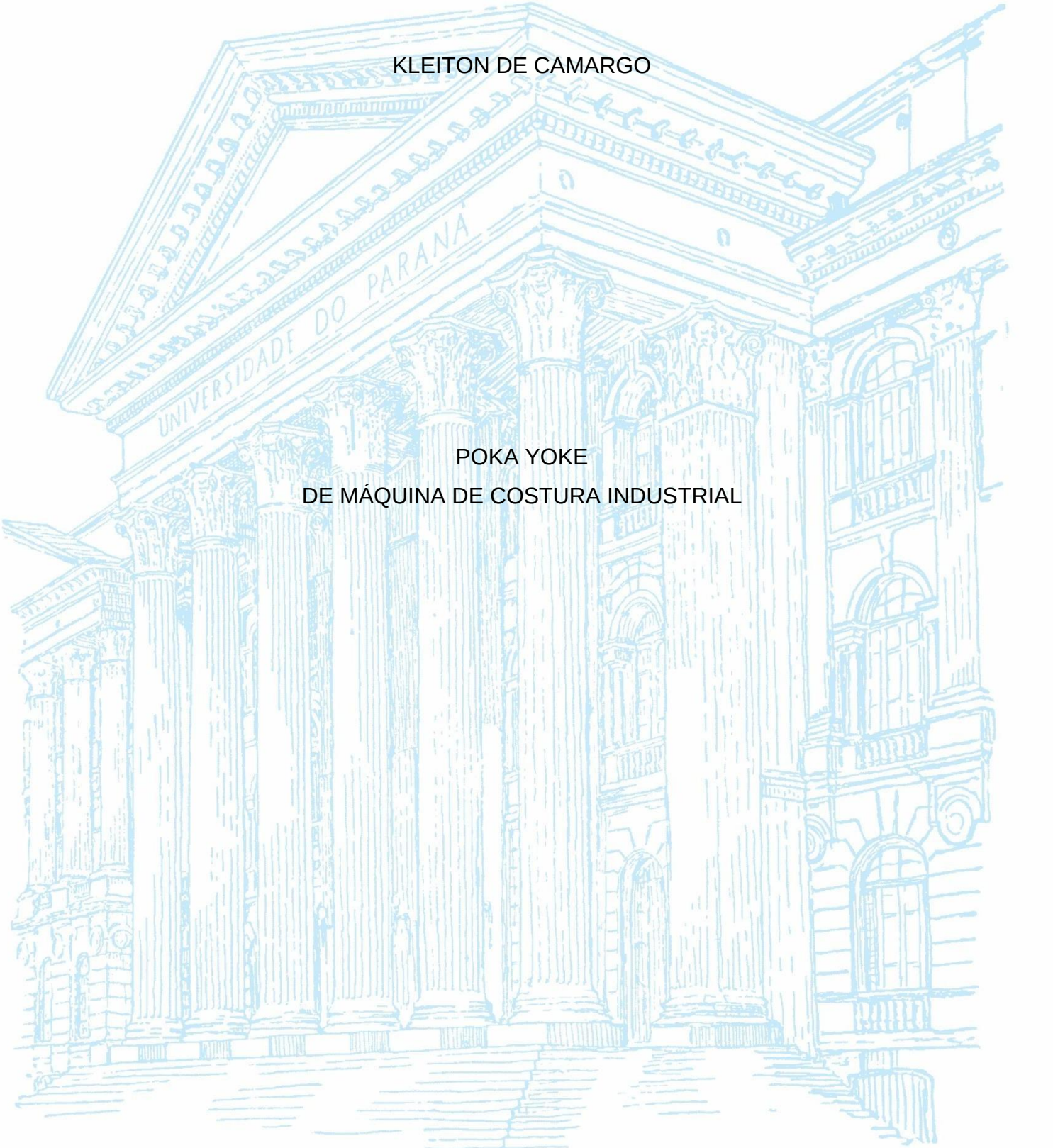
UNIVERSIDADE FEDERAL DO PARANÁ

KLEITON DE CAMARGO



POKA YOKE
DE MÁQUINA DE COSTURA INDUSTRIAL

CURITIBA

2025

KLEITON DE CAMARGO

POKA YOKE
DE MÁQUINA DE COSTURA INDUSTRIAL

Trabalho de Conclusão de Curso apresentado como requisito parcial à obtenção do título de Bacharel, Curso de Graduação em Engenharia Elétrica com ênfase em Sistemas Eletrônicos Embarcados, Setor de Tecnologia, Universidade Federal do Paraná.

Orientador: Prof. Dr. Marcos Vinicio Haas Rambo

CURITIBA
2025

TERMO DE APROVAÇÃO

KLEITON DE CAMARGO

POKA YOKE
DE MÁQUINA DE COSTURA INDUSTRIAL

Trabalho de Conclusão de Curso apresentado ao curso de Graduação em Engenharia Elétrica, Setor de Tecnologia, Universidade Federal do Paraná, como requisito parcial à obtenção do título de Bacharel em Engenharia Elétrica com ênfase em Sistemas Eletrônicos Embarcados.

_____

Prof. Dr. Marcos Vinicio Haas Rambo – Orientador Departamento de Engenharia Elétrica, UFPR

_____

Prof. Dra. Luiza Beana Chipansky Freire
Departamento de Engenharia Elétrica, UFPR

_____

Prof. Dra. Ana Flávia dos Reis
Departamento de Engenharia Elétrica, UFPR

Curitiba, 12 de julho de 2025.

DEDICATÓRIA

Dedico este trabalho à minha mãe e esposa que sempre me apoiaram e não me deixaram desistir, sendo minha mãe um exemplo de força por vencer as grandes enfermidades caídas sobre ela, e minha esposa um exemplo de mãe, carinhosa e guerreira que me acompanha nessa grande viagem que se chama vida.

AGRADECIMENTOS

À minha mãe, Celi, que sempre me apoiou e insistiu para que eu concluísse essa parte da minha vida, sendo o principal motivo pelo qual estou terminando essa jornada.

À minha esposa, Beatriz, que dedica sua vida ao meu lado e é responsável por me motivar a sempre melhorar como pessoa e profissional, não me deixando desistir nos momentos de fraqueza.

A todos os colegas e amigos que fiz durante o curso, que certamente fizeram total diferença para a conclusão das matérias, seja pela ajuda com conteúdo, explicações, ou mesmo pelo incentivo e apoio.

Ao professor Marcos Vinicio Haas Rambo, que dedicou seu tempo e empenho para que eu pudesse finalizar com excelência esse ciclo da minha vida.

"Errei mais de 9.000 arremessos na minha carreira. Perdi quase 300 jogos. Em 26 ocasiões, confiaram em mim para o arremesso da vitória e eu errei. Eu falhei inúmeras vezes na minha vida. E é por isso que eu tenho sucesso, nunca desista de seus objetivos, mesmo que pareçam inatingíveis. Cada falha é uma oportunidade de aprender e chegar mais perto do triunfo."

– Michael Jordan

RESUMO

Na indústria de manufatura, especialmente nos setores têxtil, automotivo e aeroespacial, os processos de costura são essenciais para garantir a qualidade e a segurança dos produtos finais. Itens como cintos de segurança, airbags e estofados exigem costuras padronizadas, precisas e rastreáveis. Nessas aplicações, qualquer falha pode comprometer a integridade do produto e gerar sérias consequências.

Diante da crescente exigência por controle de qualidade e da necessidade de reduzir falhas humanas, soluções automatizadas vêm ganhando espaço. Entre elas, destaca-se o uso do conceito poka-yoke — termo japonês que significa “à prova de erros” — que visa prevenir equívocos operacionais por meio de sistemas que orientam ou restringem a execução incorreta de tarefas.

Este trabalho apresenta o desenvolvimento de um sistema que orienta o operador passo a passo durante o processo de costura, assegurando a conformidade com os padrões estabelecidos e a rastreabilidade de cada operação realizada. O sistema também realiza validações automáticas por meio de sensores e inspeção visual, além de controlar e registrar a identificação do operador.

A arquitetura do sistema foi projetada para ser modular, organizada e de fácil manutenção, adotando boas práticas de desenvolvimento que separam as camadas de interface, controle e dados.

Palavras-chave**:** Costura; Automação; Poka-yoke; Controle de Qualidade; Rastreamento; Inspeção Visual; Ergonomia Operacional.

ABSTRACT

In the manufacturing industry, especially in the textile, automotive, and aerospace sectors, sewing processes are essential for ensuring the quality and safety of final products. Items such as seat belts, airbags, and upholstery require stitching that is precise, standardized, and traceable. In such applications, any failure can compromise product integrity and lead to serious consequences.

With increasing demands for quality control and the need to reduce human error, automated solutions have become more prominent. One such approach is based on the concept of poka-yoke—a Japanese term meaning “mistake-proof”—which seeks to prevent operational errors through systems that guide or constrain incorrect task execution.

This work presents the development of a system that guides the operator step by step during the sewing process, ensuring compliance with established standards and traceability of each operation. The system also performs automatic validations through sensors and visual inspection, while controlling and recording user identification.

Its architecture was designed to be modular, organized, and easy to maintain, following best practices that separate the interface, logic, and data layers.

Keywords: Sewing; Automation; Poka-yoke; Quality Control; Traceability; Visual Inspection; Operator Guidance.

FIGURAS

DIAGRAMAS

TABELAS

SIGLAS

API	Application Program Interface
CA	Corrente Alternada
CC	Corrente Contínua
CSS	Cascading Style Sheets
DBMS	Database Management System
DER	Diagrama Entidade-Relacionamento
DOM	Document Object Model
HTML	HyperText Markup Language
MDF	Fibras de Média Densidade
MRC	Medical Research Council
PPP	Point-to-Point Protocol
SPI	Interface Periférica Serial
SQL	Structured Query Language
SSP	Síndrome Pós-Pólio
UI	User Interface
USB	Universal Serial Bus
SO	Sistema operacional
EFKA	Empresa responsável pelo controlador
BIO	Control ID, empresa responsável pelo leitor biométrico
WPF	Windows Presentation Foundation
MVC	Model-View-Controller
HTML	HyperText Markup Language
SQL	Structured Query Language
IoT	Internet of Things
USB	Universal Serial Bus
API	Application Program Interface
CA	Corrente Alternada
CC	Corrente Contínua
CSS	Cascading Style Sheets
DBMS	Database Management System

DER	Diagrama Entidade-Relacionamento
DOM	Document Object Model
SPI	Interface Periférica Serial
SSP	Síndrome Pós-Pólio
UI	User Interface

SUMÁRIO

1 INTRODUÇÃO

Na manufatura têxtil, especialmente em processos de costura especializados com requisitos de segurança, falhas operacionais e humanas são causas recorrentes de desperdícios, retrabalhos e comprometimento da qualidade final dos produtos. Pequenos erros na sequência de costura podem resultar em peças descartadas, prejuízos econômicos e, em casos mais críticos, risco à integridade de componentes cuja segurança estrutural depende da precisão do processo costural, como ocorre em setores automotivo, aeroespacial e médico-hospitalar.

A ausência de controle rigoroso sobre a execução de cada etapa da costura compromete a rastreabilidade e dificulta a padronização da produção. Em ambientes industriais com alta demanda, a repetitividade da atividade e a dependência de ações humanas sem assistência técnica adequada tornam o sistema suscetível a falhas que poderiam ser evitadas com mecanismos de prevenção de erros.

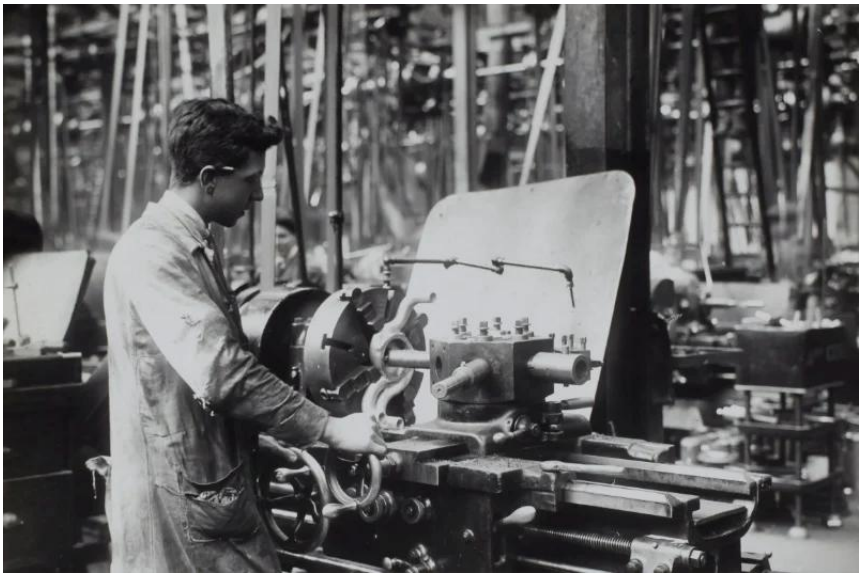
A necessidade de evitar tais falhas é antiga. Desde a Antiguidade, o ser humano busca formas de automatizar tarefas para minimizar erros. Ctesíbio de Alexandria, por volta de 270 a.C., projetou um regulador de fluxo para relógios de água, um dos primeiros sistemas de controle automático por realimentação (Bedini, 1963). No século IX, os irmãos Banu Musa descreveram sistemas de controle automático em dispositivos hidráulicos (Hill, 1991).

Contudo, foi somente com a Revolução Industrial, no século XVIII, que a automação passou a transformar significativamente os processos produtivos. O uso de máquinas para substituir a mão de obra manual introduziu um novo paradigma de eficiência e controle. Um marco relevante nesse contexto foi a invenção da máquina de costura por Thomas Saint em 1790, posteriormente aprimorada por inventores como Barthélemy Thimonnier em 1829, que viabilizou a produção mecanizada de roupas e tecidos (Cooper, 1976).

Apesar dos avanços, até os dias atuais, muitas máquinas de costura industriais operam de forma independente de sistemas inteligentes de controle, o que permite que etapas importantes do processo sejam negligenciadas ou executadas de forma incorreta. Isso reforça a necessidade de métodos que orientem o operador na execução correta das costuras e que possibilitem a verificação da conformidade de cada peça, prevenindo falhas e aumentando a confiabilidade do processo.

A figura 1 mostra uma cena típica de uma fábrica durante a Revolução Industrial ou o início do século XX**.** Nela, um operário trabalha em uma máquina-ferramenta (provavelmente um torno mecânico), utilizada para usinagem de peças metálicas. O ambiente é repleto de correias e polias no teto, que fazem parte de um sistema de transmissão de energia por eixos e correias, comum antes da eletrificação individual das máquinas.

Figura 1 – Operação de máquina.



FONTE: PROGRESSIVE AUTOMATION (2024)

1.1 OBJETIVO

1.1.1 Objetivo Geral

Desenvolver um sistema visual de poka-yoke para máquinas de costura industriais, com foco na prevenção de falhas operacionais, garantia da conformidade do processo e aumento da eficiência produtiva. O sistema será composto por uma aplicação em C# baseada na arquitetura MVC, com interface desenvolvida em WPF, integrada a sensores indutivos, saídas digitais e câmera de visão Cognex. A plataforma será executada em um PC industrial com Windows 10 Pro, comunicando-se com os dispositivos por meio de relés de interface e protocolos industriais como Ethernet/IP ou Modbus TCP. Todas as informações do processo — como registros de operação, falhas, imagens e dados de usuários — serão armazenadas em um banco de dados SQL Server, com controle de acesso biométrico para garantir a rastreabilidade e segurança das operações.

Dessa forma, o objetivo principal é desenvolver um sistema integrado de monitoramento e controle da máquina de costura, utilizando tecnologias de automação e visão computacional para aumentar a precisão, a eficiência e a confiabilidade do processo.

1.1.2 Objetivos específicos

Este trabalho tem como objetivos:

- Identificar e especificar os componentes de hardware necessários para o sistema, incluindo sensores indutivos, saídas digitais, relés, máquina de costura e câmera Cognex da linha In-Sight SnAPP.
- Desenvolver o circuito de controle responsável pela contagem precisa dos pontos de costura e pelo acionamento das saídas digitais para liberação da costura e corte da linha.
- Implementar o backend em C# para controle da lógica de negócios e integração com os dispositivos físicos, além de gerenciar os dados coletados.

- Construir a interface gráfica do sistema em WPF, utilizando recursos como data binding, templates e triggers para fornecer visualização em tempo real e controle intuitivo das operações.
- Integrar o sistema de visão Cognex para realizar inspeção visual automática das costuras, validando a conformidade dos produtos com base em imagens processadas em tempo real.
- Implementar banco de dados em Microsoft SQL Server para armazenar informações de operação, logs de erro, imagens capturadas e dados dos operadores com consistência e integridade.
- Desenvolver e integrar um sistema de autenticação biométrica com suporte ao Windows Biometric Framework (WBF), garantindo acesso seguro ao sistema e rastreabilidade individual de cada operação.
- Testar e validar o sistema completo em ambiente real de produção, assegurando sua precisão, desempenho, confiabilidade e integração eficiente entre hardware e software.

2 REVISÃO BIBLIOGRÁFICA

2.1 POKA-YOKE

O termo poka-yoke tem origem no idioma japonês e significa “à prova de erros” ou “prevenção de erros”. Foi cunhado por Shigeo Shingo, engenheiro industrial da Toyota, na década de 1960, dentro do contexto do Sistema Toyota de Produção. A expressão original era baka-yoke (“à prova de idiotas”), mas foi modificada para poka-yoke por respeito aos operadores da linha de produção (WIKIPÉDIA, 2025).

A proposta central do poka-yoke é simples, mas poderosa: criar mecanismos físicos, visuais ou lógicos que impeçam a ocorrência de falhas humanas ou que as sinalizem imediatamente, antes que causem defeitos no produto final. Shingo exemplificava com casos como a ausência de molas em interruptores, resolvida por meio de gabaritos que exigiam a inserção das peças para liberar o próximo passo do processo (SHINGO, 1987).

Esses dispositivos são geralmente classificados em três categorias principais: por contato (verificação de forma, tamanho ou posição), por valor fixo (controle de quantidade) e por sequência (ordem correta das etapas). Além disso, podem atuar de forma preventiva (impedindo a continuidade do processo até que o erro seja resolvido) ou alertando (avisando sobre a falha, mas permitindo o prosseguimento após intervenção do operador) (GOSKILLS, 2025; BLOGS.MTU.EDU, 2025).

Com o avanço das práticas Lean Manufacturing e Six Sigma, o poka-yoke passou a ocupar posição estratégica nas rotinas de melhoria contínua, sendo amplamente utilizado para aumentar a qualidade, reduzir desperdícios, agilizar treinamentos e elevar o nível de confiabilidade dos processos produtivos. Sua implementação, quando bem planejada, representa um investimento com retorno elevado e impacto direto na cultura organizacional voltada à excelência operacional (6SIGMA.US, 2025; TULIP.CO, 2025).

Diversos exemplos do dia a dia ilustram o conceito de forma prática: fornos de micro-ondas que não funcionam com a porta aberta, transmissões automáticas que exigem pedal no freio para serem acionadas, tomadas com travas de segurança infantil ou conectores USB que só encaixam de um lado são todos exemplos de poka-yoke aplicados para evitar erros por descuido ou falta de atenção (UNLEASHEDSOFTWARE.COM, 2025; BEEWATEC.COM, 2025).

Entretanto, como qualquer ferramenta, sua adoção deve ser ponderada. Projetos mal dimensionados podem gerar sistemas rígidos demais ou de difícil manutenção. Além disso, pode haver um custo inicial de implementação, especialmente em processos que exigem sensores, atuadores ou lógica de controle adicional (FRACTORY.COM, 2025).

Ainda assim, o poka-yoke se mantém como uma das estratégias mais eficazes para garantir a qualidade desde a origem, atuando na prevenção e não apenas na detecção de defeitos. Sua simplicidade conceitual contrasta com seu impacto profundo nas operações industriais, sendo uma das bases para alcançar a meta de “zero defeitos” nas organizações.

2.2 MÁQUINA DE COSTURA

A definição genérica de máquina de costura segundo Grace Rogers Cooper em seu livro “The Sewing Machine: A Technical History” é um dispositivo mecânico ou eletromecânico projetado para unir tecidos usando um fio de costura. Esse equipamento automatiza o processo de costura, tornando-o mais rápido e consistente em comparação com a costura manual, e pode ser usado para diferentes materiais, como tecidos, couro e outros materiais sintéticos.

Existem basicamente dois tipos de máquinas de costura disponíveis: industriais e domésticas.

As máquinas de costura industriais são maiores, mais rápidas e mais variadas em tamanho, custo, aparência e tarefa. Uma máquina de costura industrial pode lidar com trabalhos de costura pesados. As máquinas industriais, ao contrário das máquinas domésticas, executam uma única tarefa dedicada e são capazes de uso contínuo por longos períodos; eles têm peças móveis maiores e motores maiores classificados para operação contínua. Peças para diferentes máquinas industriais, como motores, pés de costura e bobinas, podem ser intercambiáveis, mas nem sempre é assim.

Os motores das máquinas industriais, como a maioria de seus componentes, luzes, entre outros, são separados, geralmente montados na parte inferior da mesa. As máquinas domésticas têm seus motores OEM montados dentro da máquina.

Existem dois tipos diferentes de motor disponíveis para máquinas industriais: um servo motor (que consome menos eletricidade e é silencioso quando não está em

uso) e o motor de embreagem mais tradicional (que está sempre girando, mesmo quando não está em uso).

Um motor de embreagem está sempre funcionando e fazendo barulho quando está conectado à eletricidade. A operação constante garante consistência e velocidade.

O servo motor usa menos eletricidade do que um motor de embreagem. Ele não emite nenhum som a menos que o operador pise no pedal da máquina, mas não pode suportar o mesmo tipo de uso que um motor de embreagem.

2.3 APLICATIVO WINDOWS

2.3.1 Linguagem HTML

De acordo com Mozilla (2022), HTML (Linguagem de Marcação de Hipertexto, do inglês HyperText Markup Language) é uma linguagem usada para definir a estrutura de uma página web, atribuindo também significado ao seu conteúdo, sendo o bloco fundamental da internet.

A Mozilla (2022) também define Hipertexto como links que conectam diferentes páginas da web, enquanto Marcação ou tags refere-se à forma como o navegador entende o que deve ser exibido e como. Um elemento é o conjunto formado por uma tag, seus atributos, valores e filhos.

As tags são componentes essenciais do HTML, pois com elas a estrutura da página é construída. Sua sintaxe é composta pelo nome da tag desejada, envolto por parênteses angulares (<>). Além disso, toda tag que for aberta deve ser fechada, seguindo a mesma estrutura, mas com uma barra antes do nome da tag.

2.3.2 Linguagem XAML

De acordo com a Microsoft (2023), XAML é uma linguagem de marcação declarativa que, no contexto do modelo de programação .NET, facilita a criação de interfaces de usuário para aplicativos .NET. Com o XAML, você pode definir elementos visuais da interface diretamente na marcação, permitindo a separação entre a interface do usuário e a lógica de execução por meio de arquivos code-behind, que são conectados à marcação usando definições de classes parciais.

Ao contrário de muitas outras linguagens de marcação, que são normalmente interpretadas sem um vínculo direto com um sistema de tipos, o XAML permite a instanciação direta de objetos com base em um conjunto específico de tipos definidos em assemblies. Isso possibilita um fluxo de trabalho colaborativo, onde diferentes equipes podem trabalhar simultaneamente na interface do usuário e na lógica do aplicativo, utilizando ferramentas distintas, otimizando o processo de desenvolvimento.

2.3.3 HTML vs XAML: Comparação de Linguagens de Marcação

De acordo com o artigo da Microsoft (2023) HTML (HyperText Markup Language) e XAML (Extensible Application Markup Language) são linguagens de marcação utilizadas para definir a estrutura e a interface de diferentes tipos de aplicações. Embora compartilhem certas características, como a estrutura em árvore de tags, suas finalidades e ambientes de uso são distintos.

- Finalidade e Aplicação

O HTML é utilizado principalmente para estruturar o conteúdo de páginas web. Ele organiza textos, imagens, links e outros elementos visuais que são exibidos nos navegadores. O HTML serve como a base da web, trabalhando em conjunto com CSS (para estilização) e JavaScript (para interatividade) na criação de páginas dinâmicas e interativas (W3C, HTML Living Standard).

Por outro lado, o XAML é usado para criar interfaces de usuário em aplicações desktop, principalmente no Windows Presentation Foundation (WPF) e em outras tecnologias da Microsoft, como o Xamarin. O XAML facilita a separação entre a interface gráfica e o código lógico da aplicação, permitindo a definição de componentes como botões, caixas de texto, menus, entre outros, além de vincular eventos e lógica C# (Microsoft, XAML Overview, WPF).

- Ambiente de Execução:

O HTML é interpretado por navegadores web, como Chrome, Firefox e Edge, e depende de servidores web para distribuição. Ele é usado em conjunto com protocolos de rede, como HTTP e HTTPS, para criar aplicações web acessíveis em diferentes plataformas.

Já o XAML é utilizado para criar aplicações desktop, que rodam localmente no sistema operacional Windows, através de frameworks como WPF ou UWP. Ele é ideal para criar interfaces gráficas complexas e nativas, com suporte à renderização gráfica acelerada por hardware e acesso direto aos componentes do sistema.

- Interatividade e Lógica

O HTML, por si só, não lida diretamente com lógica e interatividade; para isso, é necessário o uso de linguagens como JavaScript ou frameworks como React.js e Angular. O HTML define apenas a estrutura básica, enquanto JavaScript gerencia eventos como cliques, animações e outras interações.

No XAML, há uma integração direta com C# e outros frameworks .NET. Eventos e lógica de interação, como cliques em botões ou entradas de texto, são associados diretamente aos elementos da interface dentro do próprio XAML, permitindo uma interação mais rica e nativa com os controles e eventos do sistema operacional. Por exemplo, no WPF, é possível associar um botão diretamente a um evento C# no XAML, algo que em HTML exigiria JavaScript.

- Estilos e Customização:

O HTML utiliza o CSS (*Cascading Style Sheets*) para definir a aparência visual de uma página, controlando cores, tamanhos, espaçamentos e animações.

O XAML possui um sistema similar de estilização, com *Styles e Templates*, que permite a personalização da aparência dos controles e a criação de temas para a aplicação. A vantagem do XAML é sua integração profunda entre lógica e interface, suportando recursos como *Data Binding* e animações mais complexas (Microsoft, *Styles and Templates in XAML*).

- Evolução e Suporte:

O HTML é amplamente suportado por todas as plataformas e continua a evoluir como parte dos padrões da web, incorporando novas tecnologias como *WebAssembly* e *Progressive* Web Apps (PWAs).

O XAML, embora mais restrito ao ecossistema da Microsoft, é uma ferramenta poderosa para o desenvolvimento de aplicações de desktop e mobile nas plataformas WPF, UWP e *Xamarin.Forms*.

A principal diferença entre HTML e XAML está no contexto de uso e na funcionalidade que cada uma oferece. O HTML é projetado para a web, sendo utilizado para estruturar e exibir páginas web interativas. Já o XAML é voltado para aplicações desktop e mobile, permitindo a criação de interfaces gráficas ricas em aplicações baseadas no Windows.

2.3.4 WPF

De acordo com o artigo da Microsoft (2023) *Windows Presentation Foundation* (WPF) é uma estrutura de interface do usuário que é independente de resolução e usa um mecanismo de renderização baseado em vetor, criado para aproveitar o hardware gráfico moderno.

A linguagem de marcação por trás da interface do WPF não é o HTML, mas sim o XAML (*Extensible Application Markup Language*). Embora ambos sejam linguagens de marcação, o XAML é utilizado especificamente para interfaces gráficas no WPF, enquanto o HTML é usado para estruturar páginas web. Diferente de uma interface web convencional, no WPF é possível associar eventos a cada componente da interface. Por exemplo, ao inserir um botão, é possível vincular diversos tipos de eventos a ele, como quando o mouse passa sobre o botão, quando é clicado, ou quando é clicado e segurado, entre outros. Há uma ampla variedade de eventos disponíveis, o que facilita a programação de interações. Programar com WPF se assemelha, de certa forma, à programação de aplicativos no Android Studio, especialmente no que diz respeito à manipulação de interfaces gráficas e eventos.

