| | |
|---|---|
| ***IDE/ C++*** | A IDE adotada para essa fase do projeto é a oferecida pela Arduino. O critério adotado para sua escolha foi por sua simplicidade e seus recursos. Ainda, a linguagem de programação adotada foi a linguagem C++.<br>Essa opção se deu pela familiaridade que os residentes ganharam durante a residência, garantindo a celeridade no desenvolvimento da solução nesta fase. |
| ***Mosquitto*** | É o intermediador da comunicação entre dispositivos e o servidor.<br>Ele recebe as mensagens do ESP32 e as distribui para quem estiver inscrito no tópico correspondente. |
| ***API*** | A API é o centro de integração, responsável por manipular os dados vindos do *subscriber* e disponibilizá-los para visualização.<br>Sua tarefa é salvar esses dados no banco de dados e oferecer *endpoints* para dashboards/aplicações. |

| | |
|---|---|
| ***Figma*** | Ferramenta utilizada para a prototipagem de alta fidelidade da interface (Dashboard), permitindo a definição da experiência do usuário (UX) antes da codificação. |
| ***Vue.js e Ecossistema*** | Framework JavaScript progressivo utilizado para o desenvolvimento da interface de usuário (Dashboard).<br>A implementação adotou a arquitetura de Aplicação de Página Única (SPA), utilizando TypeScript para tipagem estática e Pinia para gerenciamento de estado, garantindo reatividade na visualização dos dados em tempo real. |