2.3.5 Linguagem de programação C#

De acordo com a Microsoft (2000), C# (pronunciado "C sharp") é uma linguagem de programação moderna e versátil, desenvolvida e lançada em 2000 como parte da plataforma .NET. Criada por Anders Hejlsberg e sua equipe, a linguagem rapidamente se destacou como uma das mais populares para o desenvolvimento de aplicativos Windows, aplicativos web e soluções móveis. Com seu crescimento contínuo, C# se tornou uma escolha robusta e confiável em diversos cenários, de pequenas aplicações a grandes sistemas empresariais.

Entre suas características mais marcantes de acordo com a Microsoft (2003), C# se destaca por:

- Orientação a Objetos: A linguagem é totalmente orientada a objetos, o que significa que tudo em C# gira em torno de objetos e classes, tornando o código modular, reutilizável e mais fácil de manter.
- Gerenciamento Automático de Memória: Com um coletor de lixo integrado, C# lida automaticamente com a alocação e liberação de memória, reduzindo a carga do desenvolvedor e minimizando problemas comuns de gerenciamento de memória, como vazamentos.
- Tipagem Forte: C# é fortemente tipada, o que significa que os tipos de dados são rigorosamente controlados pelo compilador, ajudando a evitar erros comuns e a garantir maior segurança e precisão no código.
- Foco em Segurança: Projetada com segurança em mente, C# inclui mecanismos internos para proteger dados e evitar vulnerabilidades, tornando-a uma excelente escolha para o desenvolvimento de aplicações em ambientes empresariais e críticos.

Para programar em C#, o Visual Studio é a IDE (Ambiente de Desenvolvimento Integrado) recomendada. Oferecida pela Microsoft, essa ferramenta oferece uma ampla gama de recursos, como edição avançada de código, depuração, design de interfaces gráficas, integração com bancos de dados e suporte a diversas tecnologias e frameworks. Isso faz do Visual Studio uma poderosa plataforma para desenvolvedores, facilitando o processo de desenvolvimento e aprimorando a produtividade.

Uma das grandes evoluções da linguagem é que, além de ser poderosa e flexível, C# é *open-source*, o que permite a contribuição e o uso livre por desenvolvedores de todo o mundo. Isso significa que qualquer pessoa pode acessar, modificar e melhorar a linguagem e seus frameworks, promovendo maior transparência, inovação e colaboração na comunidade de desenvolvimento. A Microsoft abriu o código da linguagem e de seu ecossistema por meio da .NET Foundation, facilitando o uso de C# em diferentes plataformas, como Windows, macOS e Linux.

2.3.6 Padrão Arquitetural MVC (Moldal – View - Controller)

De acordo com Erich Gamma, Richard Helm, Ralph Johnson e John Vlissides (1994) o padrão de arquitetura MVC (Model-View-Controller) organiza o software em três componentes principais para promover a separação de responsabilidades e facilitar a manutenção:

- *Model* (Modelo): Gerencia os dados e a lógica de negócio, independente da interface com o usuário. Esse componente lida com as regras de negócio e notifica os outros componentes quando há mudanças, mantendo a interface atualizada.
- *View* (Visão): Exibe os dados e responde às interações do usuário. Observa o modelo e apresenta as informações conforme o estado atual dos dados, assegurando que a interface seja sempre uma representação atualizada do modelo.
- *Controller* (Controlador): Atua como mediador entre a visão e o modelo, processando as ações do usuário, que são traduzidas em mudanças no modelo e atualizações na visão.
- O MVC permite a independência de desenvolvimento e teste de cada camada, aumentando a modularidade e a escalabilidade do software. É um padrão essencial para sistemas interativos e de fácil manutenção.

2.4 BANCO DE DADOS

2.4.1 Definição

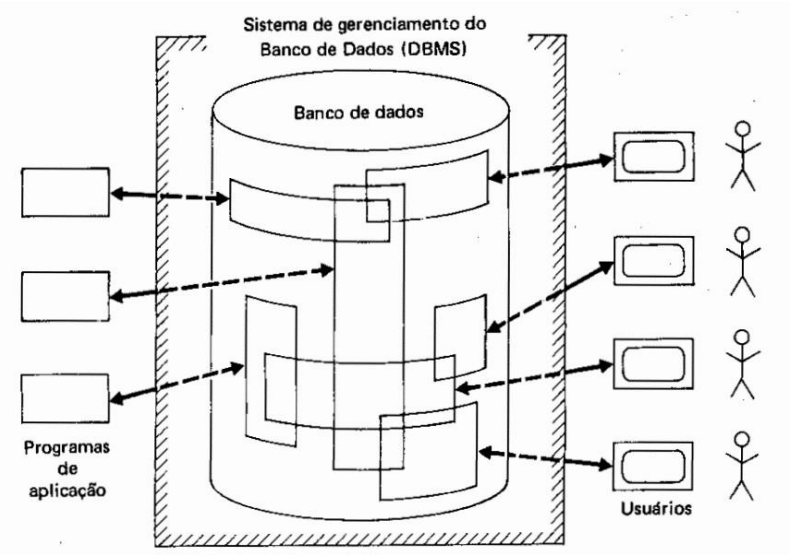
De acordo com Silberschatz, Korth e Sudarshan (2019) um Sistema de Gerenciamento de Banco de Dados (SGBD) é uma coleção de dados inter-relacionados e um conjunto de programas para acessar esses dados. A coleção de dados, geralmente chamada de banco de dados, contém informações relevantes para uma organização. O principal objetivo de um SGBD é fornecer uma maneira conveniente e eficiente de armazenar e recuperar informações do banco de dados.

Os sistemas de banco de dados são projetados para gerenciar grandes volumes de informações. A gestão dos dados envolve tanto a definição de estruturas para o armazenamento de informações quanto a disponibilização de mecanismos para a manipulação dessas informações. Além disso, o sistema de banco de dados deve garantir a segurança das informações armazenadas, mesmo em caso de falhas no sistema ou tentativas de acesso não autorizado. Quando os dados são compartilhados entre vários usuários, o sistema deve evitar possíveis resultados anômalos.

Dado o valor da informação para a maioria das organizações, cientistas da computação desenvolveram um vasto conjunto de conceitos e técnicas para o gerenciamento de dados.

A figura 2 representa a arquitetura de um Sistema de Gerenciamento de Banco de Dados (DBMS)**.** Nela, os programas de aplicação se comunicam com o banco de dados por meio do DBMS, que gerencia o acesso, a integridade e a segurança das informações. Os usuários finais acessam os dados processados através desses programas. O DBMS atua como intermediário entre os dados armazenados e os usuários, garantindo eficiência, organização e controle sobre as operações realizadas no banco.

Figura 2 – Banco de dados e suas componentes



FONTE: Date (1985).

De acordo com Silberschatz, Korth e Sudarshan (2019) diz também que os bancos de dados surgiram com o propósito de subsistirem os velhos armários de arquivos, porém trabalhando de maneira semelhante. Os softwares de gerenciamento de banco de dados (SGDBs) surgiram anos mais tarde devido à problemas com desempenho, precisão e confiabilidade na manipulação dos dados.

2.4.2 Abstração dos dados

Para garantir a eficiência no acesso aos dados e simplificar a interação dos usuários com o sistema, segundo Silberschatz, Korth e Sudarshan (2019), os bancos de dados são organizados em três níveis de abstração:

Nível físico: Descreve como os dados são armazenados de forma detalhada em estruturas de baixo nível.

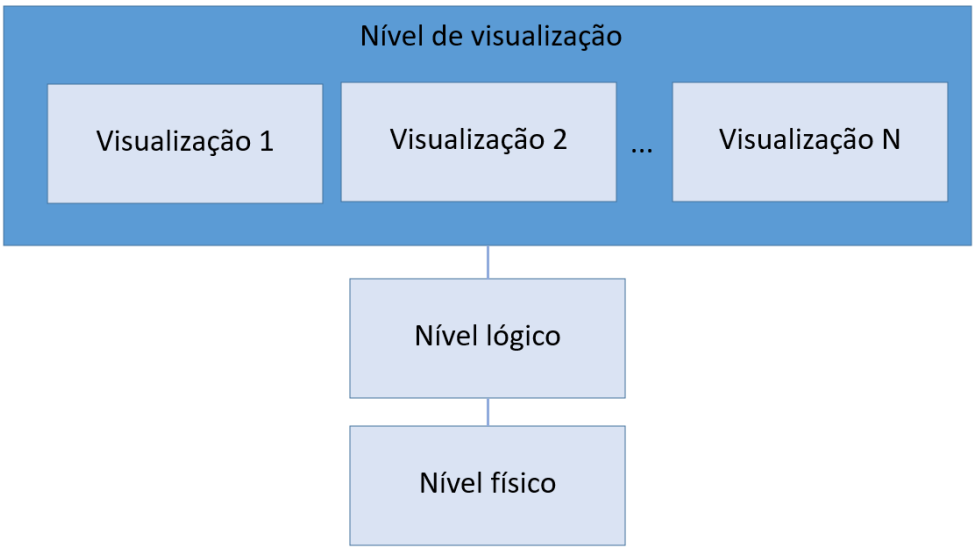
Nível lógico: Mostra quais dados são armazenados e os relacionamentos entre eles, utilizando estruturas mais simples, ocultando a complexidade do nível físico.

Nível de visão: Apresenta apenas partes específicas do banco de dados, adaptadas às necessidades dos usuários, tornando a interação mais simples e direta.

Esses níveis permitem que os usuários utilizem o sistema de forma eficiente sem lidar com a complexidade interna dos dados.

A Figura 8 ilustra a relação entre os três níveis de abstração.

Figura 3 – Níveis de abstração banco de dados



FONTE: Adaptado de Database System Concepts 6th edition

2.4.3 Instâncias e esquemas

De acordo com Silberschatz, Korth e Sudarshan (2019) os dados dos Bancos de dados (DB), mudam o tempo todo, seja deletar ou inserir. A coleção de informações armazenadas no banco de dados em um determinado momento é chamada de instância. Já o design do Banco de dados é chamado de esquema.

O conceito de esquema e instancia pode ser entendido através da associação com um programa em linguagem de programação. Um esquema corresponde a declaração de uma variável em um programa. O valor da variável a cada momento é o que podemos chamar de instância.

Os bancos de dados possuem diversos esquemas, dividido de acordo com suas camadas de abstração. O esquema físico descreve o banco em nível físico, assim como o esquema lógico descreve-o em nível lógico. No nível de visualização existem diversos esquemas, sendo também chamados de subesquemas, que descrevem as diferentes visualizações do banco.

2.4.4 Modelo de dados

De acordo com Silberschatz, Korth e Sudarshan (2019) os modelos de dados são ferramentas conceituais para descrever os dados, seus relacionamentos, semânticas e restrições de consistência em um banco de dados. Eles são essenciais para projetar bancos de dados nos diferentes níveis (físico, lógico e de visão). Existem quatro categorias principais de modelos de dados:

- Modelo Relacional: Usa tabelas para representar dados e seus relacionamentos. Cada tabela possui colunas e registros. Este é o modelo mais usados atualmente.
- Modelo Entidade-Relacionamento (E-R): Focado em entidades e os relacionamentos entre elas. Amplamente utilizado no design de banco de dados.
- Modelo Baseado em Objetos: Inspirado em linguagens de programação orientadas a objetos, combinando métodos e encapsulamento com atributos de dados.

- Modelo Semiestruturado: Permite que dados do mesmo tipo tenham diferentes conjuntos de atributos, comumente representados em XML.

Esses modelos ajudam a estruturar e gerenciar dados de forma eficiente em sistemas de banco de dados modernos.

2.4.5 Banco de dados Relacional

O modelo relacional é amplamente utilizado em aplicações comerciais de processamento de dados devido à sua simplicidade, que facilita o trabalho dos programadores em comparação com modelos anteriores, como o modelo de rede e o modelo hierárquico.

- Estrutura de Bancos de Dados Relacionais:

Um banco de dados relacional é composto por uma coleção de tabelas. Cada tabela tem um nome único e contém linhas (ou tuplas) e colunas (ou atributos).

Por exemplo, a tabela de instrutores contém as colunas ID, nome, departamento e salário. Cada linha da tabela representa um instrutor específico com essas informações.

A tabela cursos armazena informações de cursos, como ID do curso, título, departamento e créditos.

O conceito de relação em matemática está diretamente relacionado ao conceito de tabela no modelo relacional, e uma linha da tabela é chamada de tupla.

- Domínios e Valores Atômicos:

Cada coluna de uma tabela tem um domínio, que é o conjunto de valores permitidos para aquela coluna. Por exemplo, o domínio da coluna salário inclui todos os valores possíveis de salários.

No modelo relacional, os domínios devem ser atômicos, ou seja, os valores são indivisíveis. Por exemplo, se armazenamos números de telefone como um único valor, o domínio é considerado atômico. Mas se for dividido o número em partes (código de área, número local), ele deixaria de ser atômico.

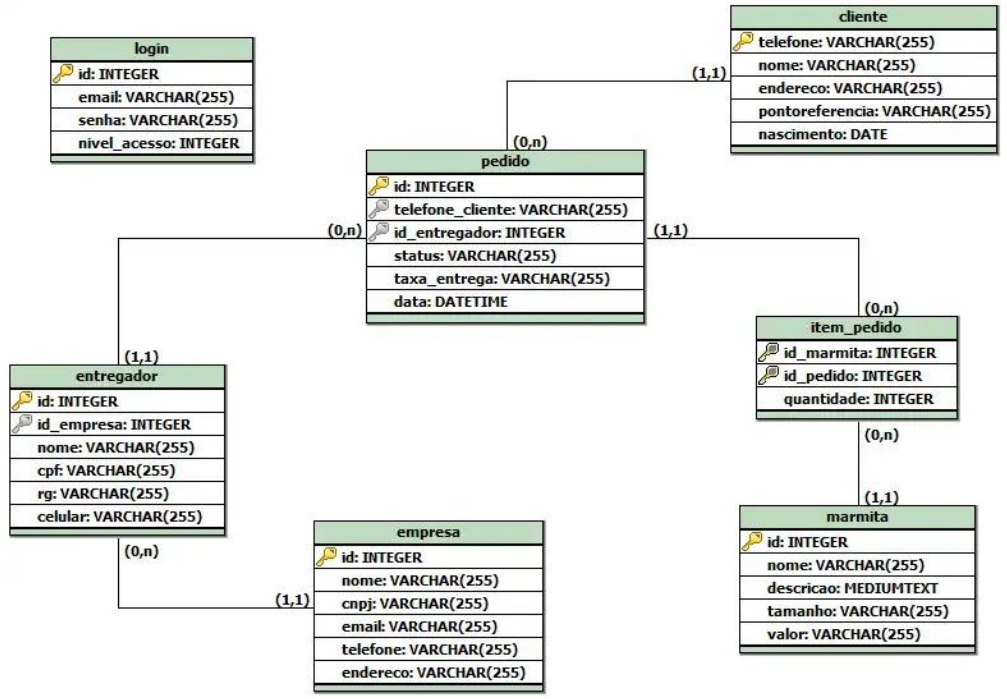
- Valores Nulos:

Um valor nulo representa a ausência ou o desconhecimento de um valor. Por exemplo, um instrutor pode não ter um número de telefone, então um valor nulo seria usado. Contudo, os valores nulos podem gerar desafios em operações de acesso e atualização no banco de dados.

Esses conceitos são a base para o funcionamento de bancos de dados relacionais.

Os bancos de dados relacionais são estruturados em tabelas, conforme o exemplo ilustrado na figura 4.

Figura 4 - Tabelas relacionais



Fonte: DIO – Digital Innovation One. Banco de dados relacional: fundamentos e aplicações

As requisições de dados são feitas por meio de códigos específicos chamados queries, onde é possível buscar os dados armazenados nas tabelas de maneira lógica, relacionando os dados de uma com a outra, retornando as informações desejadas.

Exemplo de “*Query*”:

Com base no diagrama de tabelas fornecido, pode-se consultar o nome do cliente, o status do pedido, o nome da marmita e o nome do entregador para cada pedido feito, utilizando uma query SQL exemplificado na querry 1.

Querry 1 – Exemplo de querry

```
"SELECT cliente.nome AS NomeCliente,
       pedido.status AS StatusPedido,
       marmita.nome AS NomeMarmita,
       entregador.nome AS NomeEntregador
FROM pedido
INNER        JOIN        cliente        ON
pedido.telefone_cliente = cliente.telefone
INNER  JOIN  item_pedido  ON  pedido.id  =
item_pedido.id_pedido
INNER        JOIN        marmita        ON
item_pedido.id_marmita = marmita.id
INNER       JOIN       entregador       ON
pedido.id_entregador = entregador.id;"
```

Fonte: O autor (2025).

A tabela 1 mostra o resultado da requisição ao banco de dados.

Tabela 1 - Resultado da requisição

CLIENTE	STATUS_PEDIDO	MARMITA	ENTREGADOR
JOÃO SILVA	Entregue	Marmita Média	Carlos Souza
MARIA LIMA	Pendente	Marmita Grande	José Santos

Neste exemplo, a requisição estabelece o relacionamento entre as tabelas de pedido, cliente, item do pedido, marmita e entregador, retornando informações combinadas de diversas tabelas inter-relacionadas, conforme as chaves primárias e estrangeiras definidas no modelo de banco de dados relacional.

2.4.6 Bancos de dados não-relacional

De acordo com Elmasri e Navathe (2011), os bancos de dados não relacionais, também chamados de "*NoSQL*", surgiram para atender a demandas específicas de aplicações que exigem grande volume de dados, alta velocidade de processamento e escalabilidade. Enquanto os bancos de dados relacionais organizam dados em tabelas estruturadas, com um esquema rígido e pré-definido, os bancos de dados não relacionais oferecem maior flexibilidade ao armazenar dados em formatos variados, como documentos, grafos ou pares chave-valor.

- Principais Tipos de Bancos de Dados Não Relacionais

Existem diversos tipos de bancos de dados, entretanto esses são os mais utilizados:

Bancos de Dados de Documentos: Armazenam dados em documentos (geralmente JSON, BSON ou XML), cada um com uma estrutura flexível. Exemplos: MongoDB e CouchDB.

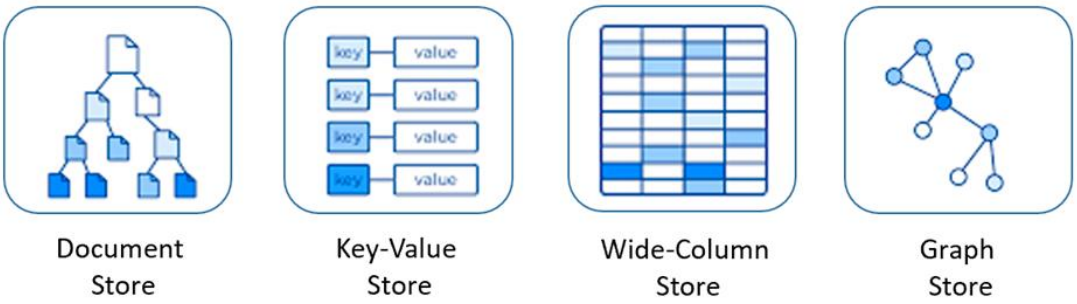
Bancos de Dados de Grafos: Modelam dados em nós (entidades) e arestas (relacionamentos), sendo ideais para representar relações complexas. Exemplos: Neo4j e ArangoDB.

Bancos de Dados Chave-Valor: Funcionam como um dicionário, onde cada chave está associada a um valor específico. São eficientes em operações simples. Exemplos: Redis e DynamoDB.

Bancos de Dados em Colunas: Armazenam dados por colunas ao invés de linhas, otimizando consultas de grandes volumes de dados. Um exemplo é o Cassandra DB.

A figura 5 apresenta os principais tipos de bancos de dados não-relacional, que se diferenciam pela forma como armazenam e organizam os dados. O *Document Store* utiliza documentos, geralmente em formato JSON, para guardar dados de maneira flexível. O *Key-Value* Store funciona com pares de chave e valor, sendo eficiente para buscas simples e rápidas. O *Wide-Column* Store armazena dados em colunas, oferecendo alta escalabilidade e bom desempenho em grandes volumes. Já o *Graph* Store representa os dados como nós e conexões, sendo ideal para modelar e consultar relacionamentos complexos.

Figura 5 – banco de dados não-relacional



FONTE: 5 conceitos da análise de dados que vão mudar a forma como você vê o mundo, 2024.

2.4.7 Diferenças Principais Entre Bancos de Dados Relacionais e Não Relacionais

- Estrutura de Dados:

No relacional organizam dados em tabelas com um esquema rígido (modelo predefinido), garantindo integridade e consistência, enquanto não relacional usa estruturas flexíveis (documentos, chave-valor, grafos), permitindo a inclusão de dados sem alterações no esquema.

- Escalabilidade:

No relacional a escalabilidade vertical (aumento de capacidade em um único servidor), enquanto no não relacional a escalabilidade é horizontal (distribuição de dados e processamento em vários servidores).

- Flexibilidade:

No relacional requer um esquema rígido para garantir a integridade dos dados, já no não relacional oferece flexibilidade, permitindo que dados diferentes coexistam sem restrições de formato.

- Consultas:

O relacional utiliza a linguagem SQL (*Structured Query Language*) para consultas complexas. O Não Relacional utiliza APIs específicas ou linguagens próprias, como a de consulta do MongoDB.

- Consistência e Disponibilidade:

O relacional: Baseia-se no modelo ACID (Atomicidade, Consistência, Isolamento e Durabilidade) para garantir a integridade transacional. Não Relacional

segue o modelo BASE (Basicamente Disponível, Estado Flexível, Consistência Eventual), priorizando disponibilidade e desempenho.

Assim, enquanto os bancos de dados relacionais são mais adequados para aplicações que exigem transações seguras e consistentes, os bancos não relacionais são ideais para sistemas distribuídos que processam grandes volumes de dados com estruturas flexíveis (Elmasri & Navathe, 2011).

A figura 6 detalha as diferenças entre os bancos de dados relacionais e não relacionais.

Figura 6 – Relacional vs Não Relacional

FACEBOOK. Imagem sobre SQL e NoSQL. 2025

2.5 RELES

De acordo com Littelfuse (2003) e Mohan, Undeland e Robbins (2003), os relés foram inicialmente desenvolvidos para amplificar sinais em longas distâncias, especialmente no telégrafo, onde era fundamental garantir a transmissão eficiente de mensagens. Com o tempo, os relés se tornaram elementos essenciais em circuitos de controle e automação, sendo usados para ligar e desligar circuitos de alta potência por meio de sinais de baixa potência.

Conforme discutido por Horowitz e Hill (2015), a introdução dos transistores permitiu que algumas funções dos relés fossem desempenhadas por componentes eletrônicos, particularmente em circuitos de baixa potência. No entanto, os relés continuam amplamente empregados em aplicações industriais, onde o isolamento elétrico é essencial e onde há necessidade de lidar com correntes e tensões elevadas.

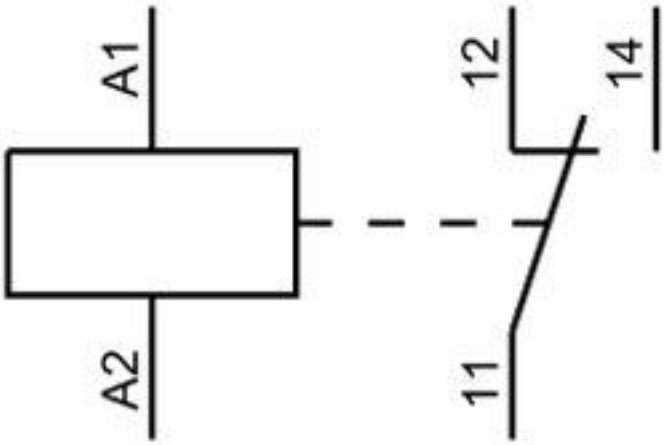
A figura 7 mostra a foto com detalhes de um relé de uso industrial que foi utilizado no projeto.

Figura 7 - Relé Eletrônico

FONTE: Phoenix Contact. 2024.

E a figura 8 ilustra o digrama geral de um relé padrão IEC de uso industrial.

Figura 8 - Diagrama de um relé IEC



FONTE: Eletrônica-PT. 2024.

3 METODOLOGIA

O desenvolvimento do projeto será realizado de forma estruturada, dividido em seis etapas principais, visando garantir a organização, a eficiência e a precisão dos resultados.

3.1 LEVANTAMENTO DE REQUISITOS

A primeira etapa consistiu no levantamento dos requisitos do sistema. Foi realizada a seleção da máquina de costura industrial mais apropriada, considerando marcas amplamente utilizadas no setor, como *Brother, Juki e Singer*, reconhecidas pela robustez e confiabilidade em ambientes de alta produção.

Foram analisados os pontos a serem controlados durante o processo, definindo as variáveis críticas que o sistema de *Poka-Yoke* deveria monitorar, como:

- A quantidade de pontos de costura realizada por ciclo;
- O momento correto para liberação da costura;
- A identificação dos remates iniciais e finais;
- A ativação do motor principal da máquina.

Essas variáveis foram consideradas essenciais para garantir que a execução da costura ocorresse de forma correta, minimizando erros e assegurando a qualidade do produto final.

3.2 ELABORAÇÃO DO PROJETO ELÉTRICO

Na segunda etapa, foi elaborado o projeto elétrico do sistema. O diagrama contemplou todos os componentes envolvidos, incluindo uma fonte de alimentação de 24VDC da Siemens, adequada para ambientes industriais, e um PC industrial da Nodka, equipado com módulos de entradas e saídas digitais.

O sistema de entradas foi projetado para captar os seguintes sinais da máquina de costura:

- Remate inicial;
- Remate final;
- Pulso de contagem de pontos via sensor indutivo;
- Motor ligado.

As saídas digitais do PC passaram a controlar três sinais de liberação da costura, acionando relés industriais responsáveis por seccionar os fios de controle. Para as conexões, foram empregados conectores Heart padrão DB9 e DB37:

- O conector DB9 foi destinado ao controle do pedal da máquina, interceptando os fios 0, 1 e 2;
- O conector DB37 foi responsável pela recepção dos sinais de controle da máquina;
- O sinal de contagem de pontos foi gerado por um sensor indutivo posicionado próximo à roldana da máquina, detectando a passagem de um elemento metálico fixado no mecanismo.

3.3 CONSTRUÇÃO DO PAINEL ELÉTRICO

Após a definição do projeto, foi iniciada a construção do painel elétrico. A montagem física dos componentes foi realizada conforme o diagrama elétrico e seguindo as normas técnicas vigentes, como a NR-10, além das boas práticas de montagem industrial.

Os relés foram instalados em trilhos DIN e organizados de forma a facilitar futuras manutenções. Também foram instaladas tomadas e conectores DB9 e DB37 para possibilitar a integração rápida e segura com a máquina de costura.

3.4 TESTES DO PAINEL ELÉTRICO

Com o painel elétrico montado, foram realizados testes de validação. Todas as conexões foram verificadas, testando a continuidade dos cabos, a resposta dos relés e o funcionamento dos sinais de entrada e saída.

Testes simulados foram conduzidos com o uso de fontes de sinal e cargas, simulando o funcionamento real da máquina de costura. Essa etapa assegurou que o painel estivesse plenamente funcional antes da integração com o sistema completo.

3.5 DESENVOLVIMENTO DO SOFTWARE

O software foi desenvolvido utilizando a linguagem C#. O processo de desenvolvimento foi dividido nas seguintes etapas:

- Criação de classes de comunicação: Foram desenvolvidas classes específicas para:
    - Comunicação com a câmera Cognex In-Sight SnAPP, para aquisição de imagens e monitoramento visual;
    - Comunicação com as entradas e saídas digitais do PC industrial, utilizando bibliotecas do fabricante;
    - Gerenciamento do banco de dados SQL Server para armazenar dados de operação, receitas, usuários, imagens e configurações;
    - Configuração de etiquetas para impressão, definindo layout e dados a serem enviados.
- Testes unitários: Cada classe foi testada individualmente para garantir seu correto funcionamento antes da integração geral.
- Desenvolvimento das interfaces gráficas: As telas foram desenvolvidas em HTML integradas ao *backend*, permitindo:
    - Criação e edição de receitas de costura;
    - Configuração dos parâmetros gerais do sistema;
    - Execução e monitoramento do ciclo de costura em tempo real;
    - Configuração da câmera Cognex.
- Desenvolvimento do *backend*: O *backend* foi responsável por implementar a lógica do sistema, integrando banco de dados, periféricos e telas. Foram programados os ciclos de operação automática com monitoramento contínuo dos sinais e tomada de decisão em tempo real.
- Testes integrados: Após a finalização do desenvolvimento, testes completos de integração foram realizados, assegurando o funcionamento harmônico e eficiente do sistema.

3.6 INSTALAÇÃO E VALIDAÇÃO NO CLIENTE

Na etapa final, o sistema completo foi instalado no ambiente do cliente. Após a instalação física e elétrica, foram realizados testes de aceitação sob condições reais de produção.

Foram verificados:

- O funcionamento correto dos sinais de controle da máquina;
- A precisão na contagem de pontos de costura;

- O acionamento exato dos relés;
- A efetividade do sistema de *Poka-Yoke*;
- A comunicação com a câmera e captura das imagens;
- O armazenamento consistente dos dados no banco de dados.

Com base nos resultados obtidos, ajustes finais foram realizados, garantindo que o sistema atendesse a todos os requisitos especificados inicialmente.

4 DESENVOLVIMENTO

O desenvolvimento do sistema consiste em duas etapas principais:

1. Hardware: inclui a configuração e montagem do painel de controle.
2. Software: abrange a implementação de funcionalidades e integração com o banco de dados.

Primeiro é relacionado os recursos necessários para a montagem do painel e codificação do projeto.

4.1 RECURSOS NECECESSÁRIOS

4.1.1 Software

Para esta aplicação, foi decidido usar como sistema operacional o Windows 10 da Microsoft por questão de confiabilidade, facilidade e principalmente pelo conhecimento de todos os envolvidos no projeto.

Para a utilização das entradas e saídas digitais a Nodka fornece bibliotecas (DLLs) específicas para o funcionamento das mesmas, assim como software para teste e alguns exemplos de códigos em C# para facilitar a aplicação. Em anexo será colocado parte das funções disponibilizadas em linguagem C.

Então o material necessário foi basicamente:

- IDE: Microsoft Visual Studio (licença gratuita).
- Sistema Operacional: Windows 10 Profissional (licença vitalícia – R$ 699,00).
- Banco de Dados: Microsoft SQL Server (licença gratuita).

4.1.2 Hardware

Para esse projeto, foi utilizada uma máquina de uso industrial, equipada com servo motor, o que representou uma redução significativa nos custos de manutenção.

O uso do servo motor melhorou a precisão da máquina em comparação com os modelos equipados com embreagem.

O modelo de máquina adotado foi a “Durkopp Adler - 867-190445-M”, cujo controle era realizado por um controlador da “EFKA DC 1550”, utilizando o modelo de comando “DA321G5321”, ambos de uso industrial, conforme ilustrado nas imagens 9 e 10.

Figura 9 - Durkopp Abler 867-190445-M

FONTE: partner.duerkopp-adler.com

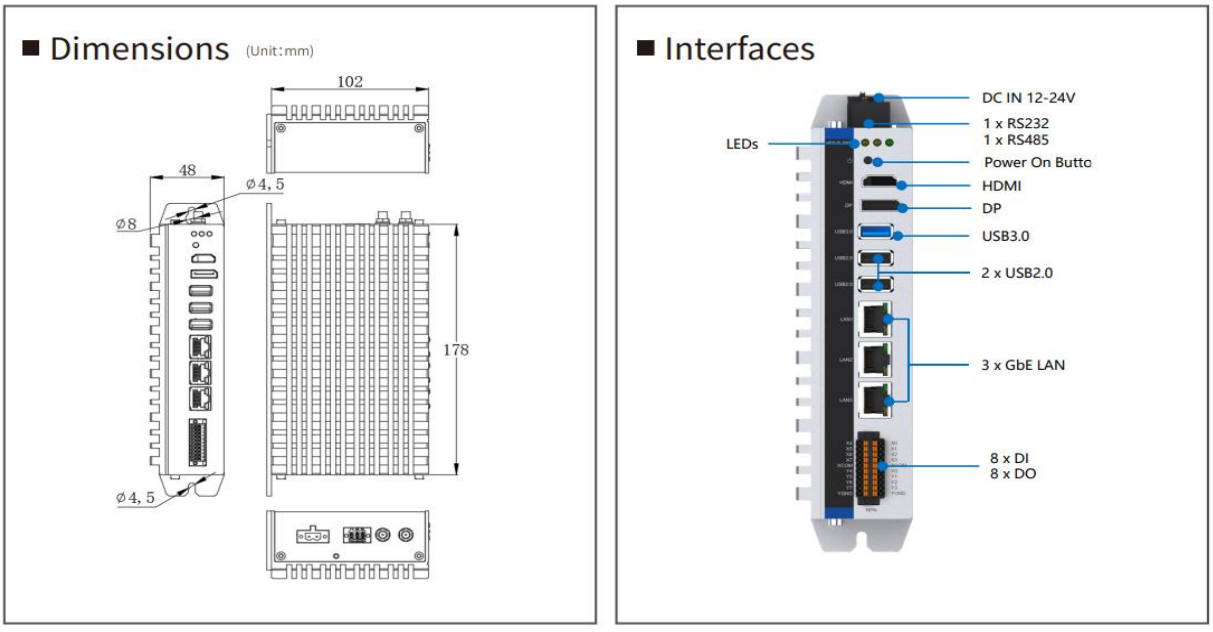
Figura 10 – Efka 1550 dc

FONTE: Manual efka – PT_BA_DA321G5321_1_2_190509_H.pdf

Tratando-se de hardware o computador utilizado contou com a seguinte configuração:

- CPU：Intel® Celeron J1900, 2.0GHz, 4 cores/4 threads, 2MB L2 cache/Intel® Celeron J6412, 2.0~2.6GHz, 4 cores/4 threads, 1.5MB L2 cache
- Memory：1 x SO-DIMM DDR3L-1333MHz (max. 16GB)/1 x SO-DIMM DDR4-2400MHz (max. 32GB)
- Storage Medium：1 x M.2 SSD 2242 SATA2.0/1 x M.2 SSD 2242
- BIOS：AMI UEFI 64Mbit/AMI UEFI 256Mb
- I/O Interface
- LAN：J1900:2 x RTL8111H GbE LAN,1 x Intel GbE LAN/J6412:3 x Intel GbE LAN
- USB：J1900:1 x USB3.0, 2 x USB2.0/J6412:3 x USB3.1
- COM：1 x RS232, 1 x RS485
- DO：1 x Programmable LED
- SSD: 512GB

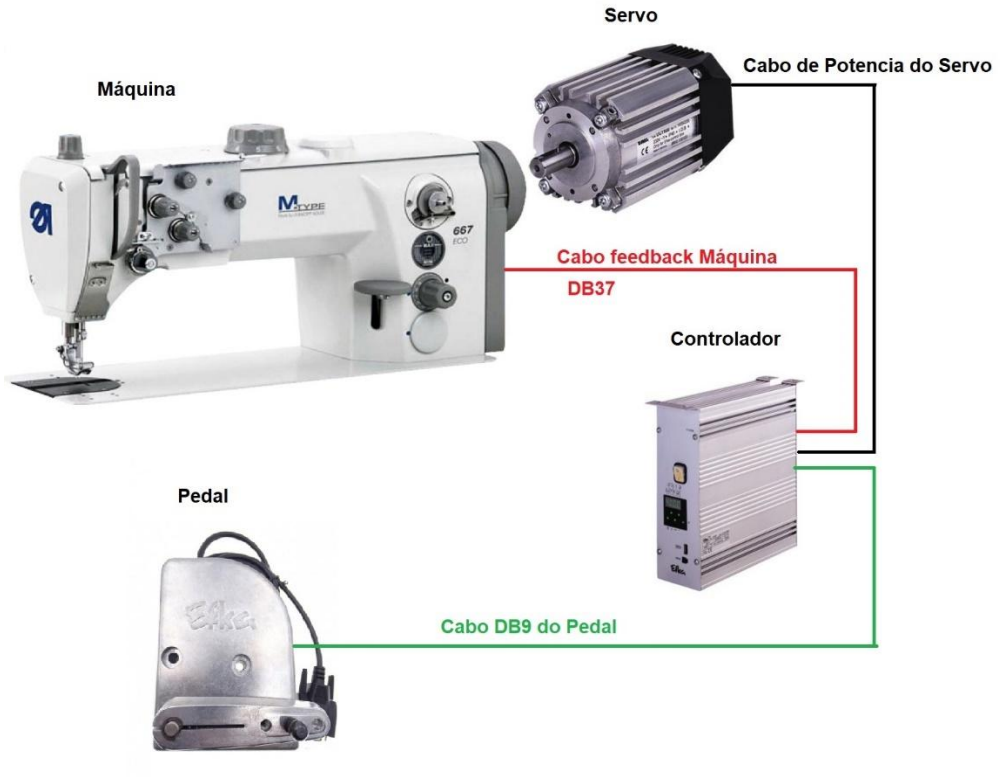
Figura 11 - Detalhes computador industrial NODKA



FONTE: Manual do produto: NP-6116-IO-J1900_Spec_EN.pdf

A configuração da máquina segue o esquema apresentado na Figura 12.

Figura 12 - Esquema máquina de Costura

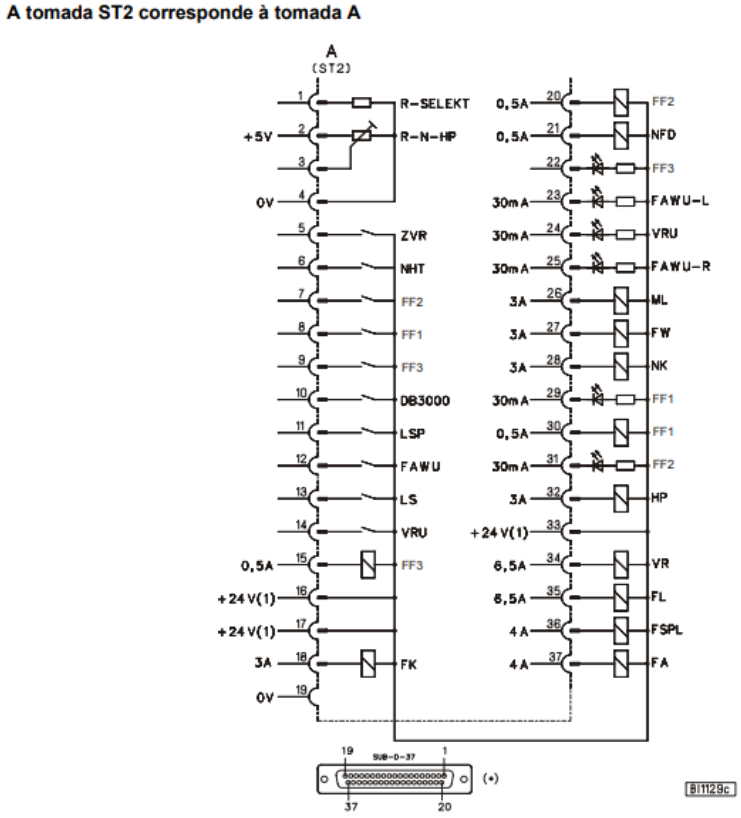


Fonte: O autor (2024).

O controle da comunicação com o pedal (identificado em verde) foi realizado por meio de um relé, que atua como um mecanismo principal de controle. O sistema foi equipado com entradas e saídas conectadas ao controlador por meio de cabos DB9 (em verde) e DB37 (em vermelho). Esses cabos foram integrados a um painel modular, permitindo facilmente conexão e desconexão, sem interferir na configuração original.

A figura 13 ilustra o esquema elétrico do conector DB37, onde é retirado para uso da aplicação as entradas: remate, remate status (final = ligado, inicial = desligado) e motor ligado.

Figura 13 - Esquema DB37



FONTE: Manual EFKA - PT_PL_DA321G5321_1_2__220107_H.

A figura 14 detalha as siglas dos pinos do conector DB37, retirada do manual da EFKA.

Figura 14 - Tabela Siglas DB37

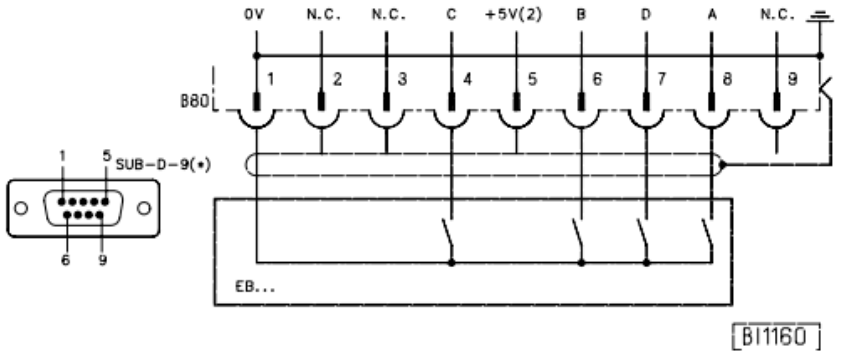
DB3000	Limitação de velocidade 3000 rpm	in8		LS	Fotocélula	in9	
FA	Corte de linha		M1	LSP	Bloqueio de marcha	in2	
FAWU	Detector de linha em baixo	in6		ML	Motor em marcha		M14
FAWU-L	Detector de linha em baixo à esquerda		M7	NFD	Pressão do pé calcador		M17
FAWU-R	Detector de linha em baixo à direita		M9	NHT	Agulha em cima/em baixo	in3	
FF1	Módulo de função A	in4	M6	NK	Refrigeração da agulha		M2
FF2	Módulo de função B	in1	M16	R-N-HP	Potenciómetro de valor teórico para limitação de velocidade dependente do curso		
FF3	Módulo de função C	in7	M12	R-SELEKT	Resistência para seleccão da máquina		
FK	Pinça de fixação de linhas		M13	TPW	Puller		M12/M9
FL	Elevação do pé calcador			VR	Remate		
FSPL	Supressão da tensão da linha		M4	VRU	Supressão / chamada do remate	i10	M8
FW	Limpa-linhas		M3	ZVR	Remate intermédiol	in5	
HP	Ajustamento de curso		M5				

1) Tensão nominal +24V, tensão de marcha em ponto morto ao máx. 30V durante pouco tempo após rede ligada
*) Perspectiva: Lado de encaixe da tomada / lado de soldadura da ficha

FONTE: Manual EFKA - PT_PL_DA321G5321_1_2__220107_H.

E a figura 15 mostra o Esquema elétrico do pedal representado em verde na figura 12:

Figura 15 - Esquema DB9 Pedal



EB.. Regulador de velocidade

Nível do pedal ➔	-2	-1	0	½	1	2	3	4	5	6	7	8	9	10	11	12
Entrada A	L	L	H	H	H	L	L	H	H	L	L	H	H	L	L	H
Entrada B	L	H	H	L	L	L	H	H	H	H	L	L	L	L	H	H
Entrada C	H	H	H	H	L	L	L	L	L	L	L	L	H	H	H	H
Entrada D	H	H	H	H	H	H	H	H	L	L	L	L	L	L	L	L

1) Tensão nominal +24V, tensão de marcha em ponto morto ao máx. 30V durante pouco tempo após rede ligada
2) Tensão nominal +5V, I_{max} = 20mA
3) Saída do nível lógico, especificação consoante HC74...
*) Perspectiva: Lado de encaixe da tomada / lado de soldadura da ficha

FONTE: Manual EFKA - PT_PL_DA321G5321_1_2__220107_H.

O pedal possui um funcionamento específico em que todas as entradas ficam em "High" no estado de repouso, devido à configuração NPN. Assim, ao interromper a conexão dos fios, todas as entradas permanecem no estado "High". Para travar o pedal foi necessário cortar o fio 4, 6 e 8 (Entradas A, B e C), onde é possível subir e descer a trava de costura, porém não é possível realizar os pontos, ou seja, o motor não é acionado.

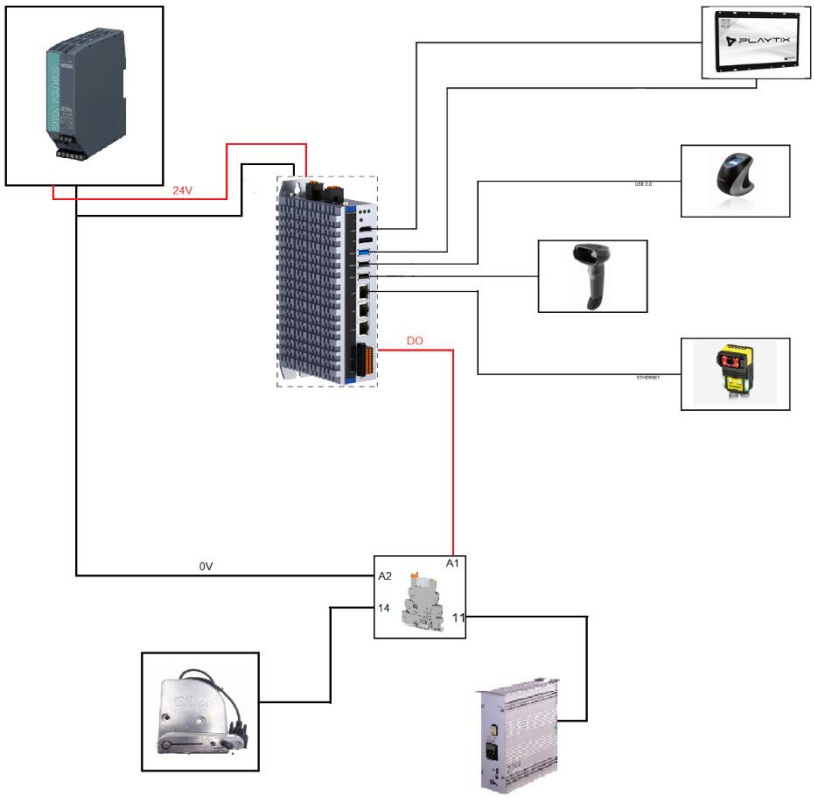
Na tabela 2 segue os demais recursos necessários para a realização do projeto.

Tabela 2 – Lista de equipamentos do painel

DESCRIÇÃO	PRODUTO	VALOR
MATERIA PRIMA	Cabo de Rede 5M	573,39
MATERIA PRIMA	Cabo de Alimentação 5M	469,07
SENSOR VISÃO	Sensor de Visão SNAP 2801	5312,93
CPU	Intel Celeron J6412 2.0GHz CPU/ 16GB DDR	1856,87
ARMARIO	Armário Compacto AX 500X500X210mm RAL 70	642,54
SINALEIRO	Sinaleiro MET 230V	61,48
DISJUNTOR MINIATURA	Disjuntor 1P C 6A	29,37
SECCIONADORA	Chave Seccionadora Siemens 5TW3032-3	257,56
MONITOR	Monitor Touch Screen Resistivo 15.6"	1999,9
LEITOR BIOMÉTRICO	Leitor Biométrico ID BIO - Control ID	438,83
LEITOR ZEBRA	Leitor Zebra DS2208 SR 2D	589
CABO	Cabo Manga 40x26AWG	604,8
	Total:	12835,74

A figura 16 representa o esquema simplificado do sistema.

Figura 16 - Esquema simplificado Hardware



FONTE: O Autor (2024).

4.1.3 Interligação do Sistema e Configuração dos Componentes

O sistema desenvolvido integra diversos dispositivos para garantir o controle do processo de costura industrial. A Figura 16 apresenta o diagrama simplificado de interligação do sistema, que inclui uma fonte de alimentação, controlador lógico programável (CLP), painel com toque na tela, um leitor de código de barras da Zebra, câmera de inspeção da Cognex, leitor biométrico da controlID e relés de interface.

Em sequência, detalha-se o funcionamento de cada componente e o procedimento de montagem.

4.1.4 Descrição do Funcionamento

- Fonte de Alimentação (24V): A fonte de alimentação fornece energia para os dispositivos do sistema, garantindo uma tensão contínua e estável. Ela é essencial para alimentar o controlador e os relés de interface, permitindo o funcionamento seguro e confiável dos componentes.
- Controlador Principal: O controlador centraliza as funções do sistema, sendo responsável por receber sinais de entrada (leitor de código de barras, câmera, leitor biométrico) e acionar as saídas digitais (relés). Ele também gerencia a comunicação com o painel e com os dispositivos conectados via Ethernet e USB.
- Painel: O painel serve como interface principal entre o operador e o sistema, permitindo a seleção de receitas, o monitoramento do processo e o controle de operações. Ele é conectado ao controlador e apresenta informações em tempo real, garantindo uma interação intuitiva.
- Leitor de código de barras Zebra: Este dispositivo é utilizado para capturar os códigos de barras das receitas, enviando os dados em formato de “string” ao controlador. Essa palavra gerada pelo leitor é responsável por chamar a receita deseja, onde foi criada pelo usuário.
- Câmera Cognex: A câmera realiza a validação da costura, capturando imagens para verificar a qualidade do produto final. Sua comunicação com o sistema ocorre via Ethernet, assegurando velocidade e precisão no envio de informações.
- Leitor Biométrico: O leitor biométrico é utilizado para autenticação de operadores, registrando as digitais e associando-as aos dados de operação no sistema. Ele é conectado ao controlador via USB.
- Relés de Interface: Conectados às saídas digitais do controlador, os relés atuam como intermediários para acionar dispositivos externos, como motores ou atuadores. Eles protegem o sistema contra sobrecargas e garantem a integridade dos componentes.

4.2 PROCEDIMENTO DE MONTAGEM

4.2.1 Planejamento e Organização do Painel

O painel de controle foi projetado com base nas normas técnicas de automação industrial, especialmente atendendo às diretrizes da NR-10 (Segurança em Instalações e Serviços em Eletricidade) e da NR-12 (Segurança no Trabalho em Máquinas e Equipamentos). Para a fixação dos componentes internos, foram utilizados trilhos DIN, que proporcionaram maior padronização, flexibilidade e facilidade de manutenção.

O espaço interno do painel foi planejado para comportar adequadamente todos os dispositivos, como fonte de alimentação, relés, módulos de entradas e saídas digitais e dispositivos de comunicação, garantindo a ventilação necessária e o acesso facilitado para intervenções técnicas.

4.2.2 Conexão Elétrica

- A alimentação dos dispositivos foi realizada por uma fonte industrial de 24V da Siemens, conectada aos barramentos de distribuição, alimentando tanto o computador industrial quanto os módulos de relés.
- A comunicação entre o painel e a máquina de costura ocorreu através de cabos equipados com conectores DB9 e DB37, assegurando estabilidade e robustez nas ligações elétricas.
- As entradas digitais que representavam sinais da máquina (Remate Inicial, Remate Final, Motor On e Sensor de Pontos) foram conectadas via DB37, enquanto as saídas digitais (responsáveis pela liberação da costura) foram acionadas por meio de relés controlados pelo computador industrial e conectadas via DB9.
- O acionamento das saídas foi realizado utilizando relés industriais, que fizeram a comutação dos fios provenientes do pedal da máquina, possibilitando a liberação controlada da costura conforme a lógica implementada no sistema.
- Para a contagem dos pontos de costura, foi instalado um sensor indutivo próximo à polia da máquina de costura. Esse sensor detecta a passagem de um elemento metálico fixado à polia, gerando pulsos contabilizados pelo sistema.

4.2.3 Integração dos Dispositivos

O sistema contou com um painel touchscreen como interface principal, oferecendo uma interação eficiente e intuitiva para o operador.
A seleção das receitas de costura foi realizada utilizando um scanner de código de barras da marca Zebra, que enviava os dados em formato de *string* diretamente ao sistema.

A validação da costura foi garantida por uma câmera Cognex da linha In-Sight SnAPP, conectada via Ethernet/IP, que foi responsável pela captura de imagens e verificação visual conforme os critérios estabelecidos na receita.

Para a autenticação dos operadores, foi utilizado um leitor biométrico conectado via USB, assegurando o controle de acesso e a rastreabilidade dos processos.

A centralização das funções de controle e armazenamento foi realizada por um computador industrial da Nodka, que gerenciou as entradas e saídas digitais, a comunicação com periféricos e o ciclo operacional completo da máquina.

4.2.4 Procedimentos de Segurança

Antes do início da montagem elétrica e mecânica, foram seguidos rigorosamente os seguintes procedimentos de segurança:

- A instalação elétrica foi realizada por profissionais capacitados e treinados conforme a NR-10.
- Todos os dispositivos foram desenergizados antes da manipulação de fiação ou componentes internos.
- É obrigatório o uso de Equipamentos de Proteção Individual (EPIs), tais como luvas isolantes, óculos de proteção, capacete e vestimentas adequadas.
- A montagem e interligações seguiram esquemas elétricos atualizados, validados previamente pelo engenheiro responsável.
- Foi realizada a conferência dos cabos e conexões quanto à continuidade, polaridade e isolamento elétrico antes da energização do sistema.

- Dispositivos de proteção, como disjuntores e fusíveis, foram instalados conforme as especificações do projeto elétrico para evitar curtos-circuitos e sobrecargas.
- A fixação mecânica de todos os módulos e componentes foi verificada para garantir que suportem vibrações e esforços do ambiente industrial.
- Testes iniciais foram realizados com cargas simuladas para evitar danos ao equipamento principal.

4.2.5 Instalação

A instalação física dos dispositivos será realizada de acordo com o diagrama elétrico apresentado na Figura 15 e ao apêndice 3 – Esquema elétrico. A disposição dos componentes e o roteamento dos cabos internos seguiram o layout mostrado na Figura 16, fornecido pela empresa responsável pelo desenvolvimento do projeto.

Durante a instalação, foram adotadas práticas como a utilização de canaletas para organização dos cabos, identificação clara dos fios e atenção especial ao aterramento dos equipamentos, assegurando a confiabilidade e a segurança da operação.

A correta montagem e instalação do painel de controle forma fundamentais para assegurar a operação segura, confiável e eficiente do sistema. O atendimento rigoroso às normas regulamentadoras NR-10 e NR-12 não apenas protege os operadores contra riscos elétricos e mecânicos, mas também assegura a conformidade legal do projeto.

Após a montagem, foram realizados testes de verificação elétrica, comunicação dos dispositivos e validação funcional de todo o sistema, no qual foi necessária para identificar possíveis falhas antes da energização definitiva e garantir o perfeito funcionamento da solução no ambiente de produção.