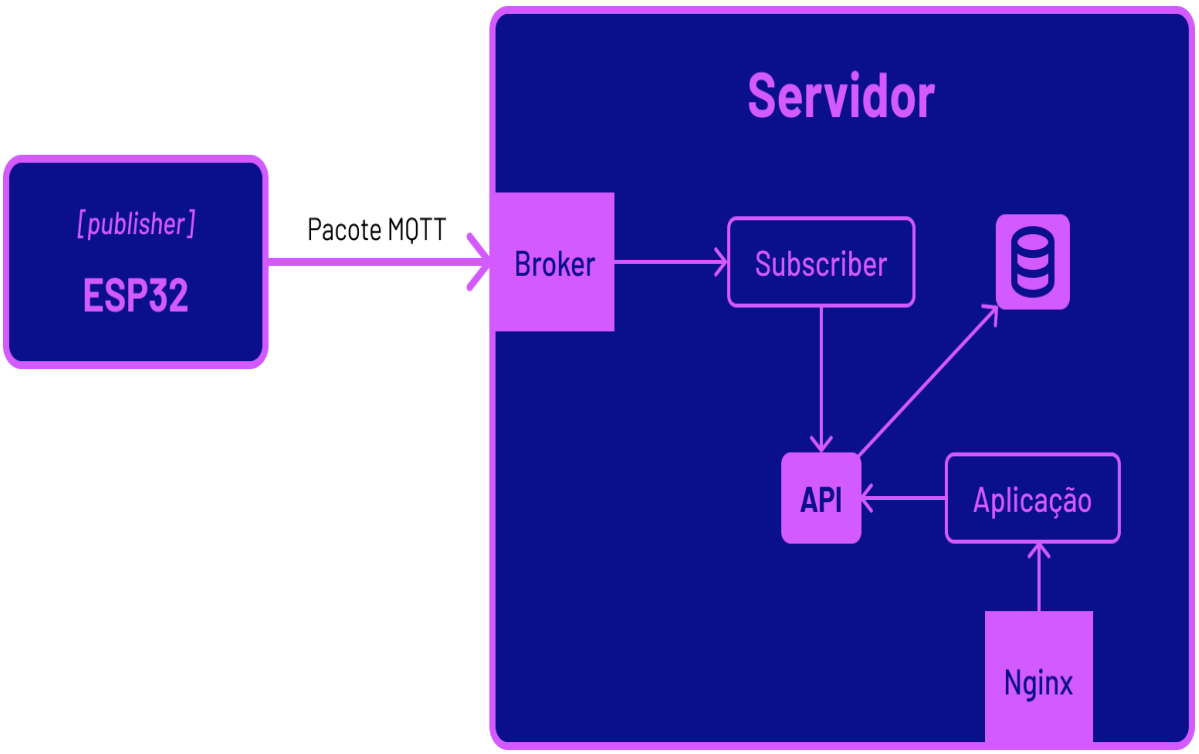
### 3.4.3 Fluxo do Sistema

**Tabela 10 – Arquitetura final na Fase 1**

| Etapa | Ação |
|---|---|
| ESP32 → MQTT | Envio de dados dos sensores |
| Broker MQTT → Subscriber | Entrega da mensagem |
| Subscriber → API | Registro dos dados |
| API → Banco | Armazenamento dos dados |
| Aplicação → API | Consulta e exibição |
| Usuário → Interface | Monitoramento em tempo real |

### 3.4.4 Diagrama do Sistema

**Figura 2 – Diagrama da arquitetura final na Fase 1**




### 3.4.5 Estratégias de Teste, Validação e Qualidade de Software

Para garantir a robustez do sistema desde as etapas iniciais de desenvolvimento do backend, foram empregadas estratégias de testes unitários. Esta abordagem foca na validação isolada dos menores componentes do software - neste caso, as funções de roteamento (*handlers*) e a lógica de negócios.

Com o uso de ferramentas adequadas e técnicas de *mocking* (simulação), foi possível verificar o comportamento de cada função diante de diferentes cenários de entrada (sucesso, erro de cliente, erro de servidor). Isso assegurou que os blocos de construção fundamentais do código estivessem funcionais antes de interações mais complexas.

A Figura 3 apresenta o resultado da execução da suíte de testes, demonstrando o sucesso nas operações de criação, leitura e validação de erros nas rotas da API.

**Figura 3 – Resultado da execução dos testes de integração no terminal**

```
PASS tests/handlers.test.js
  Handler Principal (requestListener)
    ✓ Deve retornar 404 para rota inexistente (2 ms)
    ✓ Deve lidar com OPTIONS (CORS)
    ✓ Deve capturar exceções e retornar 500 (ou erro customizado) (17 ms)
    Recurso: Carrosseis
      ✓ GET /carrosseis - Deve retornar lista (1 ms)
      ✓ POST /carrosseis - Deve criar item
      ✓ GET /carrosseis/:id - Deve retornar 404 se não existir (1 ms)
      ✓ DELETE /carrosseis/:id - Deve deletar se existir
    Recurso: Cabecas
      ✓ PATCH /cabecas/:id - Deve adicionar distância corretamente
      ✓ PATCH /cabecas/:id - Deve falhar se input for inválido (1 ms)
      ✓ PUT /cabecas/:id - Deve falhar com body vazio

Test Suites: 1 passed, 1 total
Tests:       10 passed, 10 total
Snapshots:   0 total
Time:        0.112 s, estimated 1 s
Ran all test suites.
```

**Figura 4 – Trecho do código de testes unitários**

```
test('GET /carrosseis - Deve retornar lista', async () => {
    const mockData = [{ id: 1, img: 'a.jpg' }];
    models.Carrossel.getAll.mockResolvedValue(mockData);

    const req = httpMocks.createRequest({ method: 'GET', url: '/carrosseis' });
    const res = httpMocks.createResponse();

    await requestListener(req, res);

    expect(res.statusCode).toBe(200);
    expect(JSON.parse(res._getData())).toEqual(mockData);
});
```