4.3 SOFTWARE

O software foi desenvolvido utilizando a arquitetura MVC (*Model-View-Controller*), a qual promove uma separação clara entre a interface do usuário, a lógica de negócios e o acesso a dados. Essa abordagem foi escolhida devido à sua eficiência na organização de sistemas de grande porte, proporcionando facilidade de manutenção, escalabilidade e legibilidade do código. A seguir, são detalhados o

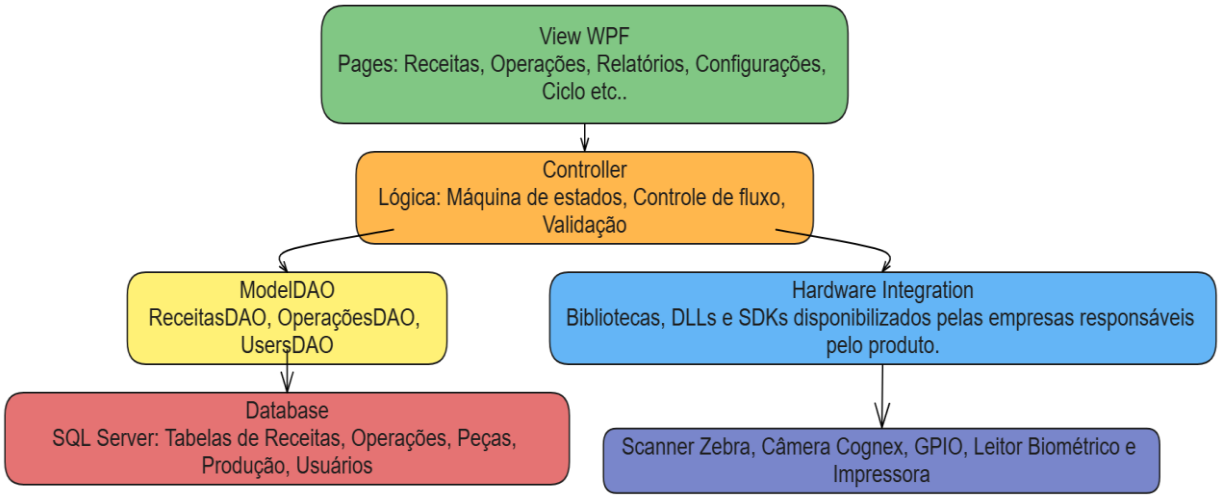
funcionamento do sistema, com foco nas funcionalidades implementadas, bem como a integração com dispositivos de hardware e com o banco de dados.

4.3.1 Arquitetura Geral

O sistema foi projetado com uma estrutura modular para atender às demandas do processo de costura industrial. A camada Model é responsável por representar os dados e a lógica de negócios, incluindo entidades como receitas, peças, operações, produção e usuários. Essas entidades estão diretamente ligadas ao banco de dados SQL Server, e sua manipulação é realizada por meio de classes DAO (Data Access Object). A camada View, construída com WPF, gerencia a interface com o usuário, fornecendo páginas dinâmicas e interativas que permitem a navegação e o controle de todas as funcionalidades do sistema. Já a camada Controller atua como mediadora entre o Model e a View, garantindo que as operações solicitadas pelo usuário sejam processadas corretamente e que as interfaces sejam atualizadas com os dados mais recentes.

No diagrama 1 relaciona de forma esquemática o modelo proposto do projeto.

Diagrama 1 - Diagrama de camadas



FONTE: O autor.

4.3.2 Principais Funcionalidades do Sistema

O software é composto por uma série de funcionalidades interligadas que cobrem todas as etapas do processo de costura, desde o gerenciamento de dados até a execução do ciclo de produção. A interface inicial serve como um ponto central de navegação, exibindo os logotipos das empresas envolvidas e os botões que levam às páginas de receitas, produção, relatórios, manutenção e configurações.

A página de receitas é um dos principais componentes do sistema, onde os usuários podem criar, editar e excluir receitas. Essas receitas contêm informações detalhadas, como nome, código de barras, peças e operações associadas. A interface exibe as receitas em uma *TreeView*, facilitando a navegação e seleção. Além disso, há botões que permitem acessar rapidamente o gerenciamento de peças e operações, que são elementos essenciais para a execução do ciclo de costura.

No gerenciamento de peças, os usuários podem cadastrar informações como nome, número da peça e projeto. Essas peças são posteriormente associadas às receitas, permitindo rastreabilidade e organização. Já no gerenciamento de operações, o sistema oferece uma interface para configurar os passos específicos do ciclo de costura, como remates, pontos e inspeções. A configuração é feita de maneira dinâmica, possibilitando que os operadores insiram sequências diretamente nas operações listadas.

Uma das funcionalidades mais importantes é a página do ciclo de produção. Nesta etapa, o sistema utiliza uma máquina de estados assíncrona para gerenciar a execução das operações de forma não bloqueante. O ciclo começa com a leitura do código de barras pelo scanner Zebra, que identifica a receita correspondente. Em seguida, as operações configuradas são executadas em sequência, incluindo a validação de qualidade realizada pela câmera Cognex. Todo o processo é registrado para análise posterior, garantindo controle completo do ciclo.

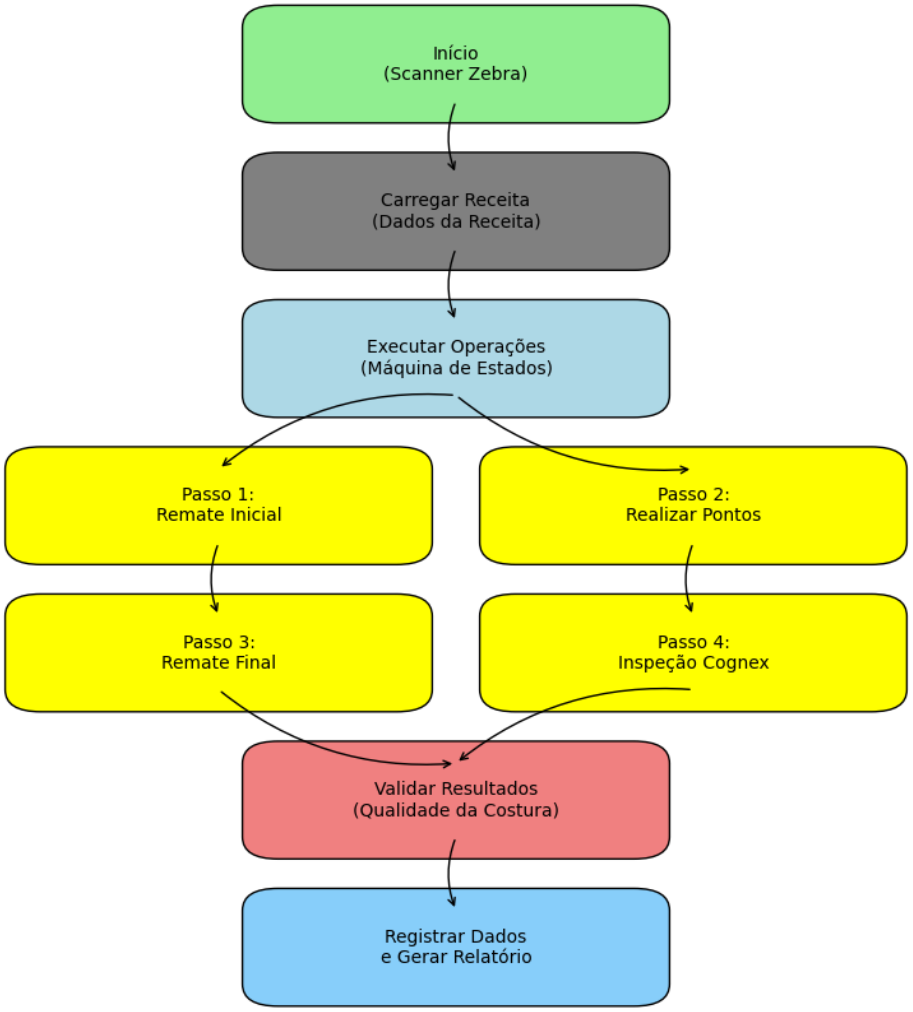
A geração de relatórios é realizada por meio de uma página dedicada, onde os usuários podem exportar *PDF* com informações detalhadas das costuras realizadas. A funcionalidade permite aplicar filtros, como intervalo de datas, para uma análise mais precisa dos dados.

O diagrama 2 representa o fluxo de operação automatizada de costura com controle de qualidade, provavelmente utilizado em uma célula industrial. O processo se inicia com a leitura de uma receita por meio de um scanner Zebra, responsável por

identificar qual costura será realizada. Em seguida, os dados da receita são carregados e processados por uma máquina de estados, que controla a sequência das ações.

A operação é dividida em quatro passos principais: o remate inicial (preparação da costura), a realização dos pontos, o remate final (fechamento da costura), e por fim, a inspeção com câmera industrial Cognex para verificar se a execução foi adequada. Após isso, os resultados da costura são validados com base na qualidade, e por fim, os dados são registrados e um relatório é gerado automaticamente.

Diagrama 2 - Ciclo máquina



Fonte: O autor.

4.3.3 Integração com Hardware

A integração com dispositivos de hardware é um aspecto fundamental do sistema, garantindo a automação e eficiência do processo de costura. Um dos dispositivos principais é o scanner Zebra, responsável por capturar os códigos de barras das receitas. A comunicação com o scanner é feita por meio de uma API personalizada em C#, que processa as leituras e as associa às receitas cadastradas no banco de dados.

A câmera Cognex desempenha um papel crucial na validação da qualidade do produto. Ela é configurada e controlada pela página de configurações / Câmera, que utiliza um WebView2 para exibir uma aplicação web dedicada. A comunicação com a câmera ocorre via TCP/IP, permitindo o envio de comandos, como TRIGGER, e a recepção de resultados de inspeção em tempo real. Essa integração assegura que cada costura seja inspecionada com precisão, atendendo aos padrões de qualidade definidos.

Outro componente essencial é o leitor biométrico, utilizado para autenticação de usuários. O sistema permite registrar e validar digitais, garantindo que apenas operadores autorizados tenham acesso a determinadas funcionalidades. A implementação é feita por meio de DLLs fornecidas pelo fabricante, acessadas via chamadas nativas em C#.

O controle de entradas e saídas digitais (GPIO) é gerenciado pela classe GPIOController, que utiliza a API da Nodka para comunicação direta com o hardware. A configuração dos canais é feita dinamicamente pelo arquivo nkio_config.ini, permitindo adaptar o sistema a diferentes configurações de hardware. As entradas digitais são monitoradas para validar sensores conectados, enquanto as saídas são ativadas para realizar ações como iniciar uma operação ou sinalizar o status de um processo.

Para garantir o funcionamento adequado de todos esses dispositivos, o sistema inclui uma página de manutenção, onde os operadores podem testar manualmente as entradas e saídas digitais. Essa funcionalidade é crucial para diagnósticos e resolução de problemas, garantindo a integridade do sistema em ambientes industriais exigentes.

4.3.4 Banco de Dados e Acesso a Dados

O sistema utiliza o SQL Server para armazenar dados como receitas, peças, operações, produções e usuários. A comunicação com o banco é feita por meio do *Entity Framework (EF)*, um ORM que permite manipular dados utilizando classes e objetos, sem necessidade de escrever SQL diretamente.

4.3.4.1 Utilização do Entity Framework

O EF está configurado para mapear entidades (exemplo: Receita, Part, Operação) para tabelas no banco. Exemplo de entidade conforme ilustrado no código 1.

Código 1 – Exemplo de entidade.

```
public class Receita
{
    public int ReceitaID { get; set; }
    public string Nome { get; set; }
    public string Barcode { get; set; }
    public int PartID { get; set; }
    public virtual Part Part { get; set; }
}
```

Fonte: O autor (2025).

4.3.4.2 Operações CRUD com EF:

O Entity Framework simplifica a criação, leitura, atualização e exclusão (CRUD) de dados no banco. Por exemplo, a inserção de uma “Receita” de acordo com o código 2.

Código 2 – Exemplo de inserção de receita.

```
using (var context =
SingletonContext.Instance)
{
    var receita = new Receita { Nome =
"Costura A", Barcode = "123456789", PartID
= 1 };
    context.Receitas.Add(receita);
    context.SaveChanges();
}
```

Fonte: O autor (2025).

No código 3 é exemplificado a consulta com navegação:

Código 3 – Exemplo consulta com navegação.

```
using (var context =
SingletonContext.Instance)
{
    var receitas =
context.Receitas.Include(r =>
r.Part).ToList();
}
```

Fonte: O autor (2025).

4.3.4.3 DAO (*Data Access Object*)

Para manter o código organizado, cada entidade possui uma classe DAO com os métodos de acesso ao banco, observe o código 4.

Código 4 – Exemplo de acesso com classe DAO.

```
ReceitaDAO:
public class ReceitaDAO
{
    public void InsertReceita(Receita
receita)
    {
        using (var context =
SingletonContext.Instance)
        {
            context.Receitas.Add(receita);
            context.SaveChanges();
        }
    }

    public List<Receita> GetReceitas()
    {
        using (var context =
SingletonContext.Instance)
        {
            return
context.Receitas.Include(r =>
r.Part).ToList();
        }
    }
}
```

Fonte: O autor (2025).

OperacaoDAO:

- AdicionarOuAtualizarOperacao(): insere ou atualiza.
- DeletarOperacao(): remove operação e suas sequências.
- Métodos de busca síncrona e assíncrona.

PartDAO:

- adicionaPart(): insere nova peça.
- RemovePart(): exclui peça.
- RetornarPart(): retorna lista de peças.

ProductionDAO:

- AddProductionAsync(): insere ou atualiza produção.
- GetProductionsByDateRange(): busca por intervalo de datas.
- UserDAO
- CadastrarUsuario() / AtualizarUsuario(): gerencia cadastro.
- DeletarUsuario(): remove usuários.
- BuscarUsuarioPorID() / PorLoginSenha(): autenticação.

4.3.4.4 SingletonContext

Para evitar múltiplas conexões simultâneas, é utilizado um contexto singleton mostrado no código 5, de forma a confirmar uma única conexão:

Código 5 – Implementação do padrão *Singleton* para o contexto de banco de dados.

```
public class SingletonContext
       {
           private static readonly
Lazy<ApplicationDbContext> instance =
                  new
Lazy<ApplicationDbContext>(() => new
ApplicationDbContext());

           public static
ApplicationDbContext Instance =>
instance.Value;
       }

public class SingletonContext
{
    private SingletonContext() { }
    private static Context ctx;
    public static Context GetInstance()
    {
        if (ctx == null)
```

```
            ctx = new Context();
        return ctx;
    }
}
```

Fonte: O autor (2025).

Fluxo de Operações CRUD com EF:

- Create: Add() ou AddAsync() + SaveChanges()
- Read: consultas com ToList(), FirstOrDefault()
- Update: modificar objeto carregado e salvar
- Delete: Remove() + SaveChanges()

Operações assíncronas como SaveChangesAsync() são usadas para manter a aplicação responsiva.

Um exemplo prático ilustrado no código 6 é a inserção de uma peça no banco de dados.

Código 6 – Inserção de uma peça no banco de dados.

```
using (var context =
SingletonContext.GetInstance())
{
    var newPart = new Part
    {
        Part_Number = "12345",
        Part_Name = "Peça A",
        Project = "Projeto Y"
    };

    PartDAO.adicionaPart(newPart);
    context.SaveChanges();
}
```

Fonte: O autor (2025).

Vantagens da Abordagem:

- Eficiência: Uso do Singleton evita múltiplas conexões.
- Manutenção: DAOs organizam e centralizam as operações.
- Abstração: EF permite foco na lógica de negócio, sem SQL direto.
- Responsividade: Métodos assíncronos melhoram desempenho da aplicação.

5 RESULTADOS

5.1 HARDWARE

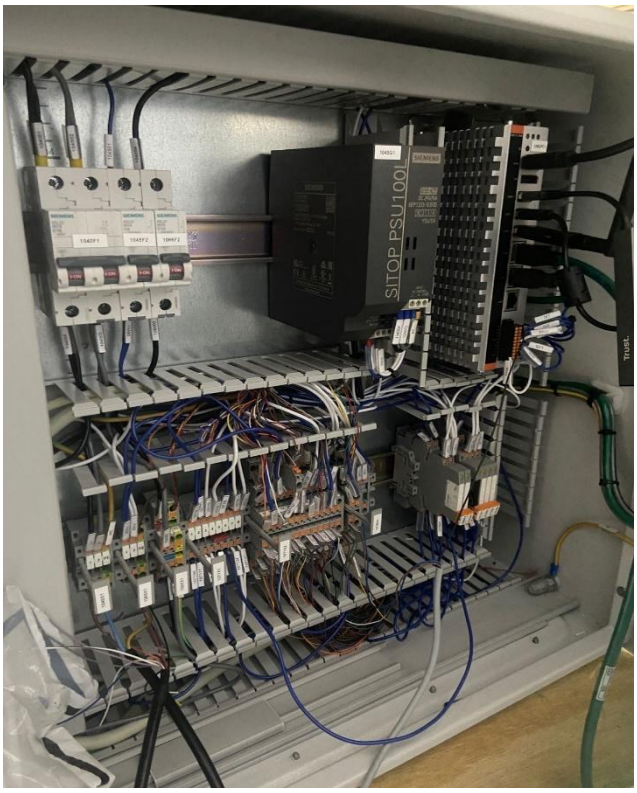
A estruturação do hardware foi realizada com base em normas de automação industrial, como a IEC 60204-1, garantindo segurança e eficiência. O painel elétrico utilizado é da marca Rittal, modelo de dimensões 50x50 cm, que acomoda todos os componentes de forma organizada e acessível. A alimentação do sistema é provida por uma fonte de 24V DC da Siemens, protegida por disjuntores individuais para os diferentes subsistemas: a fonte, o PC industrial e as saídas digitais.

Conforme ilustrado no Esquemático do Sistema (Figura 16), o coração do sistema é um PC industrial NODKA, equipado com 8 entradas e 8 saídas digitais. Para integrar o funcionamento da máquina de costura industrial, utilizou-se dois relés de interface da Phoenix Contact conectados às saídas digitais. Um dos relés habilita o funcionamento geral da máquina, enquanto o outro estava destinado a acionar a marcha, mas atualmente apenas o de habilitação está ativo. As entradas digitais captam sinais de um sensor que monitora os pulsos da máquina de costura e de um relé responsável pelo remate, permitindo o controle preciso de cada operação.

Os componentes de rede são integrados por interfaces específicas. A câmera de inspeção Snapp da Cognex opera via Ethernet, enquanto dispositivos como o leitor biométrico Control BioID, o leitor de código de barras Zebra, e o monitor touchscreen são conectados via USB. O monitor também utiliza uma interface HDMI para transmissão de vídeo. Além disso, bornes foram empregados para espelhamento das entradas e saídas dos conectores, simplificando a manutenção e ampliando a flexibilidade do sistema.

A disposição interna dos componentes no painel é apresentada na Figura 17, destacando o uso de bornes, disjuntores e trilhos para organizar os equipamentos de forma compacta e funcional. Este layout garante conformidade e organização, seguindo os esquemáticos elétricos para facilitar futuras intervenções técnicas.

Figura 17 - Painel Elétrico



FONTE: O autor (2024).

Finalizado a montagem de acordo com as normas técnicas especificadas anteriormente, foi feito os testes dos equipamentos criando telas auxiliares para teste de cada equipamento.

5.2 SOFTWARE

Conforme citado durante a progressão do trabalho, o software foi criado com o intuito de ser uma interface amigável para o usuário, com o objetivo de manipular a máquina controlando quando e como seria feito cada passo do processo. Portanto, o software foi desenvolvido na seguinte sequência:

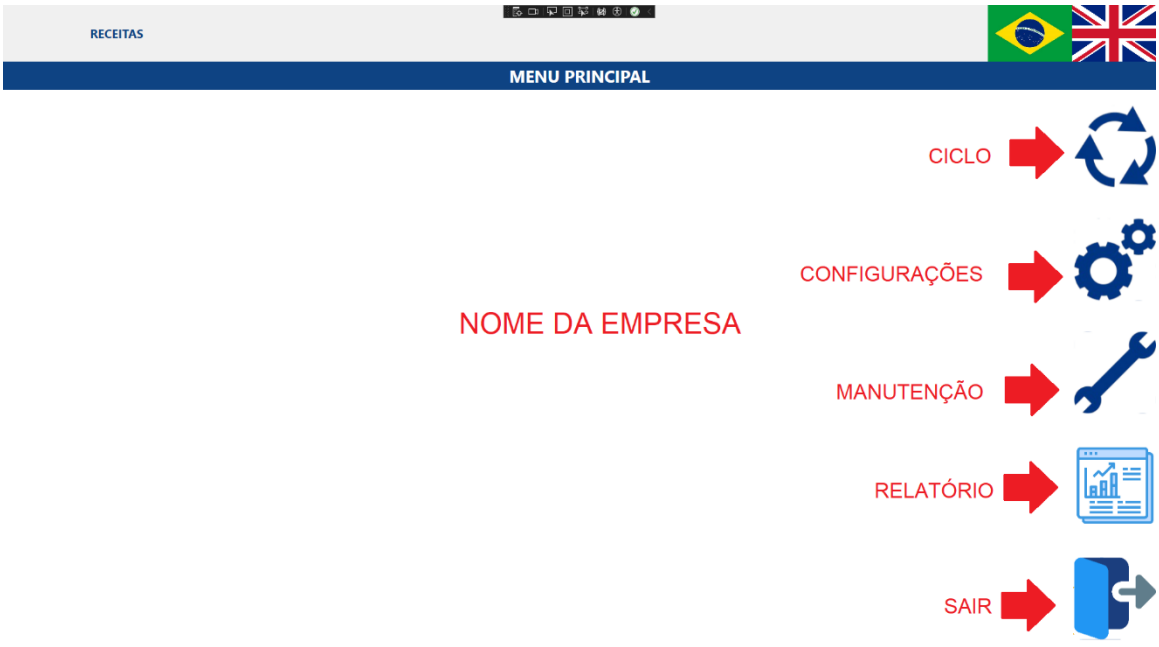
5.2.1 Tela Principal

A tela principal do software é responsável por fornecer acesso rápido a todas as principais funcionalidades do sistema. Ela inclui um menu de navegação com ícones e descrições para as seguintes seções:

- Ciclo: Permite iniciar e acompanhar o ciclo de produção.
- Configurações: Para ajustar as configurações gerais do sistema.
- Manutenção: Acesso às funcionalidades de manutenção da máquina. - Relatório: Geração de relatórios sobre o desempenho e a produção.
- Sair: Opção para sair do sistema de forma segura.

No centro da tela, há também o nome da empresa, garantindo a identidade visual do software.

Figura 18 - Principal



FONTE: O autor (2024).

5.2.2 Tela de Configurações

A tela de Configurações permite ao usuário ajustar diversas funcionalidades do sistema, garantindo que ele esteja configurado de acordo com as necessidades específicas de cada operação. Esta tela possui as seguintes seções:

- Receitas: Acesso ao gerenciamento de receitas do sistema.

- Usuários: Configuração dos usuários que podem acessar o sistema, incluindo permissões.

- Dados: Configurações relacionadas ao armazenamento e gerenciamento dos dados utilizados.

- Biometria: Ajustes para captura e utilização de biometria para acesso seguro ao sistema.

- Câmera: Configurações específicas para integrar e calibrar a câmera utilizada no processo.

Figura 19 - Configurações

FONTE: O autor (2024).

5.2.3 Tela de Ciclo

A tela de Ciclo é uma das principais do sistema, pois é nela que o operador acompanha e controla a produção em tempo real. Ela exibe informações sobre o estado atual da produção e permite ao usuário realizar ações como iniciar ou cancelar o ciclo.

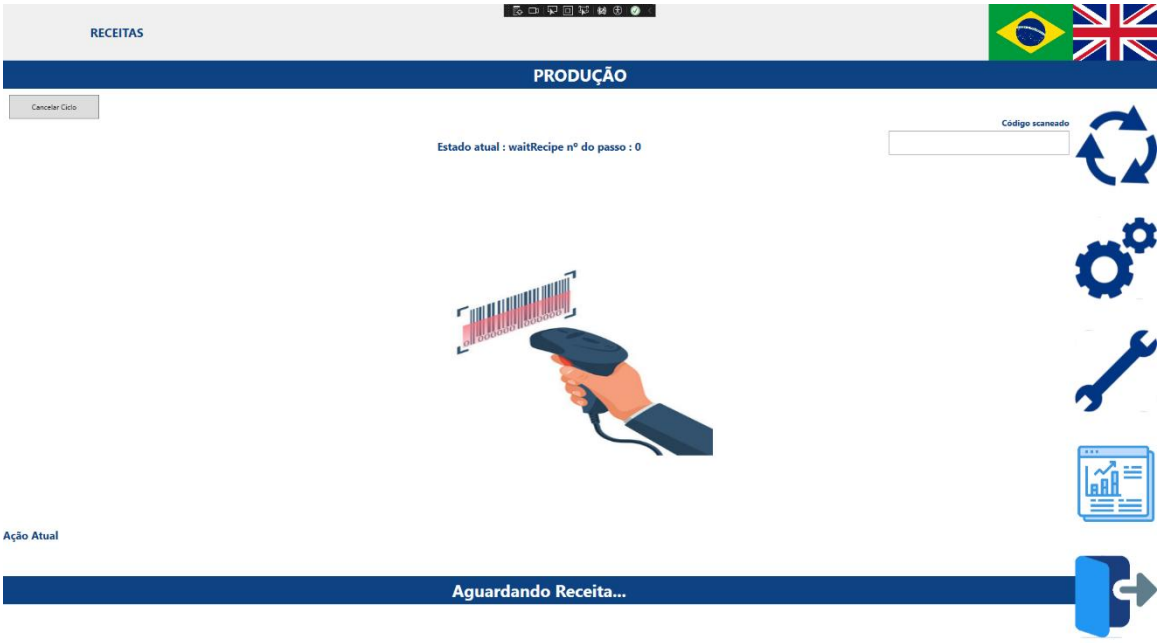
- Estado Atual: Mostra o estado atual da máquina, incluindo informações sobre o passo do ciclo em execução.

- Código Scaneado: Permite inserir ou verificar o código de barras escaneado pela leitora de código.

- Cancelar Ciclo: Um botão para interromper o ciclo de produção a qualquer momento.

Esta tela garante uma interface clara e objetiva para que o operador possa monitorar o progresso e tomar ações rapidamente quando necessário.

Figura 20 - Ciclo



FONTE: O autor (2024).

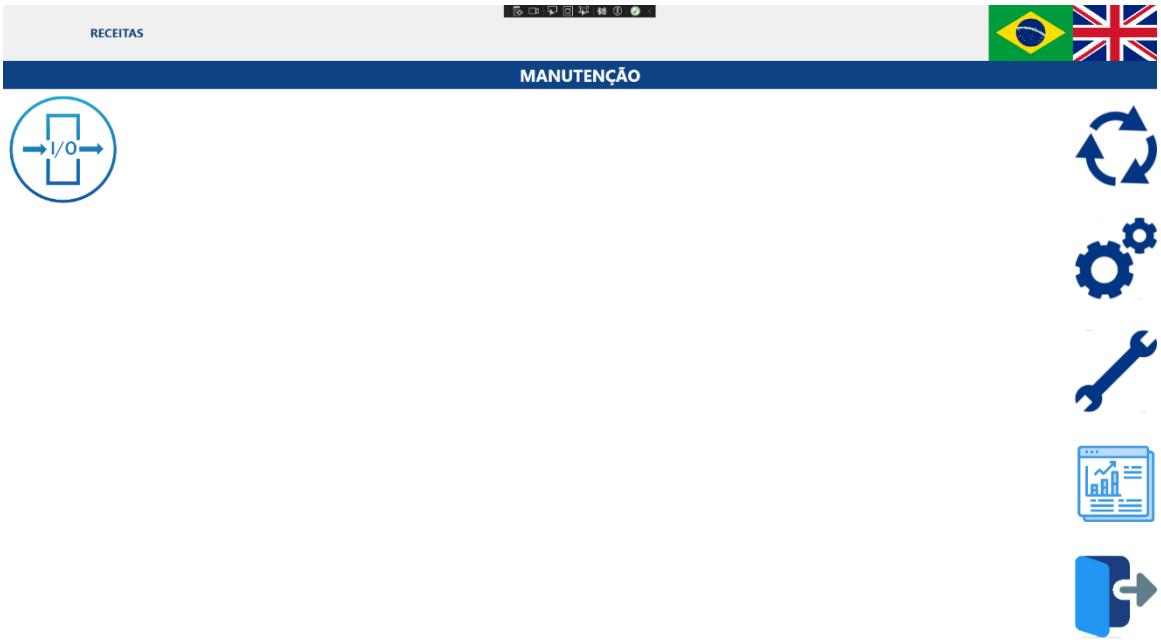
5.2.4 Tela de Manutenção

A tela de Manutenção foi criada para permitir que técnicos e operadores realizem testes e verificações nos dispositivos que compõem o sistema, garantindo que a máquina esteja funcionando corretamente.

- Entradas e Saídas (I/O): A principal funcionalidade desta tela é testar as entradas e saídas digitais da máquina. Isso garante que todos os componentes estejam funcionando como esperado e ajuda a identificar e solucionar problemas rapidamente.

Esta tela oferece uma interface simples e direta, possibilitando que o técnico realize as manutenções necessárias de maneira eficiente.

Figura 21 - Manutenção



FONTE: O autor (2024).

5.2.5 Tela de Entradas e Saídas (I/Os)

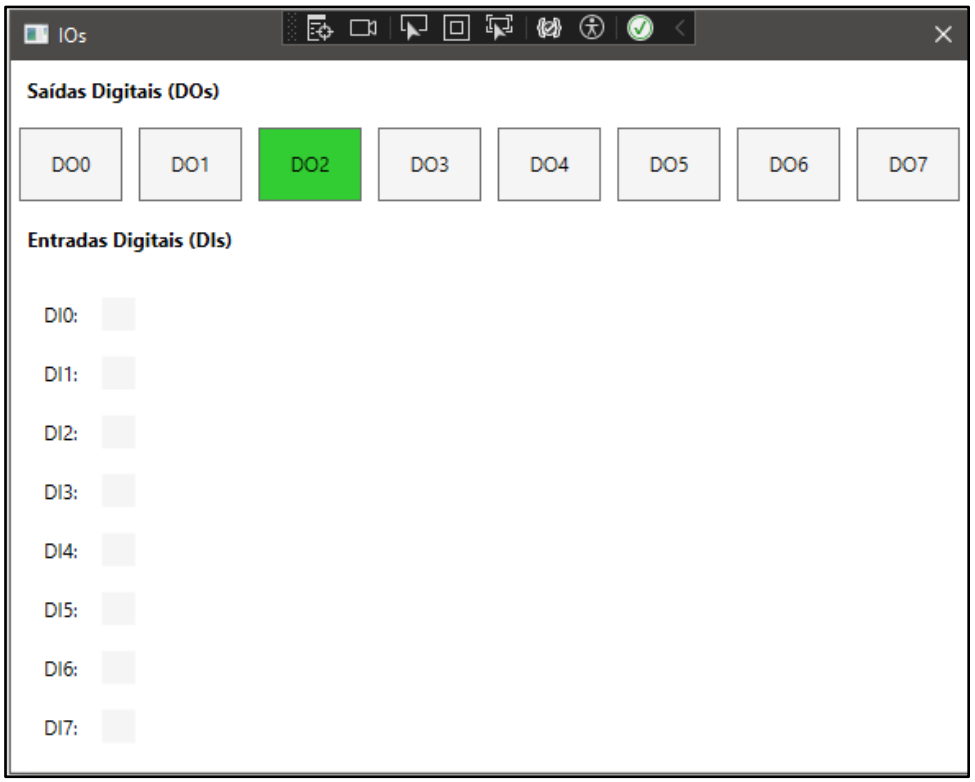
A tela de Entradas e Saídas (I/Os) é acessada através da tela de Manutenção e permite ao técnico verificar o funcionamento das saídas e entradas digitais do sistema.

- Saídas Digitais (DOs): Esta seção mostra o estado atual das saídas digitais. Cada botão representa uma saída (D00 a D07) e pode ser utilizado para ligar ou desligar cada uma das saídas, conforme necessário. As saídas que estão ativas são destacadas em verde.

- Entradas Digitais (DIs): Abaixo das saídas digitais, é apresentada a lista de entradas digitais (DI0 a DI7). Esses indicadores mostram o estado atual das entradas, ajudando a identificar sinais que estão sendo recebidos pelo sistema.

Esta tela é essencial para testar e diagnosticar problemas de hardware, garantindo que todas as entradas e saídas do sistema estejam operando corretamente.

Figura 22 - Entradas e Saídas (IO's)



FONTE: O autor (2024).

5.2.6 Tela de Gerenciamento de Receitas

A tela de Gerenciamento de Receitas é acessada através do primeiro botão da tela de Configuração. Ela permite ao usuário criar, editar, deletar ou copiar receitas utilizadas no processo de produção.

- Nome e Barcode: Campos para inserir ou visualizar o nome da receita e o código de barras associado.

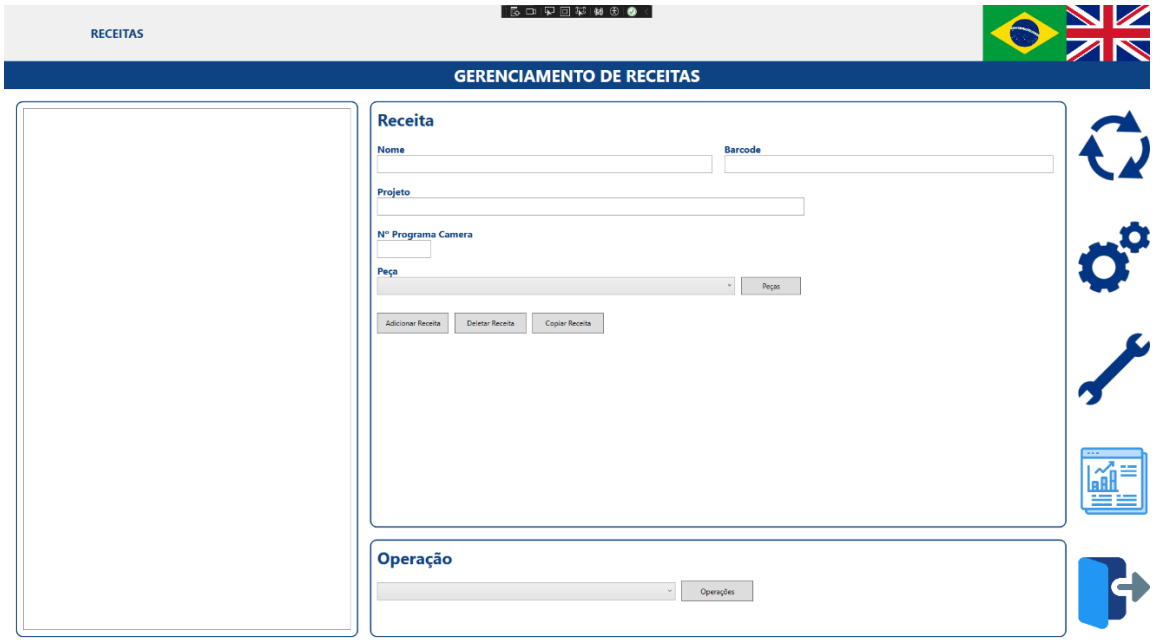
- Projeto e Nº Programa da Câmera: Campos para associar a receita a um projeto específico e ao programa de câmera correspondente.

- Peça: Permite a seleção da peça relacionada à receita.

- Botões de Ação: A tela possui botões para adicionar, deletar ou copiar uma receita, facilitando o gerenciamento e manipulação de receitas.

Essa tela é essencial para garantir que as receitas estejam devidamente cadastradas e configuradas, proporcionando controle sobre o que será utilizado em cada ciclo de produção.

Figura 23 - Receita



FONTE: O autor (2024).

5.2.7 Tela de Operações da Receita

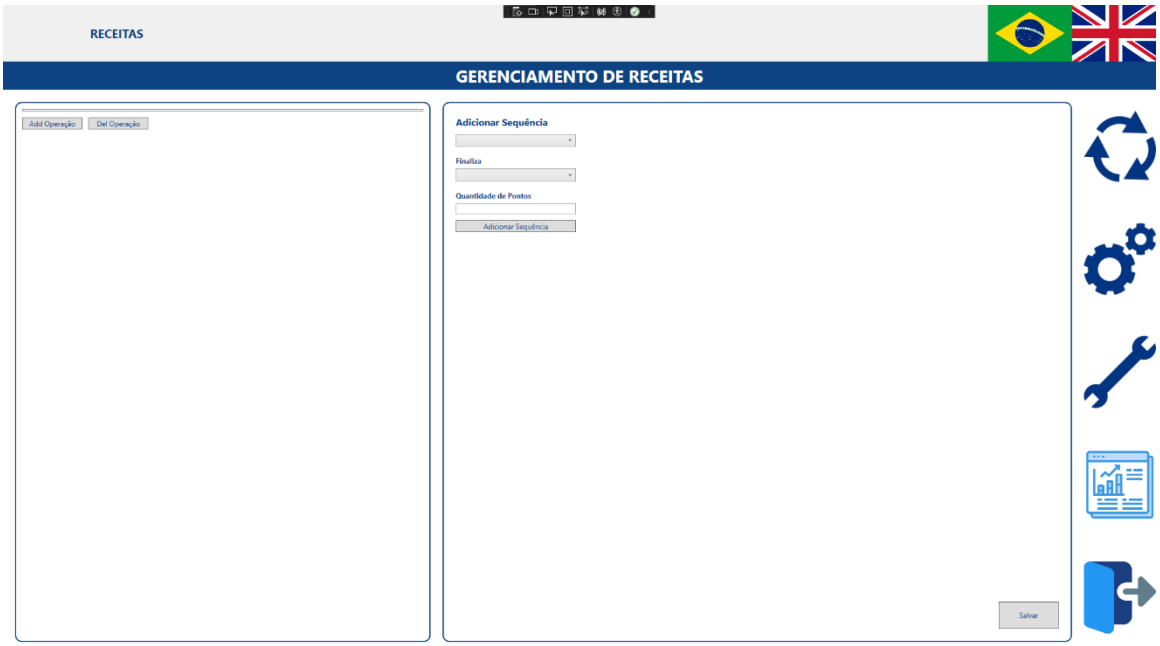
A tela de Operações é acessada através da tela de Gerenciamento de Receitas e é responsável por definir as etapas específicas que compõem cada receita. Nesta tela, o usuário pode adicionar ou remover operações, bem como criar sequências de operação.

- Lista de Operações: À esquerda, o usuário pode visualizar as operações já cadastradas. Existem botões para adicionar ('Add Operação') ou remover ('Del Operação') operações da lista.

- Adicionar Sequência: À direita, há a opção de adicionar uma nova sequência de operações, onde o usuário pode definir campos como 'Finaliza' e 'Quantidade de Pontos'. O botão 'Adicionar Sequência' salva as informações inseridas.

Esta tela é fundamental para configurar todas as etapas do processo de produção de uma receita, garantindo que as operações estejam claramente definidas e organizadas de acordo com a necessidade da produção.

Figura 24 - Operações



FONTE: O autor (2024).

5.2.8 Tela de Gerenciamento de Peças

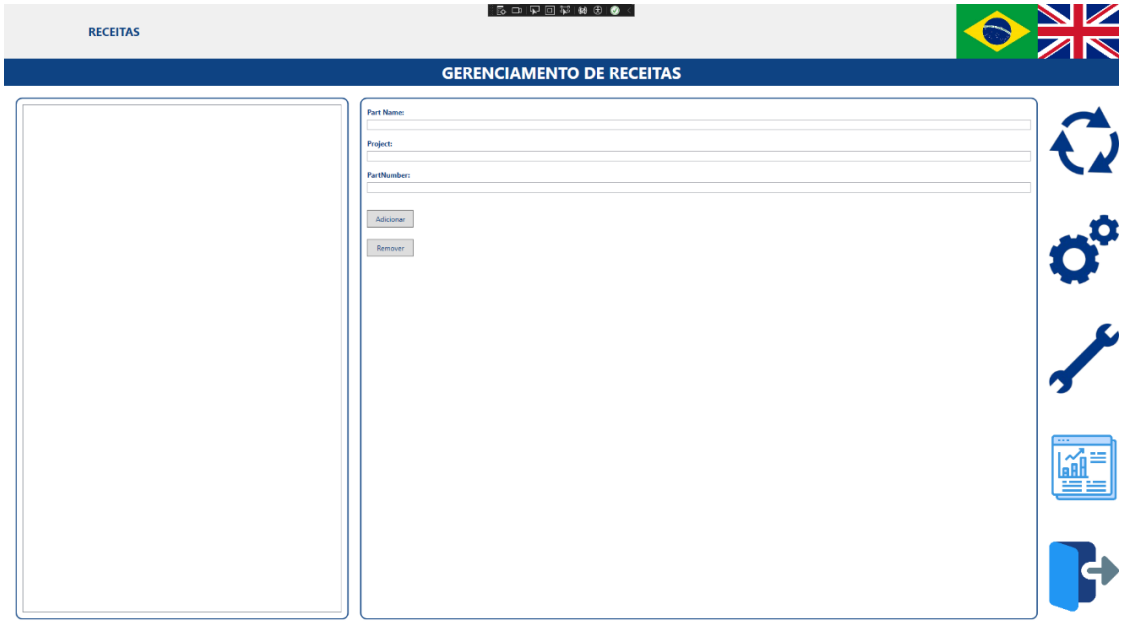
A tela de Gerenciamento de Peças é acessada através da tela de Gerenciamento de Receitas e permite ao usuário criar e gerenciar as peças que serão utilizadas nas receitas.

- Part Name, Project e Part Number: Esses campos permitem ao usuário inserir ou editar o nome da peça, associar um projeto, e definir um número de identificação para a peça.

- Botões de Ação: A tela possui botões 'Adicionar' e 'Remover', permitindo ao usuário incluir uma nova peça no sistema ou remover uma peça já cadastrada.

Esta tela é fundamental para garantir que todas as peças que serão utilizadas na produção estejam devidamente cadastradas e associadas às receitas necessárias, proporcionando um controle mais eficaz e organizado das peças durante o processo produtivo.

Figura 25 - Peças



FONTE: O autor (2025).

5.2.9 Tela de Configuração da Câmera

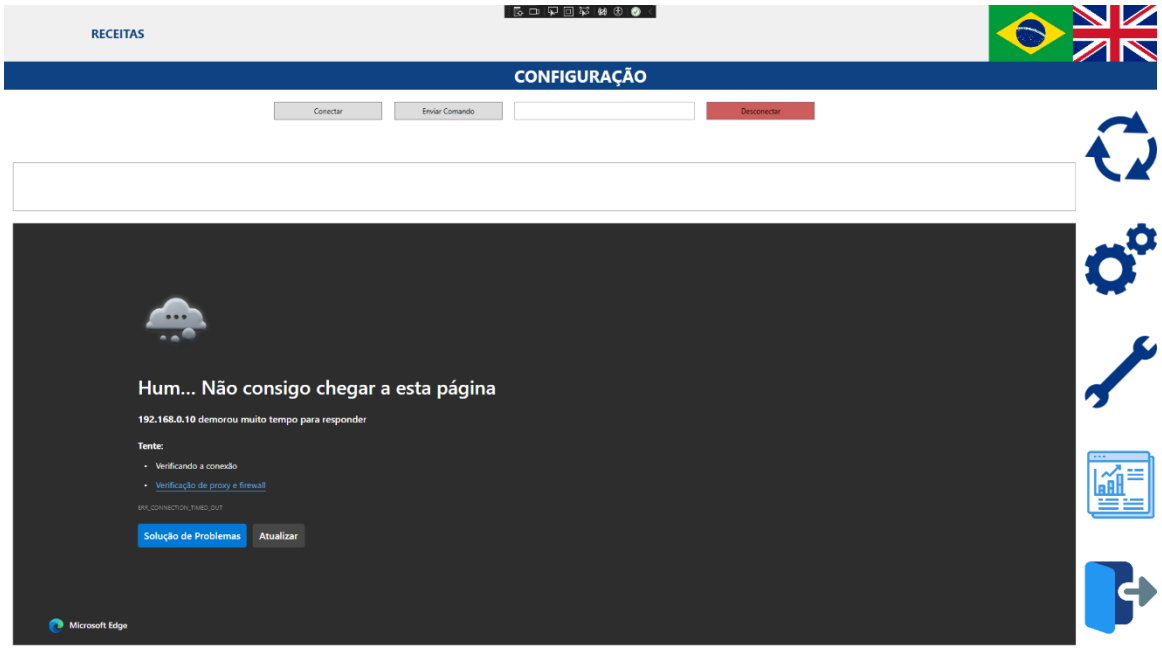
A tela de Configuração da Câmera é acessada através da tela de Configurações e permite ao usuário gerenciar a conexão e a configuração da câmera utilizada no processo produtivo.

- Botões de Conexão: A tela possui botões para conectar, enviar comando e desconectar a câmera, permitindo ao usuário estabelecer e finalizar a comunicação com o dispositivo.

- Status de Conexão: A parte inferior da tela exibe informações sobre o status atual da conexão. Em casos de falha de conexão, como mostrado na imagem, são exibidas mensagens de erro para auxiliar o usuário a identificar e resolver o problema.

Esta tela é crucial para garantir que a câmera esteja corretamente conectada e operando de forma adequada, proporcionando uma visão clara do processo e possibilitando ajustes sempre que necessário.

Figura 26 - Configuração Câmera



FONTE: O autor (2025).

5.2.10 Tela de Gerenciamento de Usuários

A tela de Gerenciamento de Usuários é acessada através da tela de Configurações e permite ao administrador gerenciar os usuários que têm acesso ao sistema.

- Campos de Cadastro: Nesta tela, o usuário pode preencher os seguintes campos:

Nome: Nome do usuário a ser cadastrado.

Número: Número de identificação do usuário.

- Nível de Acesso: Define o nível de permissão do usuário (ex.: operador, supervisor, administrador).

Senha: Define a senha de acesso do usuário.

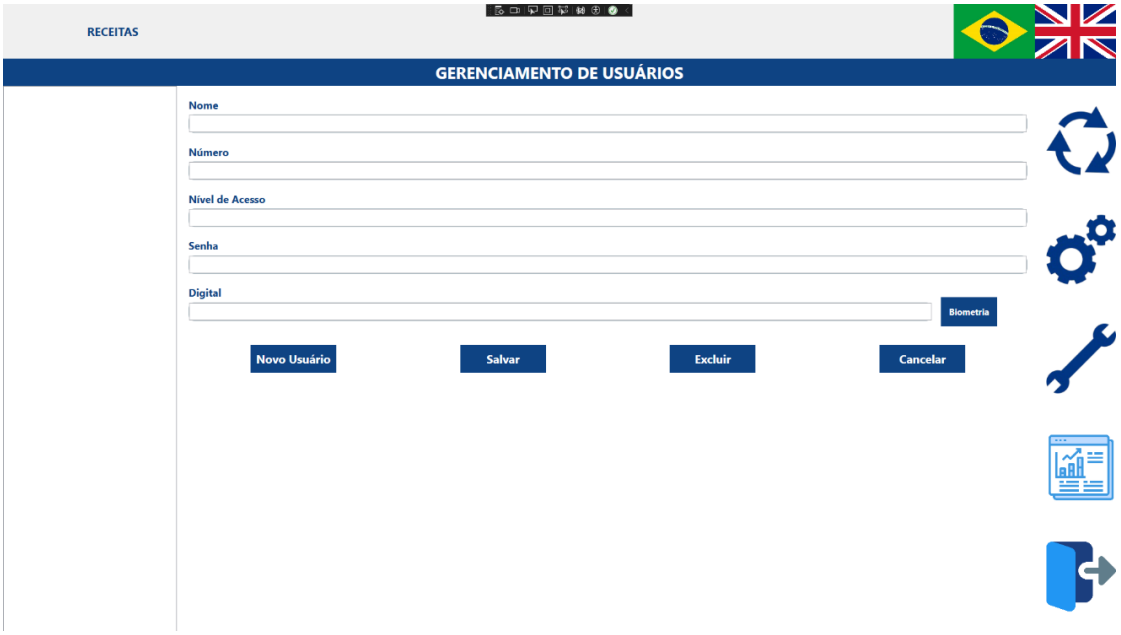
Digital: Permite o cadastro da biometria do usuário para acesso seguro.

Botões de Ação: A tela possui botões para:

- Novo Usuário: Criar um novo cadastro.
- Salvar: Salvar alterações no cadastro.
- Excluir: Excluir um usuário existente.
- Cancelar: Cancelar qualquer alteração feita.

Esta tela é essencial para garantir a segurança do sistema, assegurando que apenas pessoas autorizadas possam acessar determinadas funcionalidades.

Figura 27 - Usuários



FONTE: O autor (2025).

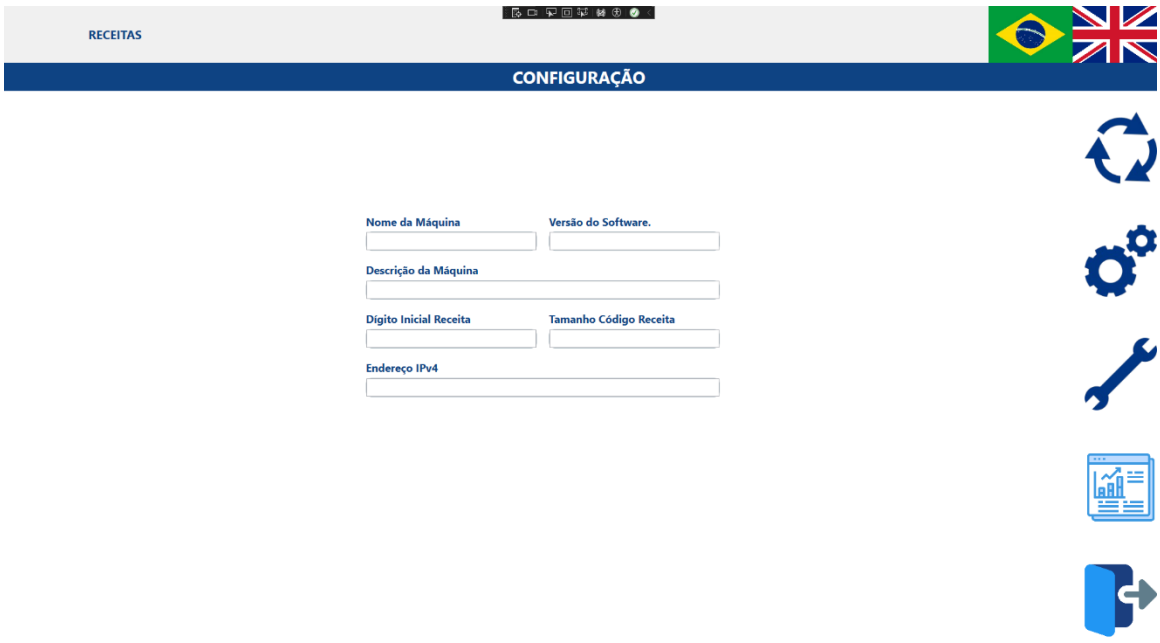
5.2.11 Tela de Parâmetros de Máquina

A tela de Parâmetros de Máquina permite configurar informações essenciais da máquina e do sistema. Esta tela é acessada através da opção de Configurações e contém os seguintes campos:

- Nome da Máquina: Define o nome da máquina utilizada no processo.
- Versão do Software: Informa a versão do software atualmente em uso.
- Descrição da Máquina: Permite adicionar uma descrição detalhada da máquina.
- Dígito Inicial da Receita e Tamanho do Código da Receita: Configura os padrões para os códigos das receitas, garantindo a consistência do sistema.
- Endereço IPv4: Define o endereço IP para conexão da máquina, essencial para comunicação em rede.

Esta tela é fundamental para assegurar que os parâmetros de funcionamento da máquina estejam corretos, proporcionando maior controle e integração do sistema.

Figura 28 - Parâmetros



FONTE: O autor (2025).

5.2.12 Tela de Relatórios

A tela de Relatórios é acessada através do segundo botão de baixo para cima no menu lateral direito. Ela permite ao usuário gerar relatórios das costuras realizadas em uma máquina específica, sendo possível filtrar por data ou gerar um relatório completo.

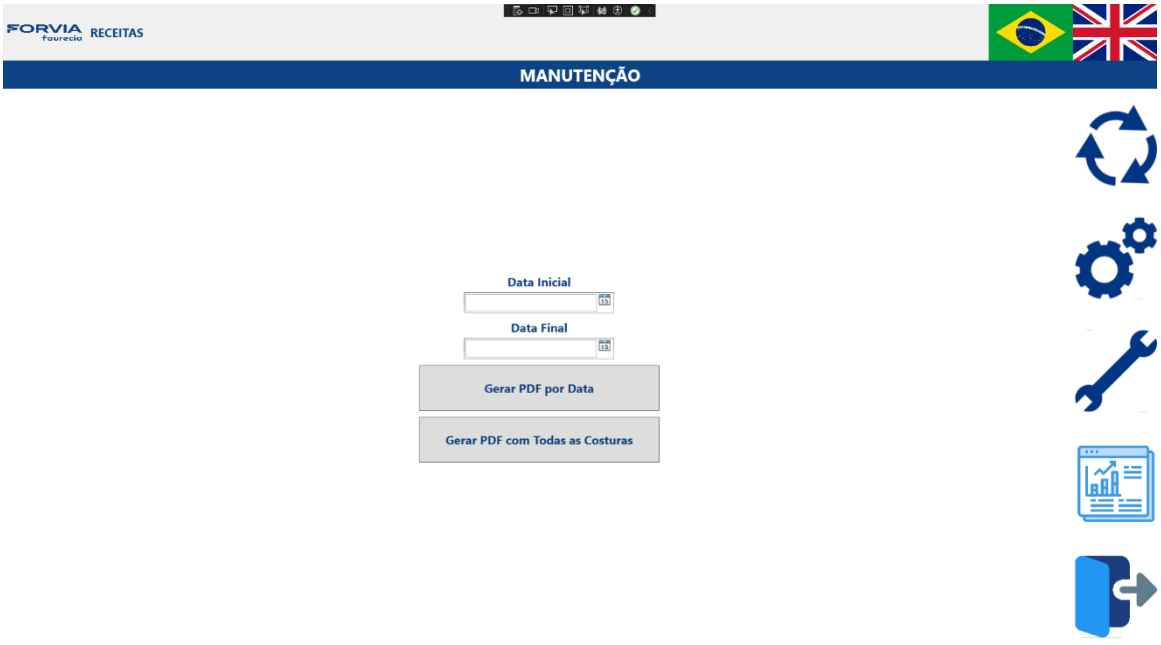
- Data Inicial e Data Final: Esses campos permitem definir um período específico para a geração do relatório, facilitando a visualização de dados específicos.

- Gerar PDF por Data: Gera um relatório em PDF contendo as costuras realizadas no período especificado.

- Gerar PDF com Todas as Costuras: Gera um relatório em PDF contendo todas as costuras realizadas, independentemente da data.

Esta tela é essencial para manter o histórico de produção e garantir a rastreabilidade das operações realizadas na máquina, proporcionando um controle detalhado das costuras executadas.

Figura 29 - Relatório



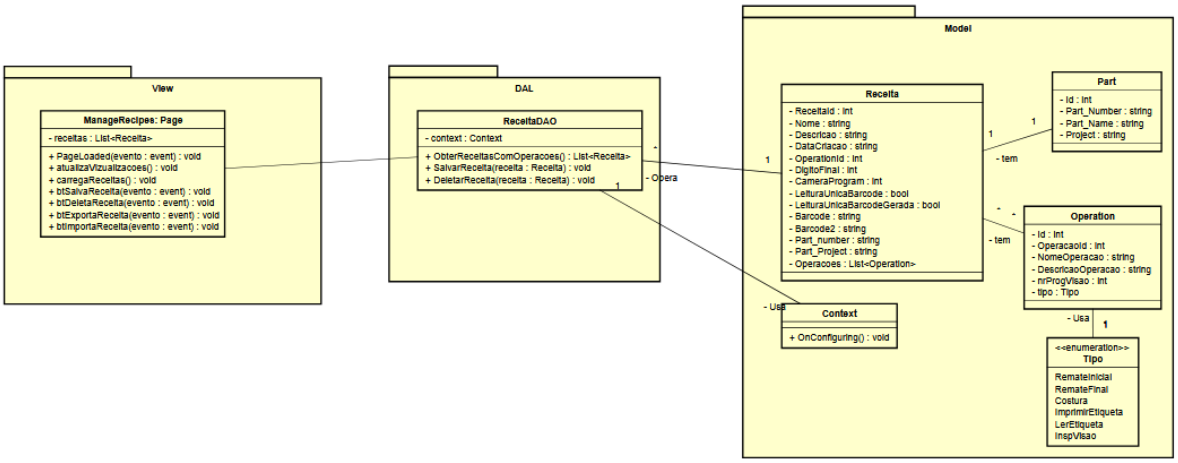
FONTE: O autor (2025).