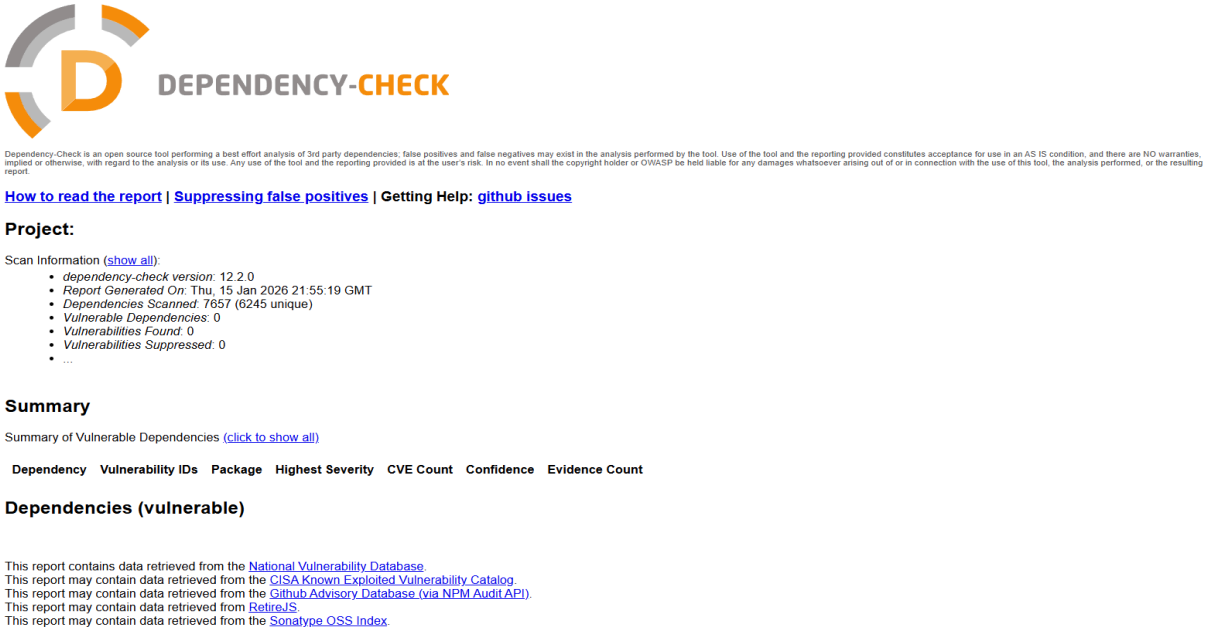
Paralelamente, para mitigar riscos associados a falhas na interface de usuário, o desenvolvimento do *frontend* adotou um ciclo de vida baseado em Integração Contínua (CI). Foram implementadas ferramentas de Análise Estática de Código (*Linting*), visando garantir a padronização e a manutenibilidade do código-fonte por meio de regras estritas de tipagem (TypeScript).

Adicionalmente, a segurança da cadeia de suprimentos de software (*Software Supply Chain Security*) foi auditada utilizando-se as ferramentas OWASP Dependency Check e Red Hat Dependency Analytics, conforme demonstrado na Figura 5. Tais ferramentas analisaram as bibliotecas de terceiros em busca de vulnerabilidades conhecidas (CVEs), garantindo que o sistema de monitoramento não introduzisse riscos de segurança à rede da empresa.

Por fim, a lógica crítica de negócios no cliente - especificamente os algoritmos responsáveis pelo cálculo de datas de vencimento da lubrificação - foi submetida a Testes Unitários Automatizados utilizando a ferramenta Vitest. Essa medida assegura matematicamente que os alertas visuais exibidos no *dashboard* correspondem exatamente às regras de manutenção preditiva definidas.

Além das validações de código, realizou-se a integração sistêmica com os serviços e aplicações. Essa etapa foi dedicada à validação da viabilidade técnica do conceito proposto (ponta a ponta). Os testes consistiram em verificação de latência de comunicação MQTT, integridade na persistência dos dados e testes de desempenho da aplicação sob carga.

**Figura 5 – Relatório de auditoria de segurança gerado automaticamente, atestando a ausência de vulnerabilidades conhecidas nas dependências do projeto.**


DEPENDENCY-CHECK

Dependency-Check is an open source tool performing a best effort analysis of 3rd party dependencies; false positives and false negatives may exist in the analysis performed by the tool. Use of the tool and the reporting provided constitutes acceptance for use in an AS IS condition, and there are NO warranties, implied or otherwise, with regard to the analysis or its use. Any use of the tool and the reporting provided is at the user's risk. In no event shall the copyright holder or OWASP be held liable for any damages whatsoever arising out of or in connection with the use of this tool, the analysis performed, or the resulting report.

How to read the report | Suppressing false positives | Getting Help: github issues

**Project:**

Scan Information (show all):

- *dependency-check version*: 12.2.0
- *Report Generated On*: Thu, 15 Jan 2026 21:55:19 GMT
- *Dependencies Scanned*: 7657 (6245 unique)
- *Vulnerable Dependencies*: 0
- *Vulnerabilities Found*: 0
- *Vulnerabilities Suppressed*: 0
- ...

**Summary**

Summary of Vulnerable Dependencies (click to show all)

| Dependency | Vulnerability IDs | Package | Highest Severity | CVE Count | Confidence | Evidence Count |
|---|---|---|---|---|---|---|

**Dependencies (vulnerable)**

This report contains data retrieved from the National Vulnerability Database.
This report may contain data retrieved from the CISA Known Exploited Vulnerability Catalog.
This report may contain data retrieved from the Github Advisory Database (via NPM Audit API).
This report may contain data retrieved from RetireJS.
This report may contain data retrieved from the Sonatype OSS Index.

## 3.5 COMPONENTES E ARQUITETURA DO SISTEMA NA FASE 2

Esta seção dedica-se a apontar cada componente do Hardware a ser utilizado, suas funcionalidades dentro da aplicação e as possíveis ferramentas e *softwares* a serem adotados. E por fim, o plano de migração da Fase 1 para a Fase 2, onde a Fase 1 se caracterizou por ser a Prova de Conceito (*PoC*) e a Fase 2 tendo como característica a adequação do sistema para o ambiente industrial.

### 3.5.1 Hardware

**Tabela 11 – Componentes de hardware a serem utilizados**

| | |
|---|---|
| ***CLP*** | Equipamento robusto e dedicado ao uso industrial. Esse dispositivo irá desempenhar a tarefa a qual o ESP32 era dedicado na Fase 1. |
| ***Sensor de proximidade indutivo*** | Sua finalidade é medir a distância total de deslocamento da cabeça de impressão e fornecer esses dados ao CLP. |
| ***Hardware para armazenamento de dados e serviços*** | O Hardware para armazenamento de dados e serviços, não está definido. A equipe do Desafio 01 irá alinhar este ponto com a equipe de manutenção da Guararapes. |

### 3.5.2 Softwares e Tecnologias

**Tabela 12 – Softwares e Tecnologias a serem utilizadas**

| | |
|---|---|
| ***MasterTool IEC e linguagem Ladder*** | A IDE adotada para essa fase do projeto é a MasterTool IEC e a linguagem de programação adotada será a linguagem Ladder. O critério adotado foi baseado na recomendação da equipe de manutenção da Guararapes. |
| ***Comunicação*** | A ser definida com a equipe de manutenção. |
| ***API*** | A ser definida com a equipe de manutenção. |

### 3.5.3 Plano de Migração da Fase 1 para Fase 2

**Tabela 13 – Plano de migração da Fase 1 para Fase 2**

| | |
|---|---|
| ***Mapear os sinais de I/Os*** | Listar todos sensores que estavam no ESP32 e equivalentes em módulos do CLP. |
| ***Escolher protocolo de comunicação*** | Analisar a viabilidade dos protocolos de comunicação, podendo ser MQTT nativo ou Modbus. |
| ***Definir topologia de rede*** | Analisar a viabilidade do uso de WI-FI, cabo, switch industriais, VLANs. |
| ***Desenvolver a lógica no CLP*** | Usar a linguagem Ladder ou estruturada juntamente com a IDE para fazer a leitura e tratamento de sinais vindos do sensor. |
| ***Ajustar subscriber/API*** | Aceitar e validar novos formatos, testar ingestão e armazenamento de dados provenientes do sensor. |
| ***Testes em bancada*** | Simular sensores, testar o recebimento e armazenamento dos dados. |
| ***Testes em campo*** | implementar o dispositivo em máquina e verificar o recebimento e armazenamento dos dados. |