5.3 ARMAZENAMENTO DE INFORMAÇÕES NO BANCO DE DADOS

O armazenamento das informações no banco de dados do software foi projetado para garantir organização, eficiência e facilidade de manutenção. Para isso, foi utilizado o **Entity Framework Core (EF Core)** como ferramenta principal de mapeamento objeto-relacional (ORM), permitindo que as classes em C# sejam diretamente associadas às tabelas do banco de dados relacional.

A **Figura 30** apresenta o diagrama de classes responsável por ilustrar o fluxo completo de persistência dos dados, desde a entrada de informações na interface do usuário até a gravação no banco de dados.

Figura 30 - Diagrama de classe receita



FONTE: O autor (2025).

5.3.1 Estrutura e Funcionamento das Classes

O projeto segue uma arquitetura dividida em três principais camadas:

Model**:** Nesta camada estão localizadas as classes que representam as entidades do sistema. A classe Receita, por exemplo, representa uma receita de produção com todos os atributos associados, como Nome, DataCriacao, DigitoInicial, OperationID, CameraProgram, Barcodes, entre outros. Além disso, possui relacionamentos com outras entidades, como Part (Peça) e Operation (Operação). Essas relações são expressas através de propriedades de navegação e lista de objetos, facilitando a manipulação das dependências entre os dados.

DAL (Data Access Layer)**:** Representada pela classe ReceitaDAO, essa camada é responsável por realizar a mediação entre as entidades e o banco de dados. Nela estão os métodos para salvar (SalvarReceita), obter (ObterReceitasComOperacoes) e excluir (DeleteReceitaReceita) as informações no banco de dados. Essa classe utiliza uma instância do contexto do EF Core (classe Context) para realizar as operações de persistência.

View (Interface de Usuário)**:** Representada pela classe ManageReceitasPage, essa camada permite ao usuário visualizar e interagir com as receitas. Ao carregar a página ou interagir com os botões de salvar, deletar ou importar receita, os eventos chamam métodos da ReceitaDAO, que acessam ou modificam os dados no banco. A lista de receitas carregada (receitas) é populada a partir da DAL e refletida na interface gráfica para o usuário.

5.3.2 Gerenciamento do Contexto de Banco de Dados

Um ponto importante no projeto é o uso de um padrão “Singleton” para a classe de contexto (Context), garantindo que todas as operações com o banco de dados compartilhem a mesma instância. Isso evita múltiplas conexões desnecessárias e mantém a integridade e consistência dos dados.

5.3.3 Resumo do Fluxo de Dados

1. O usuário interage com a interface (ManageReceitas) para inserir ou modificar uma receita.
2. Os dados são enviados para a camada de acesso (ReceitaDAO), que, por meio do contexto compartilhado, realiza a operação desejada (salvar, buscar ou deletar).
3. O EF Core realiza o mapeamento da entidade Receita (e suas dependências) para as tabelas do banco de dados.
4. Os dados são persistidos ou lidos de forma transparente, com o EF Core gerenciando as queries necessárias.

Esse fluxo garante que as informações inseridas na interface do sistema sejam corretamente estruturadas e armazenadas no banco, promovendo organização e facilitando futuras consultas ou manutenções.

5.4 TECNOLOGIAS EXISTENTES E POTENCIAL MERCADOLÓGICO

O sistema desenvolvido para o controle e automação de máquinas de costura industrial representa uma solução tecnológica que se destaca por sua flexibilidade, usabilidade e integração. Em um mercado que busca cada vez mais eficiência, qualidade e rastreabilidade nos processos produtivos, o uso de tecnologias como o Entity Framework Core, WPF, e a comunicação com equipamentos industriais como câmeras e sistemas de controle de produção, coloca o produto em uma posição competitiva e promissora.

5.4.1 Tecnologias Utilizadas e Comparação com o Mercado Atual

No contexto atual da indústria 4.0, a automação e a digitalização dos processos produtivos são essenciais para aumentar a competitividade das empresas. A solução proposta utiliza tecnologias modernas, como:

Entity Framework Core (EF Core): Facilita o acesso ao banco de dados, proporcionando maior flexibilidade no armazenamento e consulta de dados. Comparado a sistemas que ainda utilizam acesso direto ao banco via SQL puro, o EF Core permite maior segurança, legibilidade e manutenção mais fácil do código.

WPF (Windows Presentation Foundation): Utilizado para desenvolver a interface gráfica, o WPF oferece uma experiência de usuário altamente personalizável, essencial para o ambiente industrial. Muitas empresas ainda utilizam interfaces limitadas e de difícil interação, o que torna o WPF um diferencial em termos de usabilidade.

Padrão de Projeto Singleton: Implementado para garantir a consistência das conexões ao banco de dados, proporcionando um fluxo seguro e eficiente na comunicação com o banco.

Integração com Equipamentos Industriais: O sistema foi desenvolvido para se comunicar com câmeras industriais Cognex, além de sensores e outros dispositivos, permitindo um controle abrangente do processo de costura e garantindo uma qualidade constante no produto final.

5.4.2 Potencial Mercadológico

O mercado de automação industrial está em constante crescimento, impulsionado pela busca por eficiência operacional, redução de custos e aumento da produtividade. Dentro desse contexto, o sistema desenvolvido apresenta um grande potencial mercadológico por várias razões:

Automação e Eficiência: O sistema oferece uma automação completa do processo de costura, permitindo que as empresas reduzam a dependência de intervenções manuais, o que leva a uma diminuição dos erros e aumento da qualidade do produto final. Isso é extremamente atrativo para indústrias que desejam aumentar sua competitividade em um mercado cada vez mais exigente.

Rastreabilidade e Qualidade: Com o uso de câmeras industriais e a geração de relatórios completos sobre as produções, o sistema proporciona uma rastreabilidade precisa de cada peça produzida. A rastreabilidade é um fator cada vez mais relevante em indústrias que necessitam comprovar a qualidade dos seus produtos, principalmente nas áreas automotiva e de moda, onde o controle da qualidade é essencial.

Flexibilidade e Customização: O sistema permite a criação e gerenciamento de receitas de costura e operações de forma dinâmica, possibilitando a adaptação do processo às necessidades específicas de cada cliente. Essa flexibilidade é um diferencial competitivo importante em um mercado que busca soluções sob medida para demandas variadas.

Adoção da Indústria 4.0: O conceito de Indústria 4.0 tem sido cada vez mais adotado por empresas que desejam transformar seus processos fabris em ambientes altamente tecnológicos. O sistema proposto está alinhado com essa visão, proporcionando conectividade e coleta de dados em tempo real, o que possibilita a tomada de decisão mais rápida e baseada em dados concretos.

Potencial de Escalabilidade: O software desenvolvido é escalável e pode ser facilmente adaptado para diferentes tipos de máquinas e linhas de produção. Esse potencial de expansão o torna interessante para empresas que desejam integrar diferentes etapas da produção em um único sistema de controle, melhorando a integração e otimizando a logística interna.

6 CONCLUSÃO

Ao longo do desenvolvimento deste projeto, demonstrou-se, através de experimentos, que a utilização de um sistema automatizado para controle e armazenamento de dados de uma máquina de costura industrial se faz eficiente, trazendo benefícios claros para a otimização do processo produtivo e a rastreabilidade de cada etapa.

Realizados testes no protótipo do sistema e comparados os valores obtidos das operações, os resultados indicaram que a automação proporcionou uma melhoria significativa na precisão e no controle das costuras, reduzindo a incidência de erros humanos. Concluiu-se, pelos resultados obtidos, que a integração dos módulos de hardware foi realizada com sucesso, alcançando os objetivos inicialmente propostos.

Focando no aprimoramento do hardware, implementou-se uma fonte de alimentação dedicada e o controle de entradas e saídas digitais para garantir uma maior autonomia e independência do sistema, eliminando a necessidade de interfaces complexas para a alimentação e comunicação dos dispositivos. Técnicas de otimização foram aplicadas para reduzir o consumo energético e aumentar a eficiência do sistema, como a escolha de componentes de baixo consumo e a configuração adequada do PC industrial Nodka.

Quanto ao software, seguiu-se o conceito das telas principais - "Receitas", "Ciclo de Produção" e "Configurações", garantindo que todas as funcionalidades fossem organizadas de forma lógica e acessível. Constatou-se, através de testes, que as telas do sistema se comunicam adequadamente, permitindo ao operador uma interação fluida e intuitiva durante o uso. Nota-se, também, que as funcionalidades relacionadas ao banco de dados se mostraram adequadas para a aplicação, uma vez que todas as operações estipuladas, como criação de receitas, salvamento de dados de costura e geração de relatórios, foram realizadas sem erros. A integração com o Entity Framework garantiu um acesso rápido e seguro aos dados armazenados.

A comunicação com os periféricos, apresentou alguns desafios relacionados à recepção de dados, acessos as entradas e saídas entre outros, mas foi possível validar a comunicação entre os dispositivos e o sistema através da interface implementada, tornando a situação um problema de ajuste no algoritmo. Assim, avaliando-se as necessidades iniciais propostas pela problemática do trabalho, conclui-se que o sistema desenvolvido atende de maneira satisfatória às expectativas,

fornecendo ao operador controle total sobre o processo de costura e garantindo a rastreabilidade dos dados. Além disso, a possibilidade de gerar relatórios detalhados e armazenar informações para futuras análises contribui diretamente para a eficiência do processo. Com isso, é possível destacar, ao final do trabalho, o aprendizado sobre a construção de um sistema completo de automação, desde a integração de hardware até o desenvolvimento de software com foco na usabilidade. Aprendeu-se a estruturar um sistema modular e escalável, além de garantir a comunicação entre diferentes dispositivos para o armazenamento seguro dos dados.

6.1 TRABALHOS FUTUROS

Com relação ao sistema desenvolvido, estão previstas diversas otimizações para torná-lo ainda mais robusto e competitivo. Uma das principais melhorias é a compactação do painel de controle, visando reduzir o custo de fabricação, simplificar a instalação e facilitar a manutenção. Ao otimizar o design do hardware, espera-se uma significativa redução no consumo de energia, além de menor espaço físico necessário, o que torna o sistema mais atraente para pequenos e médios fabricantes do setor têxtil.

Além disso, o design da interface do usuário será refinado para incluir fontes personalizadas, ícones modernos e navegação otimizada, melhorando a experiência dos operadores e reduzindo o tempo de treinamento. Esses ajustes aumentam a produtividade e reduzem a margem de erro, criando uma interface mais intuitiva e agradável.

Para ampliar o alcance comercial, o sistema está sendo desenvolvido com suporte a múltiplos idiomas, incluindo a internacionalização de textos, formatos de data, unidades de medida e mensagens de status. Essa abordagem permitirá a comercialização do sistema em diversos mercados, atendendo às demandas de clientes globais e facilitando a adaptação a novos ambientes de produção.

Novas funcionalidades estão sendo implementadas para permitir a integração com sensores adicionais, como medidores de temperatura, sensores de umidade, sistemas de tensão de linha, monitores de velocidade de corte e sensores de pressão. Essa integração possibilita o monitoramento em tempo real de variáveis críticas do processo de costura, melhorando a qualidade do produto final e reduzindo o desperdício de materiais.

Além disso, estão sendo avaliadas soluções para habilitar o monitoramento remoto do sistema, permitindo que operadores e gestores acompanhem o processo de costura a partir de dispositivos móveis ou painéis web. Isso será possível através do uso de protocolos de comunicação industrial modernos, como Wi-Fi, Ethernet/IP ou Modbus TCP, garantindo alta confiabilidade, baixa latência e maior segurança nas trocas de dados.

REFERÊNCIAS

PROGRESSIVE AUTOMATION. Operação de máquina, 2024.

DIO**.** Banco de dados relacional: fundamentos e aplicações. *DIO - Digital Innovation One*, [2023]. Disponível em: https://www.dio.me/articles/banco-de-dados-relacional-fundamentos-e-aplicacoes. Acesso em: 10 nov. 2024.

FACEBOOK. [Fotografia comparando SQL e NoSQL]. Facebook, 2025. Disponível em:https://www.facebook.com/photo/?fbid=1468736426942794&set=a.9919472312883. Acesso em: 8 jul. 2025.

PHOENIX CONTACT. Módulo de relé PLC-RSC/24DC/21 – ficha técnica. Phoenix Contact, 2025. Disponível em: https://www.phoenixcontact.com/en-de/products/relay-module-plc-rsc-24dc-21-2966171?type=pdf. Acesso em: 8 jul. 2025.

BROWN, Marvin. *Power supply cookbook*. 2. ed. Oxford: Newnes, 2001.
COOPER, Grace Rogers. *The sewing machine: a technical history*. 2. ed. New York: McGraw-Hill, 1982.

SILBERSCHATZ, Abraham; KORTH, Henry F.; SUDARSHAN, S. *Database System Concepts*. 6. ed. New York: McGraw-Hill, 2010. ISBN 0-07-352332-1.

DATE, C. J. *An introduction to database systems*. 8. ed. Boston: Pearson, 2004.
ELMASRI, Ramez; NAVATHE, Shamkant B. *Fundamentals of database systems*. 7. ed. Boston: Pearson, 2015.

GAMMA, Erich; HELM, Richard; JOHNSON, Ralph; VLISSIDES, John. *Design patterns: elements of reusable object-oriented software*. Reading: Addison-Wesley, 1995.

HOROWITZ, Paul; HILL, Winfield. *The art of electronics*. 3. ed. Cambridge: Cambridge University Press, 2015.

MOHAN, Ned; UNDELAND, Tore M.; ROBBINS, William P. *Power electronics: converters, applications, and design*. 3. ed. Hoboken: John Wiley & Sons, 2003.

RASHID, Muhammad H. *Power electronics: circuits, devices, and applications*. 4. ed. Upper Saddle River: Pearson, 2013.

SILBERSCHATZ, Abraham; KORTH, Henry F.; SUDARSHAN, S. *Database system concepts*. 6. ed. New York: McGraw-Hill, 2010.

UDALL, J. E.; COOMBS, C. F. *Printed circuits handbook*. 6. ed. New York: McGraw-Hill, 2001.

COFFIN, Judith G. Credit, consumption, and images of women's desires: selling the sewing machine in late nineteenth-century France. *French Historical Studies*, v. 17, n. 4, p. 749-783, 1994. Disponível em: <https://www.jstor.org/stable/286691>. Acesso em: 08/10/2024.

ADAFRUIT. *USB and resistor configuration instructions*. 2013. Acesso em: 13/10/2024.

BRANDON, Ruth. *A capitalist romance: Singer and the sewing machine*. Philadelphia: Lippincott, 1977.

COOPER, G. R. *The sewing machine: its invention and development*. Smithsonian Institution Press, 1976.

FACEBOOK. Foto sobre a comparação de bancos de dados relacional vs não relacional. Acesso em: 24/10/2024.

MICHAEL JORDAN. Citação de motivação sobre falhas e sucesso. Acesso em: 03/10/2024.
MICROSOFT. *XAML overview para WPF e comparação com HTML*. Disponível em: https://learn.microsoft.com/pt-br/dotnet/desktop/wpf/xaml/?view=netdesktop-9.0. Acesso em: [29/10/2024].

MICROSOFT. *Styles and templates in XAML: detalhes sobre estilização e customização em XAML*. Disponível em: https://learn.microsoft.com/pt-br/dotnet/desktop/wpf/xaml/?view=netdesktop-9.0. Acesso em: [05/11/2024].

MOZILLA. *HTML: definição e detalhes*. Disponível em: [link se houver]. Acesso em: 29/10/2024.
W3C. *HTML living standard: padrões e evolução do HTML*. Acesso em: 14/11/2024.

SHINGO, Shigeo. *Poka-Yoke: Improving Product Quality by Preventing Defects*. Nikkan Kogyo Shimbun, 1987.
WIKIPÉDIA. *Poka-yoke*. Disponível em: https://en.wikipedia.org/wiki/Poka-yoke. Acesso em: 08 jul. 2025.

GOSKILLS. *Poka yoke definition: What is poka yoke?*. Disponível em: https://www.goskills.com/Lean-Six-Sigma/Resources/Poka-yoke-definition. Acesso em: 08 jul. 2025.

6SIGMA.US. *Poka-Yoke & Six Sigma*. Disponível em: https://www.6sigma.us/lean-tools/poka-yoke-six-sigma. Acesso em: 08 jul. 2025.

BLOGS.MTU.EDU. *Poka-yoke: Mistake-proofing*. Disponível em: https://blogs.mtu.edu/improvement/2013/08/20/poke-yoke-mistake-proofing/. Acesso em: 08 jul. 2025.

TULIP.CO. *The Ultimate Guide to Poka-Yoke*. Disponível em: https://tulip.co/ebooks/poka-yoke. Acesso em: 08 jul. 2025.

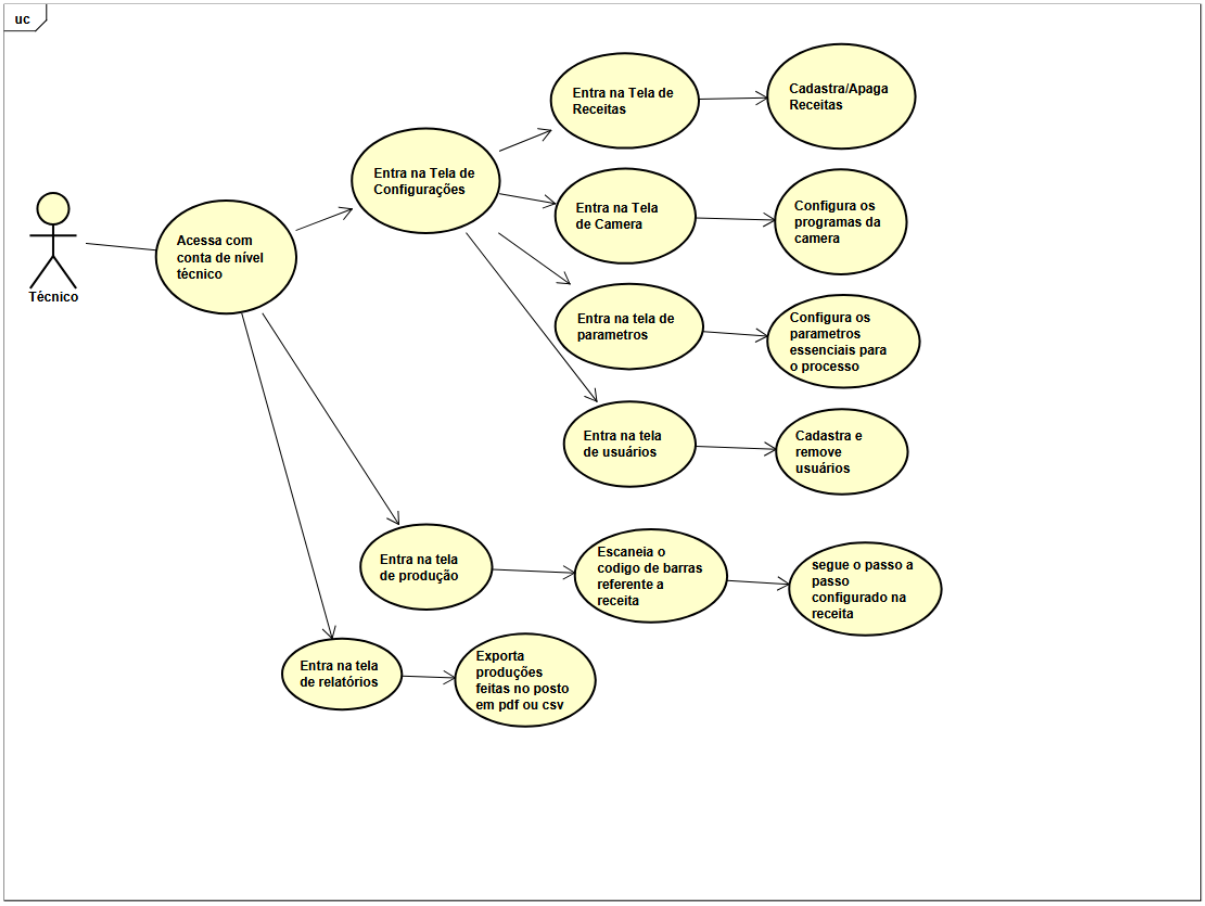
UNLEASHEDSOFTWARE.COM. *Poka-yoke in manufacturing: methods, pros, cons & examples*. Disponível em: https://www.unleashedsoftware.com/blog/poka-yoke-in-manufacturing-methods-pros-cons-examples. Acesso em: 08 jul. 2025.

BEEWATEC.COM. *Poka-yoke simply explained*. Disponível em: https://www.beewatec.com/en/blog/poka-yoke-simply-explained-simply-avoiding-errors-in-the-industry. Acesso em: 08 jul. 2025.

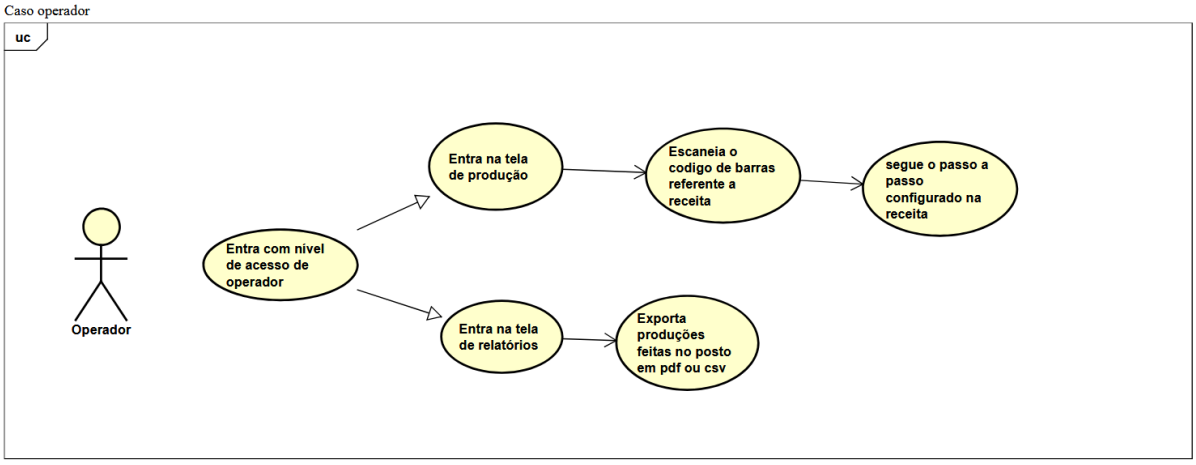
FRACTORY.COM. *Poka-yoke in manufacturing*. Disponível em: https://fractory.com/poka-yoke-in-manufacturing. Acesso em: 08 jul. 2025.