| | |
|---|---|
| ***Ajustar limiares*** | Calibrar o dispositivo com dados reais e estressar o dispositivo a fim de verificar o comportamento. |
| ***Homologar com TI/OT*** | Analisar a segurança e backup, com o objetivo de assegurar a integridade dos dados recebidos. |
| ***Documentação*** | Documentar a solução por meio de diagramas, mapeamento do comportamento e rotinas de manutenção. |

## 4 DESENVOLVIMENTO E RESULTADOS OBTIDOS

Inicialmente, o projeto teve como foco o setor de *Silk Screen*, que é o setor de maior receita para a empresa, especificamente a máquina de carrossel de estamparia, onde foram identificados pontos críticos que deveriam ser tratados para que a produção não fosse parada por questões de manutenção. Portanto, mensurar o desgaste das pastilhas e casquilhos localizados na cabeça de impressão, a fim de estabelecer o momento ideal para a substituição desses componentes, foi a direção tomada pela equipe do Desafio 01 em comum acordo com a equipe de manutenção da empresa.

Com o problema identificado, a equipe do desafio se dedicou a compreender o funcionamento da máquina, e suas características, fazendo a modelagem do comportamento da mesma por meio de um autômato e realizando a análise de requisitos funcionais e não funcionais. Concluídas essas atividades, iniciou-se a Fase 1 do projeto.

Nesta etapa, a equipe estudou os possíveis componentes para a prototipação do dispositivo, respeitando os requisitos levantados e apontamentos sugeridos pela equipe de manutenção. Este processo exigiu da equipe bastante atenção, dada a variedade de sensores que poderiam nos gerar alguma métrica, contudo essas métricas poderiam não ser eficientes para assegurar com fidelidade o desgaste dos componentes.

A implementação da Fase 1 do projeto permitiu validar, em ambiente controlado, a arquitetura proposta para o sistema de monitoramento e manutenção preditiva das pastilhas e casquilhos localizados na longarina do Carrossel.

Os resultados demonstram que o fluxo de aquisição, transmissão, armazenamento e visualização dos dados operou de forma satisfatória, estável e coerente com os requisitos previamente definidos antes do início da Fase 1.

Dessa forma, com todo arcabouço adquirido ao fim da Fase 1, a equipe do Desafio 01 pôde partir para a segunda fase do projeto. Esta fase se propõe a realizar a substituição do ESP32 e do sensor, migrando para o CLP juntamente com os sensores industriais.

## 4.1 DESENVOLVIMENTO DA INTERFACE E VALIDAÇÃO DE INTEGRAÇÃO

Para garantir que os dados coletados fossem apresentados de forma clara e acionável para a equipe de manutenção, o desenvolvimento da interface seguiu duas etapas: prototipação de alta fidelidade e implementação da aplicação reativa.

Inicialmente, foi desenvolvido um protótipo utilizando a ferramenta Figma. O design foi focado na experiência do usuário (UX/UI), criando um Dashboard com duas visões principais: uma "Visão Geral", que permite monitorar o status de múltiplos carrosséis simultaneamente através de indicadores visuais (semáforo); e uma visão de "Detalhes", onde é possível acompanhar o desgaste específico das peças e o histórico de lubrificação de cada cabeça de impressão.

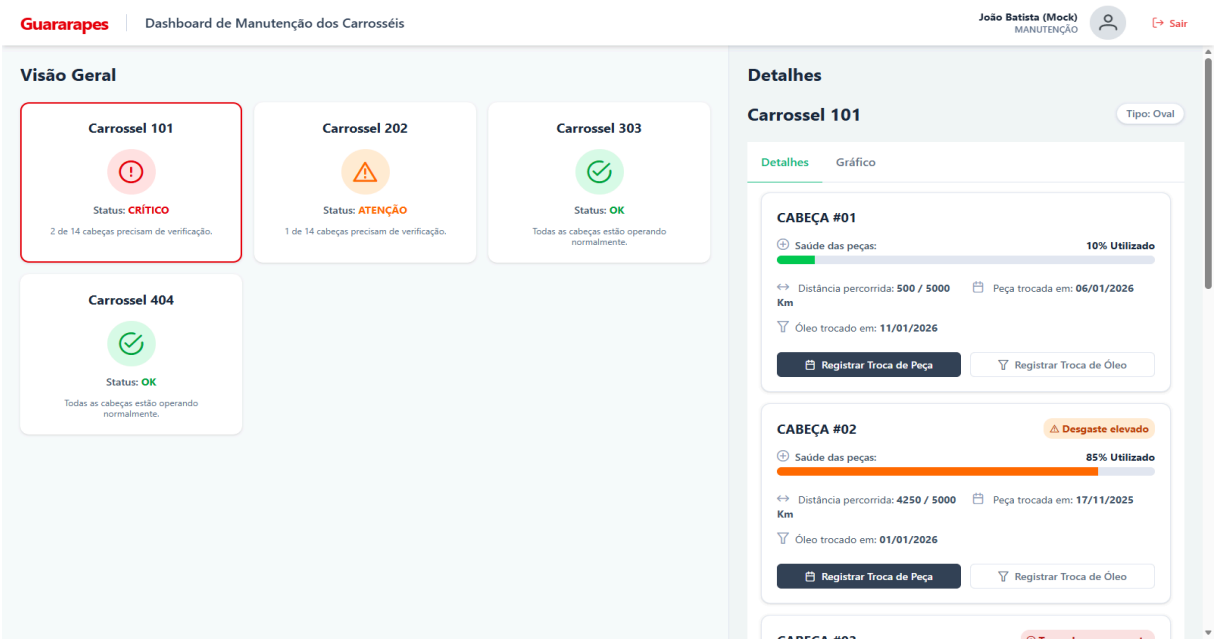
**Figura 6 – Protótipo de Alta Fidelidade do Dashboard de Manutenção**

Com o design validado, partiu-se para a implementação final utilizando o framework Vue.js. Diferente de uma página web estática, optou-se por uma arquitetura de Aplicação de Página Única (SPA).

Essa abordagem permitiu incorporar regras de negócio complexas diretamente no navegador, como o cálculo automático da saúde das peças (baseado na telemetria recebida) e a verificação temporal da validade do óleo (alertas automáticos após 30 dias).

A interface final também conta com responsividade avançada (Drawer lateral) para acesso via dispositivos móveis no chão de fábrica.

**Figura 7 – Interface Final do Dashboard implementada em Vue.js, apresentando a visualização dos dados de telemetria e o painel de detalhes**




## 5 DISCUSSÃO

Embora a Fase 1 tenha sido realizada poucas vezes em ambiente industrial, os resultados indicam que a arquitetura é funcional, escalável e suficiente para a fase de integração com sensores industriais e CLP.

Esta etapa inicial demonstrou que o protótipo do sistema é tecnicamente viável e atende aos requisitos iniciais do projeto. O fluxo completo da coleta, transmissão, processamento e visualização dos dados funcionou de forma estável, com boa precisão de

leitura e comunicação confiável via MQTT. Os resultados foram satisfatórios, especialmente na integração entre ESP32, *broker*, API e dashboard, que operaram de forma esperada.

Dessa forma, o protótipo permite passar da Fase 1 para a fase seguinte como planejado. No entanto, é necessário discutir as limitações e ameaças à validade como boa prática para o início da segunda fase do projeto.