APÊNDICE 1 – DIAGRAMA DE CASO DE USO, DESCRIÇÃO DE CENÁRIOS

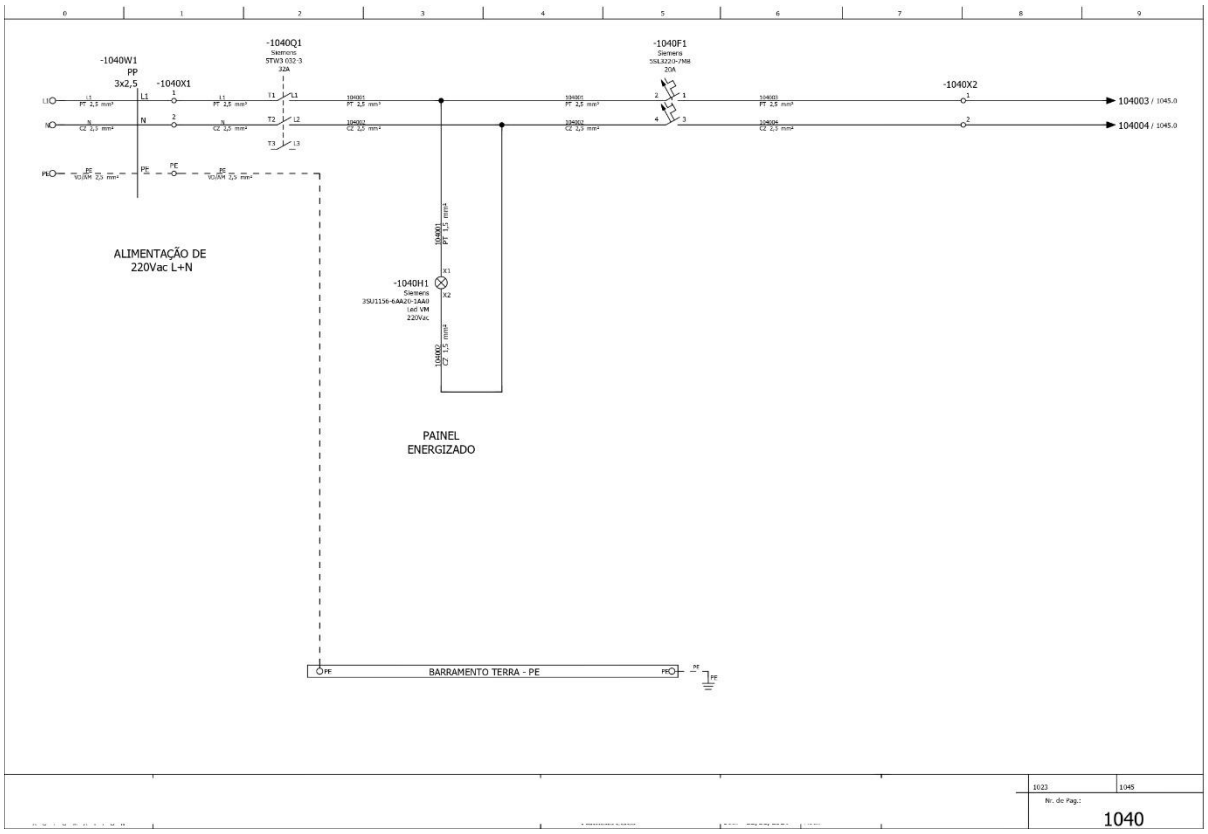
Caso Técnico 2025/06/05

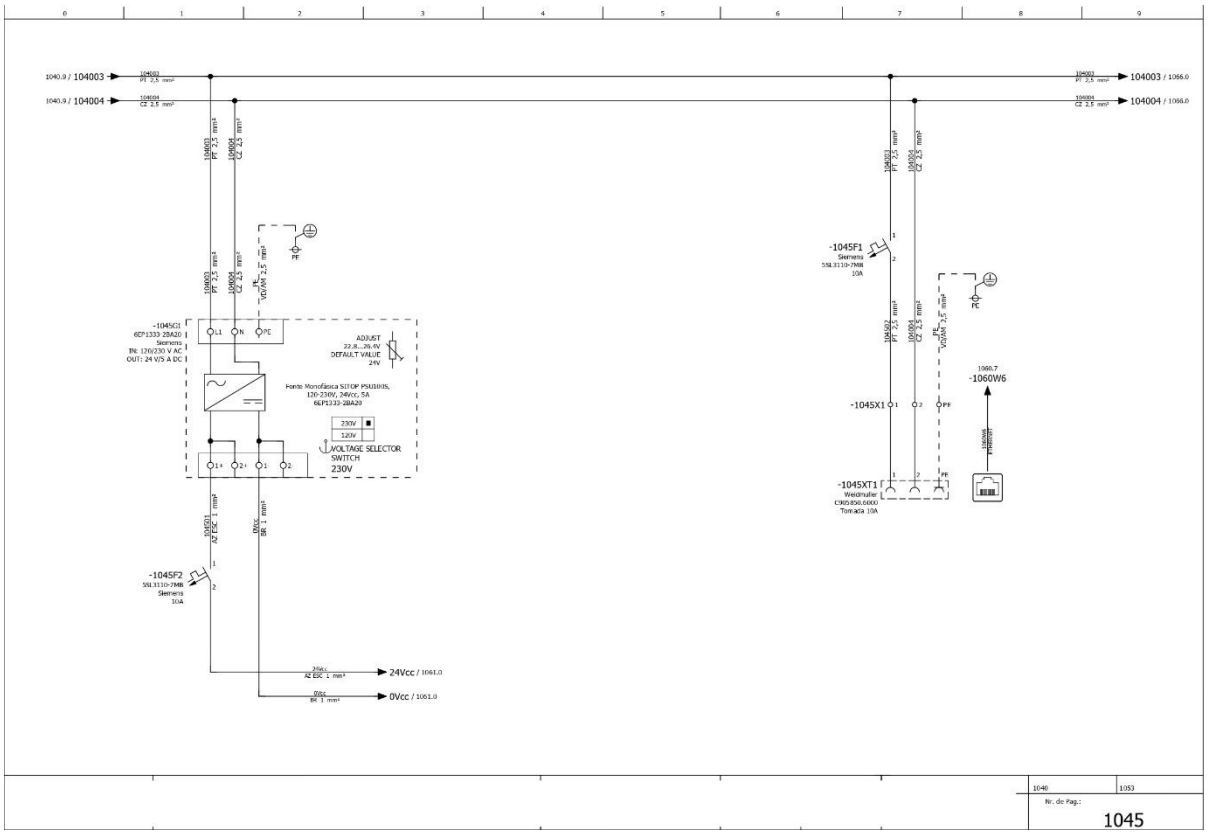


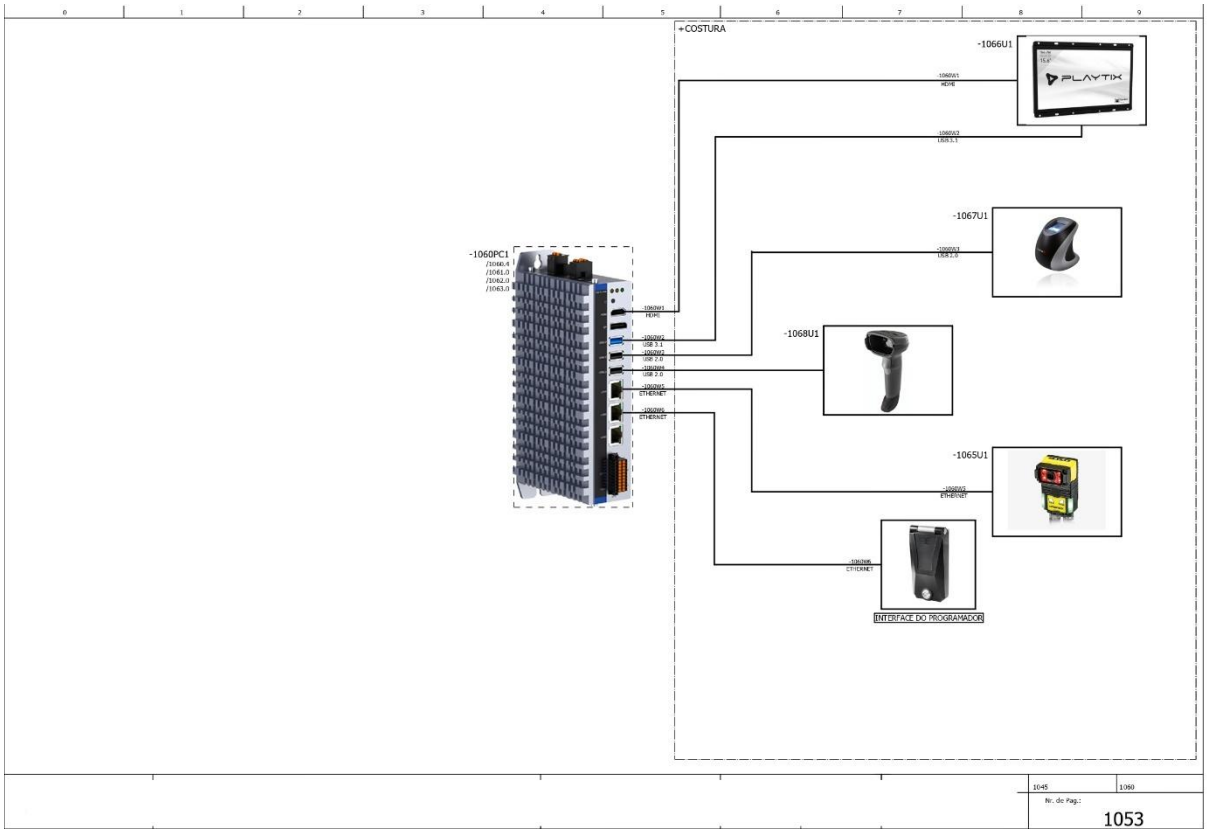
Caso operador



APENDICE 2 – DIAGRAMA ELÉTRICO



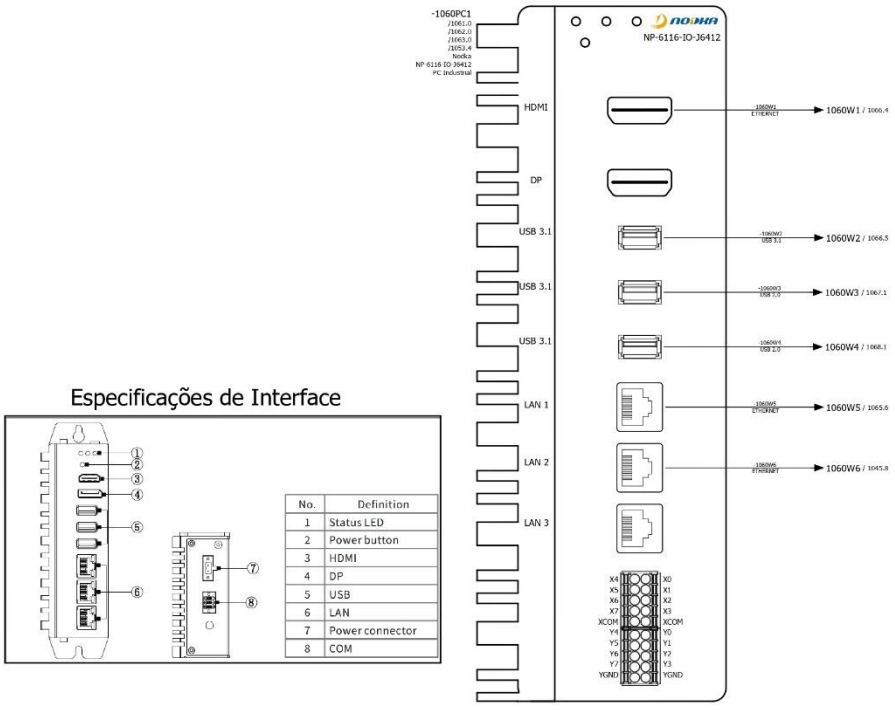




1045 | 1060

Nr. de Pag.:

1053



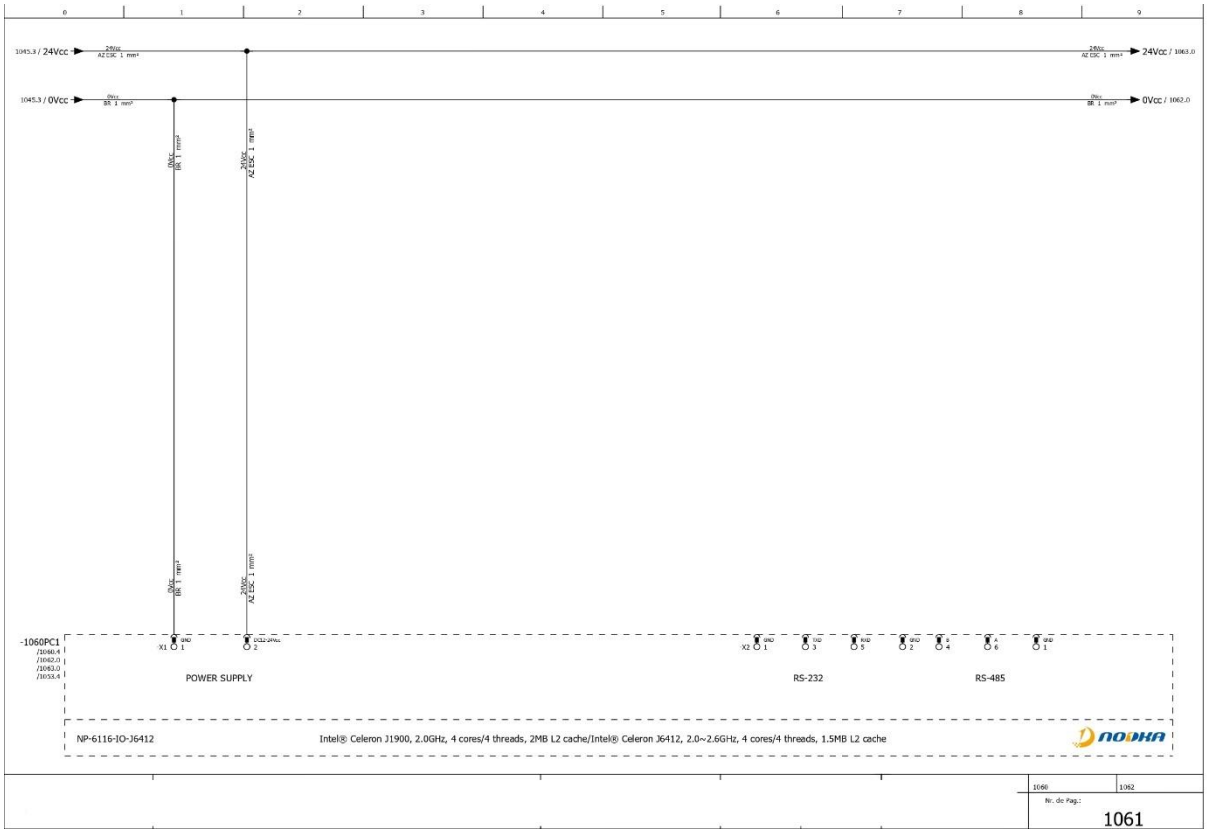
Especificações de Interface

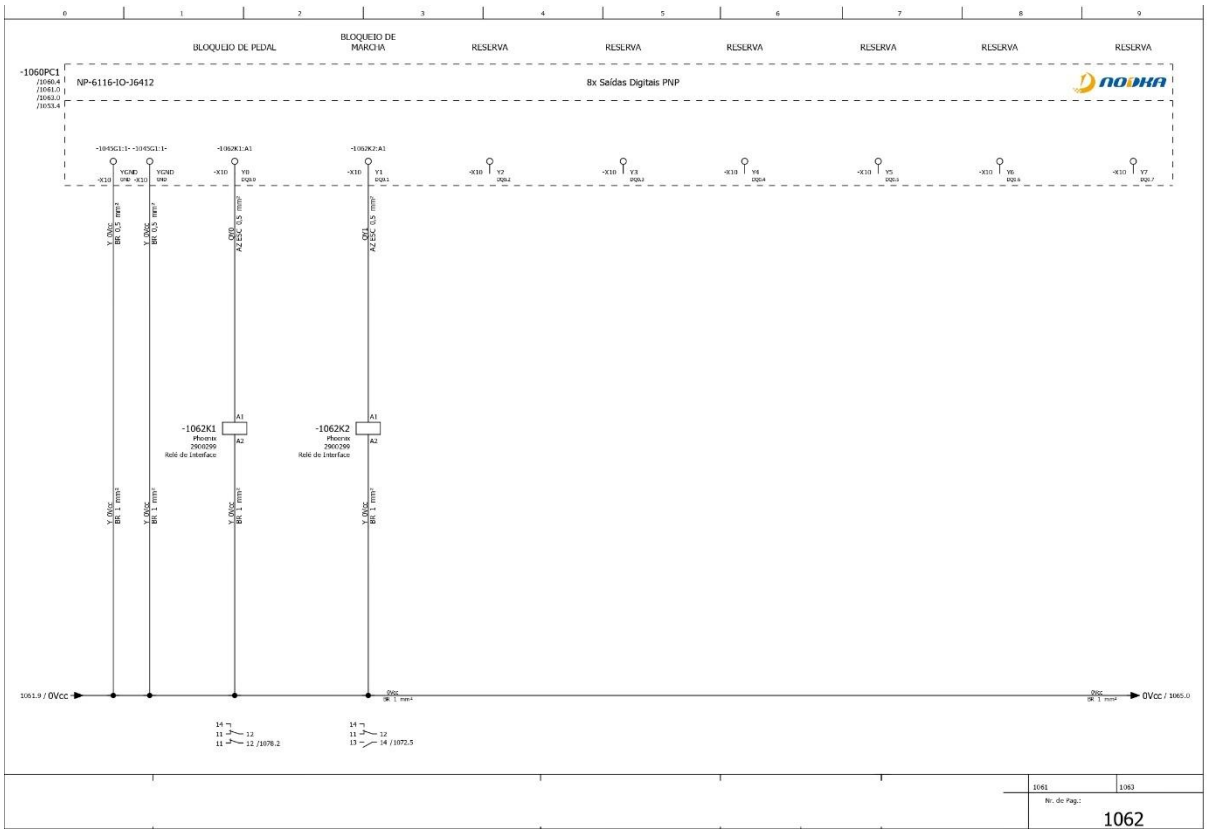
No.	Definition
1	Status LED
2	Power button
3	HDMI
4	DP
5	USB
6	LAN
7	Power connector
8	COM

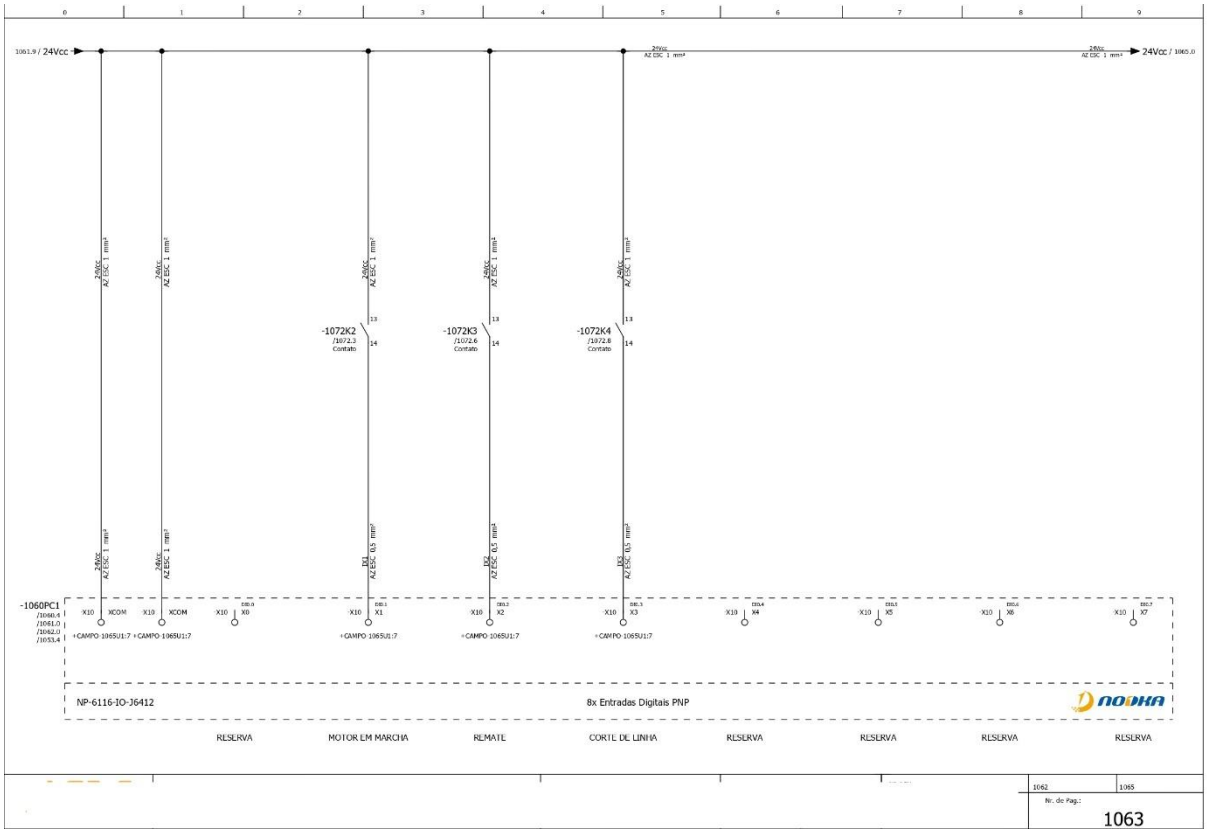
1053 | 1061

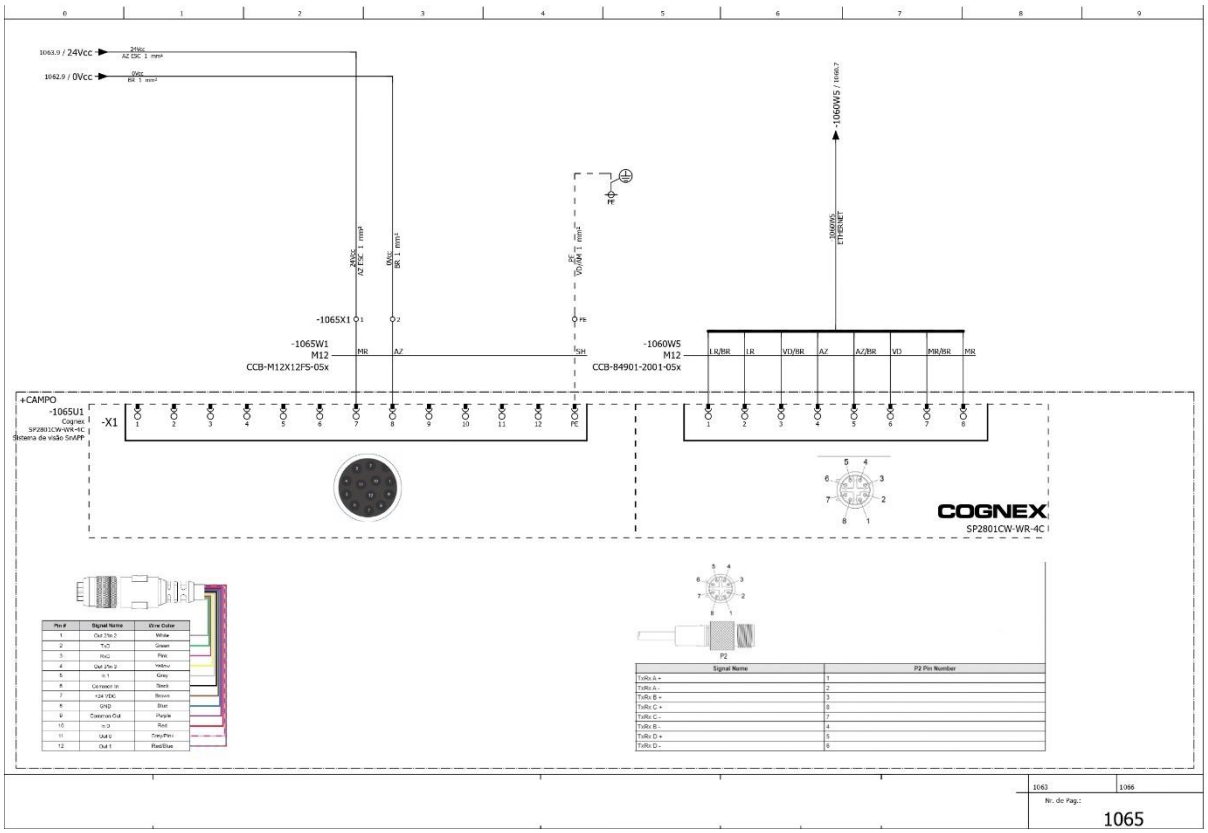
Nr. de Pag.:

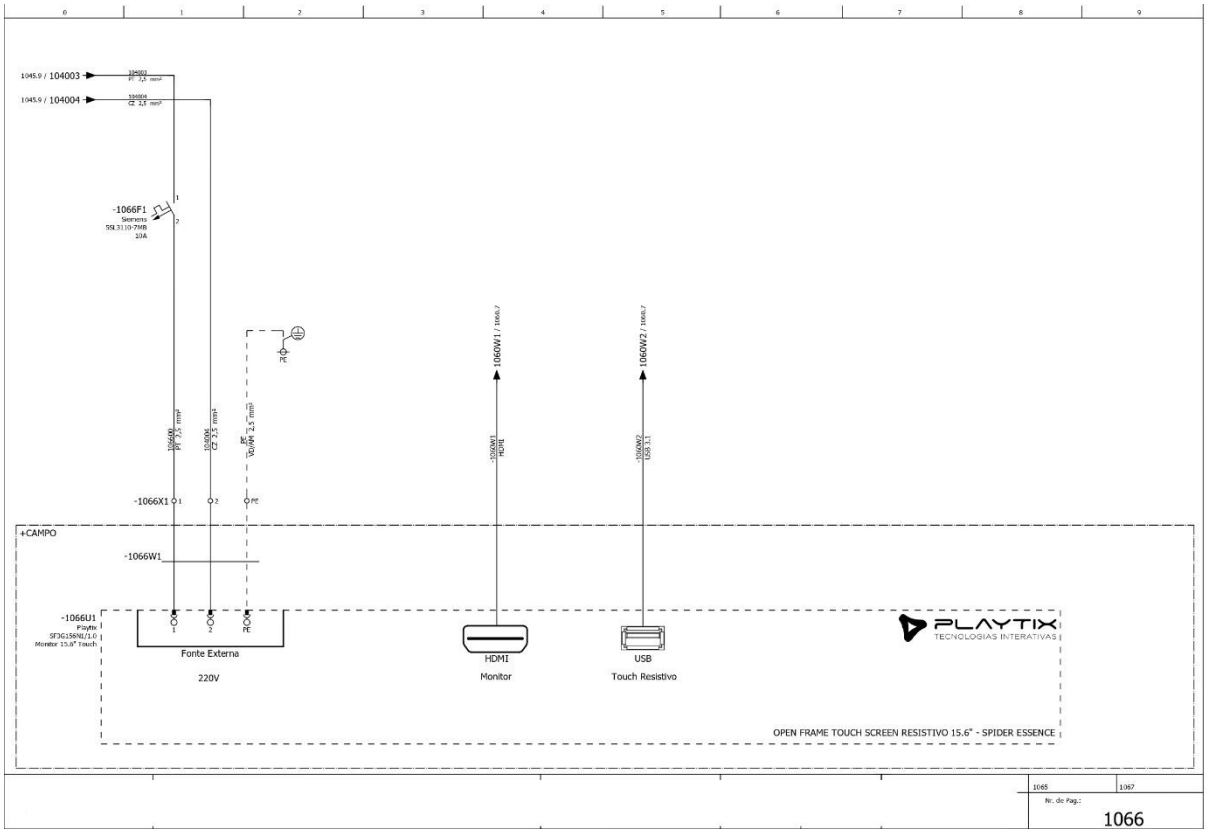
1060

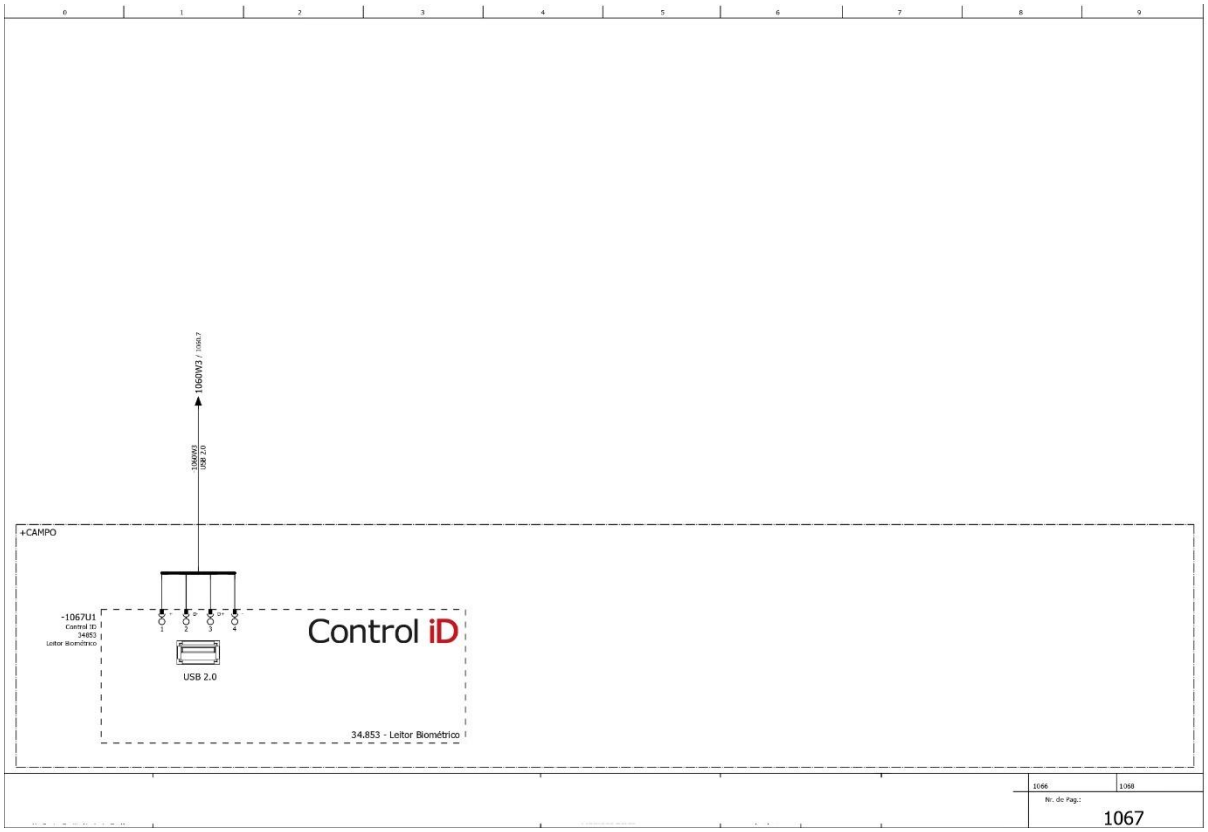

1045.3 / 24Vcc
1045.3 / 0Vcc
24Vcc / 1063.0
0Vcc / 1062.0
-1060PC1
/1060.4
/1062.0
/1063.0
/1033.4
X1
X2
POWER SUPPLY
RS-232
RS-485
NP-6116-IO-J6412
Intel® Celeron J1900, 2.0GHz, 4 cores/4 threads, 2MB L2 cache/Intel® Celeron J6412, 2.0~2.6GHz, 4 cores/4 threads, 1.5MB L2 cache
1060
1062
Nr. de Pag.:
1061