## 5.1. LIMITAÇÕES IDENTIFICADAS NA FASE 1

- Robustez ambiental: O uso de componentes de prototipagem (ESP32, sensor VL53L0X) não garante resistência a condições industriais severas, como vibrações contínuas, temperaturas elevadas e interferência eletromagnética.
- Correlação indireta do desgaste: A métrica de "distância percorrida" é um *proxy* para o desgaste real dos componentes (pastilhas e casquilhos). A validação dessa correlação exige dados de campo e calibração com eventos reais de manutenção.
- Segurança em rede industrial: A implementação atual utiliza autenticação básica em MQTT. Em ambiente fabril, são necessárias camadas adicionais de segurança, como criptografia TLS/SSL e segmentação de rede via VLANs.

## 5.2. AMEAÇAS A VALIDADE

- Validade interna: A conclusão de que o sistema prevê desgaste baseia-se em uma premissa não completamente validada em campo. A relação causal entre distância acumulada e falha requer confirmação com dados reais.
- Validade externa: O protótipo foi testado em simulação de uma única cabeça de impressão. A generalização para múltiplas máquinas ou diferentes perfis de operação requer testes adicionais.

## 5.3 TRANSIÇÃO PARA FASE 2

Os resultados satisfatórios da Fase 1 não apenas validam a arquitetura, mas também destacam a importância do plano de migração. A substituição do ESP32 por um CLP, do sensor óptico por um sensor de proximidade indutivo e a adoção de protocolos industriais como Modbus serão essenciais para:

- Garantir robustez operacional em ambiente real;
- Coletar dados para calibrar os limiares de alerta com base em estatística e eventos reais;
- Permitir a escalabilidade do sistema para múltiplas cabeças e máquinas.

Portanto, a discussão reforça que o êxito da Fase 1 reside não apenas na funcionalidade demonstrada, mas na clareza e no embasamento do caminho traçado para a industrialização, posicionando este trabalho como uma contribuição prática e replicável para a manutenção preditiva na Indústria 4.0.

## 6 CONCLUSÃO

O principal desafio encontrado na Fase 1 foi encontrar o sensor para fazer a captura dos dados. Há vários tipos de sensores que de alguma forma pudessem fornecer métricas para averiguar o desgaste, porém, o principal critério estabelecido pela equipe de manutenção da empresa foi que o sistema deveria ser não invasivo, ou seja, o sistema deveria ser capaz de não afetar o funcionamento da máquina mesmo estando acoplado à mesma.

Ao final da Fase 1 pudemos concluir que obtivemos êxito na construção do protótipo, pois conseguimos fazer a coleta dos dados com os componentes próprios para prototipação, de modo que sensores, microcontroladores, serviços e aplicações integraram de forma esperada e sem afetar o funcionamento da máquina.

Desta forma, a segunda fase se propõe a realizar a substituição do ESP32 e do sensor, migrando para o CLP juntamente com os sensores industriais. Em seguida, iniciaremos testes em campo com dois ou mais sensores e coletar dados reais por três semanas, rotulando eventos de manutenção corretiva para validação. Com a base coletada, refinamos os limiares de alerta usando análise estatística e modelos de tendência, reduzindo falsos positivos.

Por fim, será realizada a homologação, cobrindo segurança de rede, políticas de backup e critérios de aceitação operacional. No final da Fase 2, se espera que o sistema seja escalável ao ponto que várias cabeças da mesma máquina possam ser monitoradas, tornando a solução replicável para outras máquinas de carrossel.

## REFERÊNCIAS


ALBERTO, João Vitor Rodrigues. **Sensores e sua importância**: descrição e uso na indústria 4.0. [S. l.: s. n.], 2023.

ALMEIDA, R. M. A.; MORAES, C. H. V.; SERAPHIM, T. F. P. **Programação de Sistemas Embarcados**: desenvolvendo software para microcontroladores em linguagem C. 1. ed. Rio de Janeiro: LTC, 2021.

ALTUS. **XP325**. 2025. Disponível em: https://www.altus.com.br/produto/xp325. Acesso em: 7 dez. 2025.

ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS. **ABNT NBR 5462**: confiabilidade e mantenabilidade. Rio de Janeiro: ABNT, 1994.

BALDISSARELLI, Luis. **Manutenção Preditiva na indústria 4.0**. Scientia cum industria, [S. l.], 2019.

BRASIL. Lei nº 9.610, de 19 de fevereiro de 1998. Altera, atualiza e consolida a legislação sobre direitos autorais e dá outras providências. **Diário Oficial da União**: seção 1, Brasília, DF, 1998. Disponível em: http://www.planalto.gov.br/ccivil_03/leis/19610.htm. Acesso em: 7 dez. 2025.

CANINOS LOUCOS. **Labrador 64 bits**. 2025. Disponível em: https://caninosloucos.org/pt/labrador-64-pt/. Acesso em: 6 dez. 2025.

CHASE, Otavio. **Sistemas Embarcados**. [S. l.: s. n., 2025?]. Disponível em: http://www.lyfreitas.com.br/ant/pdf/Embarcados.pdf. Acesso em: 5 nov. 2025.

CHWIF, L.; MEDINA, A. C. **Modelagem e simulação de eventos discretos**: teoria e aplicações. 3. ed. São Paulo: Ed. do Autor, 2010.

COSTA, Isabele Morais; LISBOA, Stella Neves Duarte; SANTOS, Talita Pitanga. **Automação industrial**. Natal: UFRN, 2003.

ESPRESSIF. **ESP32**. 2025. Disponível em: https://www.espressif.com/en/products/socs/esp32. Acesso em: 7 dez. 2025.

LIMA FILHO, Marcos Camargo et al. **Conceitos fundamentais sobre incidência solar e estudo de seguidores utilizando dispositivos com prototipagem livre**. [S. l.: s. n., 2025?].

MARIADB. **MariaDB Documentation**. 2025. Disponível em: https://mariadb.org/documentation/. Acesso em: 6 dez. 2025.

MOSQUITTO. **Um broker MQTT de código aberto**. 2025. Disponível em: https://mosquitto.org/. Acesso em: 6 dez. 2025.

MQTT. **O Padrão para mensagens MQTT**. 2025. Disponível em: https://mqtt.org/mqtt-specification/. Acesso em: 6 dez. 2025.

OLIVEIRA, S. de. **Internet das coisas com ESP8266, Arduino e Raspberry PI**. São Paulo: Novatec, 2017.

RODRIGUES, Gabriel Santos. **Internet das Coisas (IoT) aplicado no controle online de variação de velocidade de um motor de indução**. [S. l.: s. n.], 2024.

SANTOS, Bruno P. et al. **Internet das coisas**: da teoria à prática. In: SIMPÓSIO BRASILEIRO DE REDES DE COMPUTADORES E SISTEMAS DISTRIBUÍDOS, 31., 2016. **Minicursos...** [S. l.]: SBRC, 2016. p. 16.

SICK. **Sensor IME12-04BPSZC0S**. 2025. Disponível em: https://www.alldatasheet.com/datasheet-pdf/pdf/1008421/SICK/IME12-04BPSZC0K.html. Acesso em: 7 dez. 2025.

SIQUEIRA, Iony Patriota de. **Manutenção centrada na confiabilidade**. 3. ed. Rio de Janeiro: Qualitymark, 2014.

SOUZA, Valdir Cardoso de et al. Utilização das tecnologias da indústria 4.0 na manutenção preditiva através do monitoramento de equipamentos e instalações. **Brazilian Journal of Development**, v. 8, n. 1, p. 7063-7083, 2022.

STMicroelectronics. **Datasheet VL53L0X**. 2025. Disponível em: www.alldatasheet.com/datasheet-pdf/pdf/948120/STMICROELECTRONICS/VL53L0X.html. Acesso em: 7 dez. 2025.

WOLF, M. **Computers as Components**: principles of embedded computing system design. 4. ed. Cambridge: Elsevier, 2017.