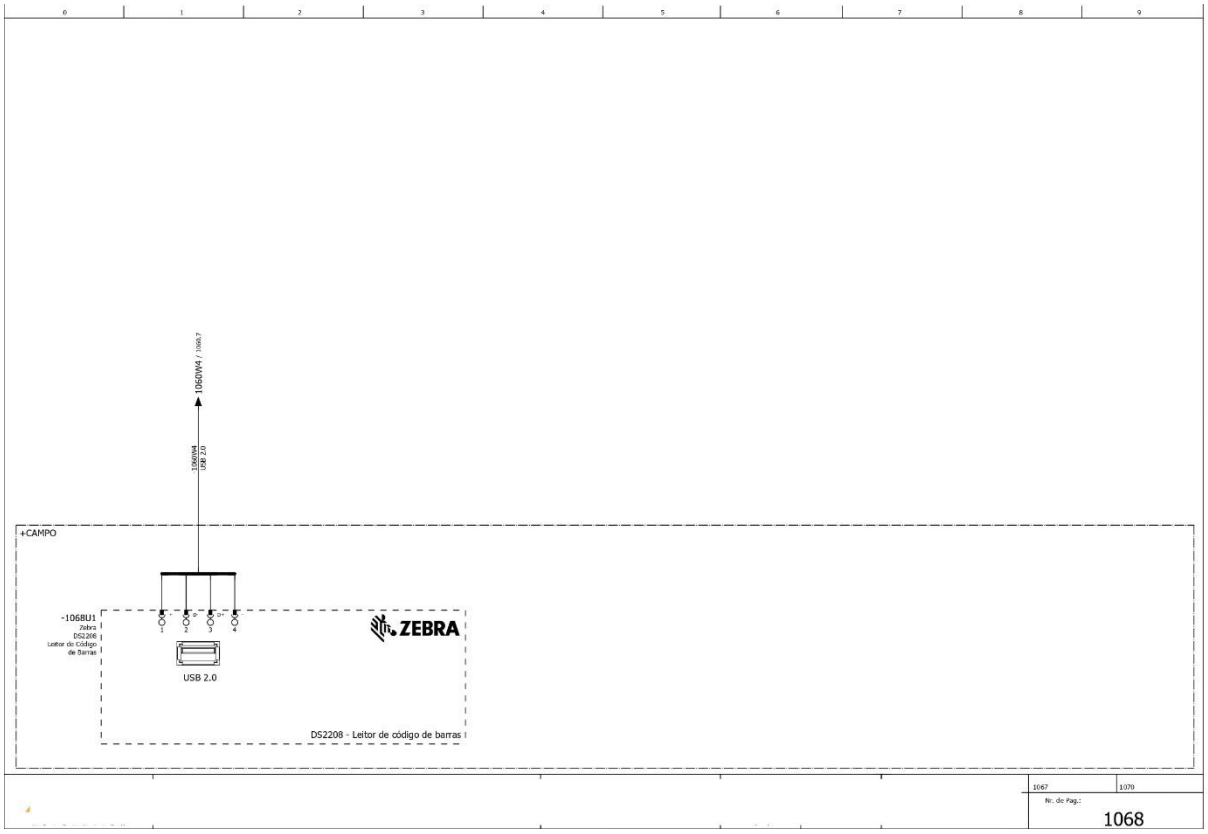

BLOQUEIO DE PEDAL
BLOQUEIO DE MARCHA
RESERVA
-1060PC1
/1060.4
/1061.0
/1062.0
/1033.4
NP-6116-IO-J6412
8x Saídas Digitais PNP
-1062K1
Phoenix
2900299
Relé de Interface
-1062K2
Phoenix
2900299
Relé de Interface
1061.9 / 0Vcc
0Vcc / 1065.0
12 /1078.2
14 /1072.5
1061
1063
Nr. de Pag.:
1062


1061.9 / 24Vcc
24Vcc / 1065.0
-1072K2
/1072.3
Contato
-1072K3
/1072.6
Contato
-1072K4
/1072.8
Contato
-1060PC1
/1060.4
/1061.0
/1062.0
/1063.4
+CAMPO-1065U1:7
NP-6116-IO-J6412
8x Entradas Digitais PNP
RESERVA
MOTOR EM MARCHA
REMATE
CORTE DE LINHA
RESERVA
RESERVA
RESERVA
RESERVA


1063.9 / 24Vcc
1062.9 / 0Vcc
-1065X1
-1065W1
M12
CCB-M12X12FS-05x
-1060W5
M12
CCB-84901-2001-05x
-1060W5 / 1060.7
+CAMPO
-1065U1
Cognex
SP2801CW-WR-4C
Sistema de visão SmAPP
-X1
COGNEX
SP2801CW-WR-4C
P2
Signal Name
P2 Pin Number


1045.9 / 104003
1045.9 / 104004
-1066F1
Siemens
5SL3110-7MB
10A
1060W1 / 1066.7
1060W2 / 1066.7
1060W1
HDMI
1060W2
USB 3.1
-1066X1
1
2
PE
+CAMPO
-1066W1
-1066U1
Playtix
SF3G156NE/1.0
Monitor 15.6" Touch
Fonte Externa
220V
HDMI
Monitor
USB
Touch Resistivo
PLAYTIX
TECNOLOGIAS INTERATIVAS
OPEN FRAME TOUCH SCREEN RESISTIVO 15.6" - SPIDER ESSENCE
1065
1067
Nr. de Pag.:
1066


1060W3 / 1066.7
1060W3
USB 2.0
+CAMPO
-1067U1
Control iD
34853
Leitor Biométrico
1
2
3
4
USB 2.0
Control iD
34.853 - Leitor Biométrico
1066
1068
Nr. de Pag.:
1067

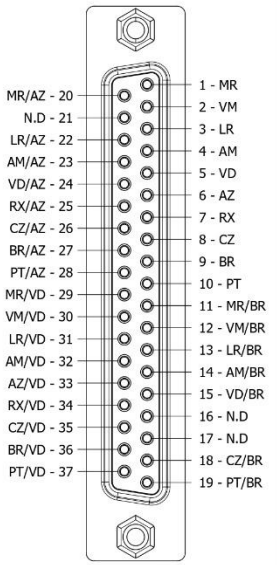


1067 | 1070

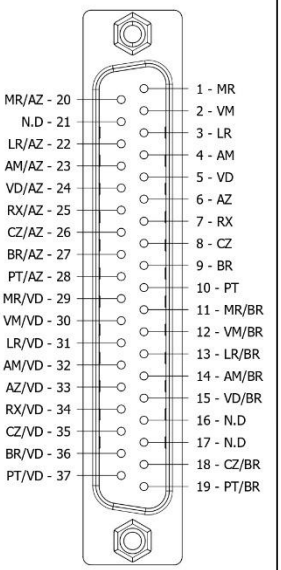
Nr. de Pag.:

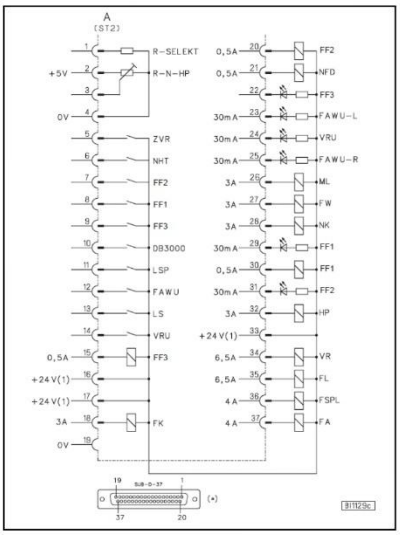
1068

PINAGEM / CABO FÊMEA



PINAGEM / CABO MACHO



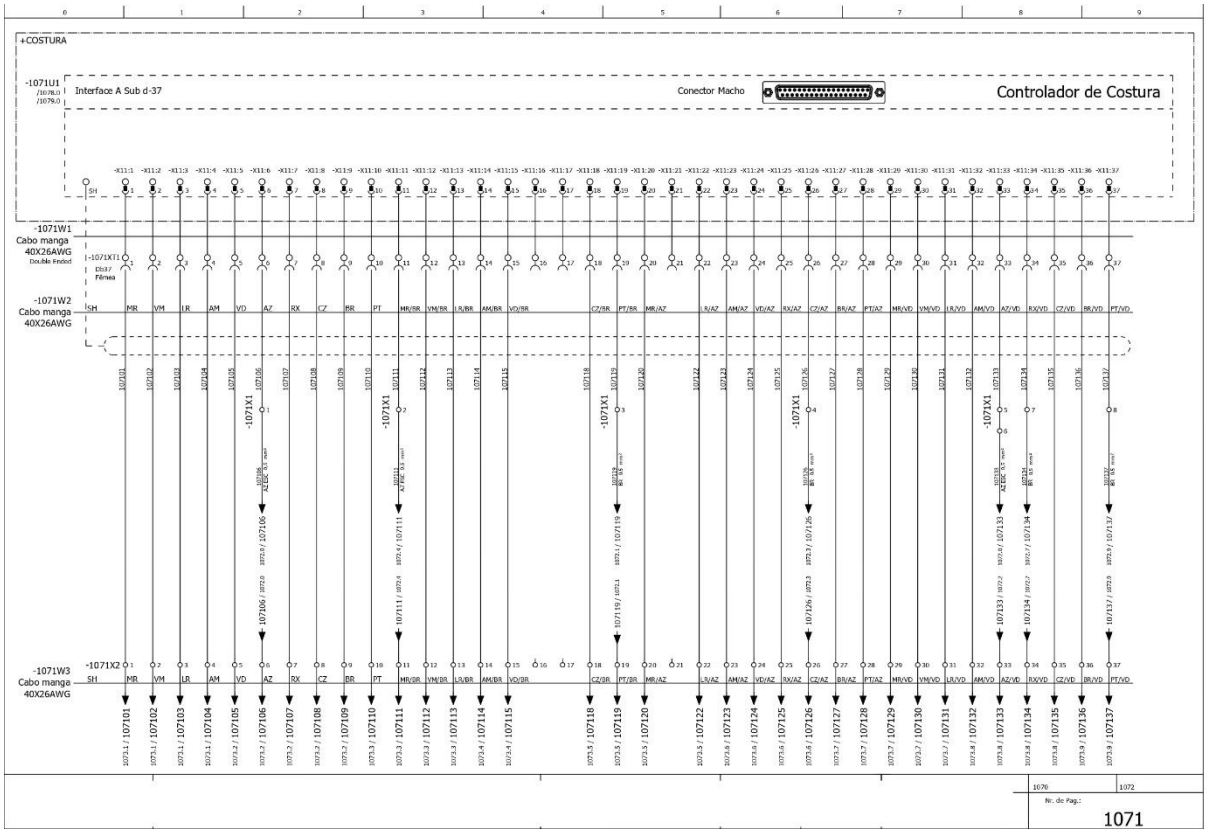


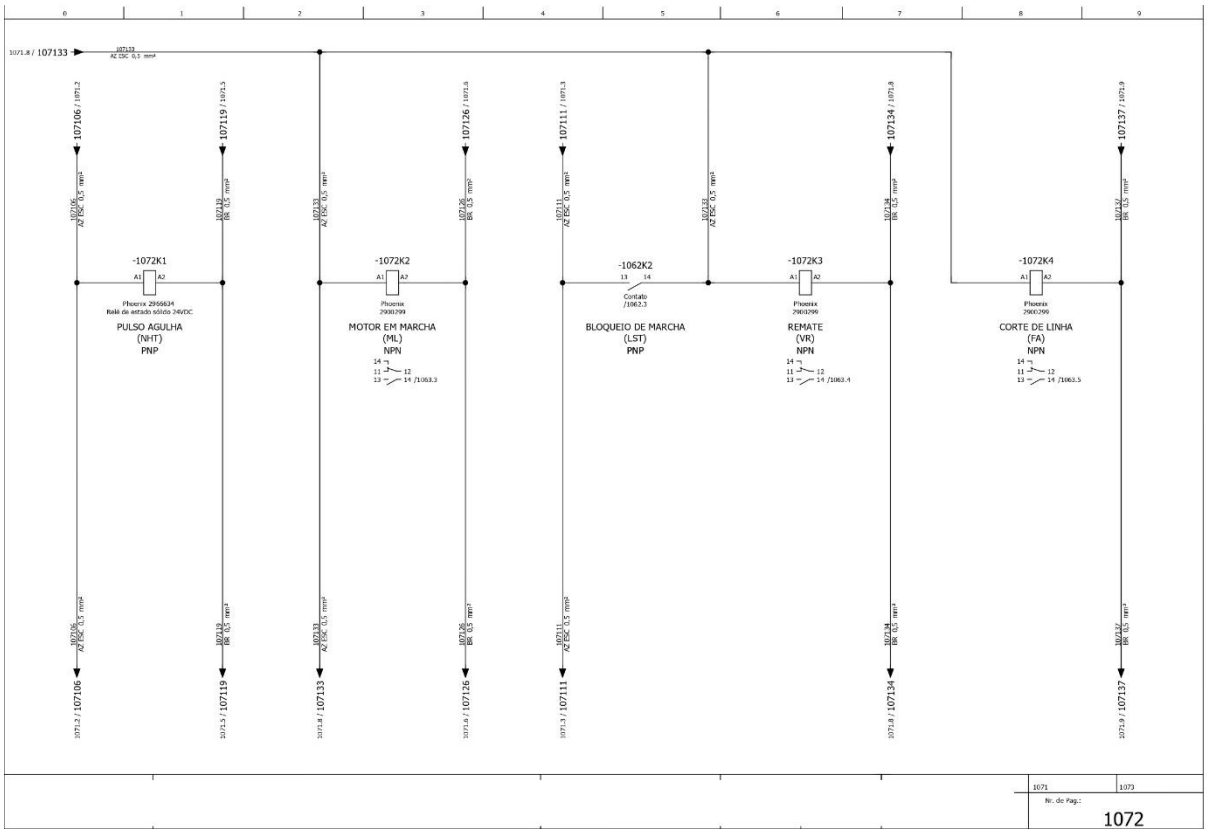
DB3000	Limitação de velocidade 3000 rpm	in8		LS	Fotocélula	in9	
FA	Corte de linha		M1	LSP	Bloqueio de marcha	in2	
FAWU	Detector de linha em baixo	in6		ML	Motor em marcha		M14
FAWU-L	Detector de linha em baixo à esquerda		M7	NFD	Pressão do pé calcador		M17
FAWU-R	Detector de linha em baixo à direita		M9	NHT	Agulha em cimo/em baixo	in3	
FF1	Módulo de função A	in4	M6	NK	Refrigeração da agulha		M2
FF2	Módulo de função B	in1	M18	R-N-HP	Potenciómetro de valor teórico para limitação de velocidade dependente do curso		
FF3	Módulo de função C	in7	M12	R-SELEKT	Resistência para selecção da máquina		
FK	Pinça de fixação de linhas		M13	TPW	Puller		M12/M9
FL	Elevação do pé calcador			VR	Remate		
FSPL	Supressão da tensão da linha		M4	VRU	Supressão / chamada do remate	in0	M8
FW	Limpa-linhas		M3	ZVR	Remate intermédio	in5	
HP	Ajustamento de curso		M5				

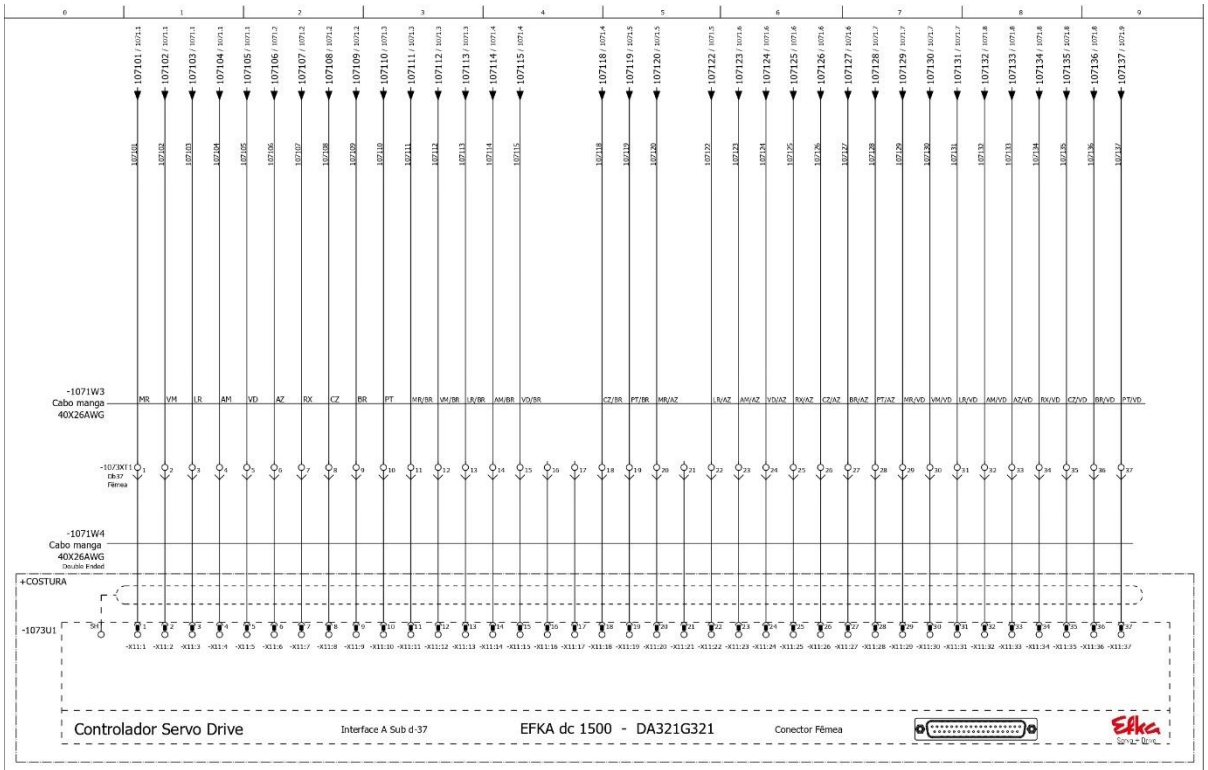
1068 | 1071

Nr. de Pag.:

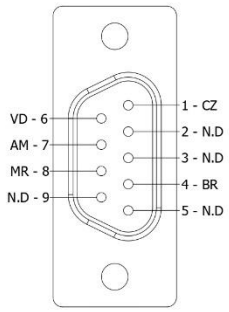
1070


+COSTURA
-1071U1
/1078.0
/1079.0
Interface A Sub d-37
Conector Macho
Controlador de Costura
-1071W1
Cabo manga
40X26AWG
Double Ended
-1071X1
DB37
Fêmea
-1071W2
Cabo manga
40X26AWG
-1071X1
-1071W3
Cabo manga
40X26AWG
-1071X2
SH MR VM LR AM VD AZ RX CZ BR PT MR/BR VM/BR LR/BR AM/BR VD/BR CZ/BR PT/BR MR/AZ LR/AZ AM/AZ VD/AZ RX/AZ CZ/AZ BR/AZ PT/AZ MR/VD VM/VD LR/VD AM/VD AZ/VD RX/VD CZ/VD BR/VD PT/VD
1070
1072
Nr. de Pag.:
1071

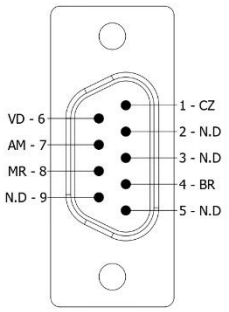

-1072K1
PULSO AGULHA
(NHT)
PNP
Phoenix 2966634
Relé de estado sólido 24VDC
-1072K2
MOTOR EM MARCHA
(ML)
NPN
Phoenix
2900299
-1062K2
Contato
/1062.3
BLOQUEIO DE MARCHA
(LST)
PNP
-1072K3
REMATE
(VR)
NPN
Phoenix
2900299
-1072K4
CORTE DE LINHA
(FA)
NPN
Phoenix
2900299
1071
1073
Nr. de Pag.:
1072

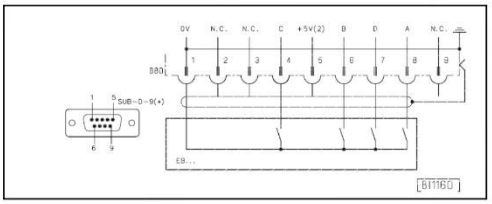


PINAGEM / CABO FÊMEA

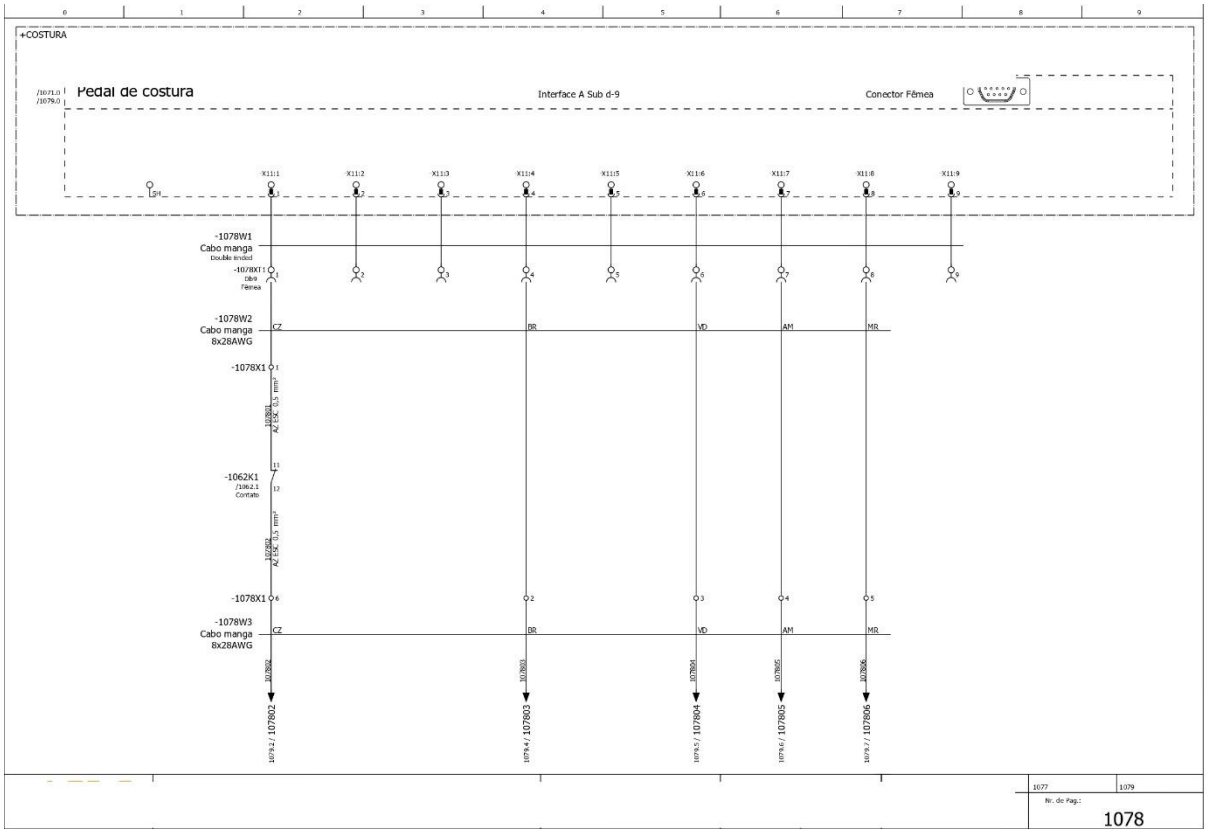


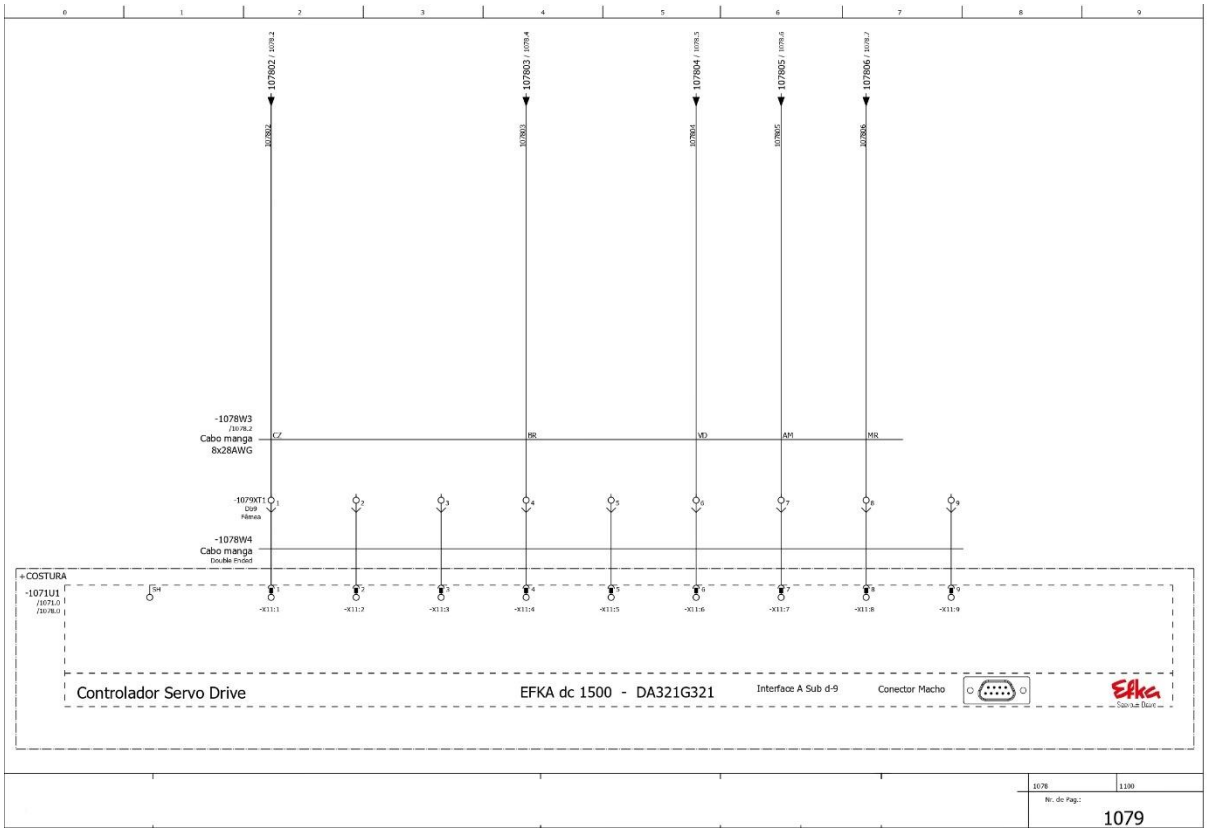
PINAGEM / CABO MACHO

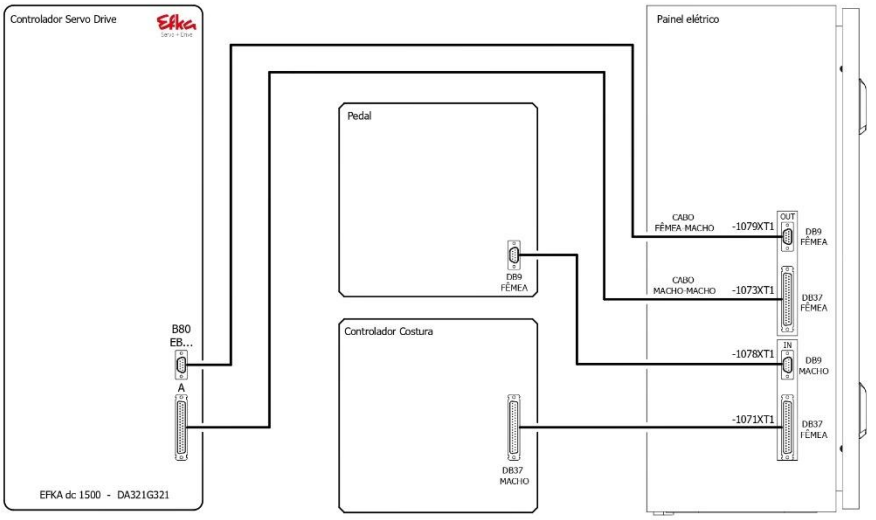


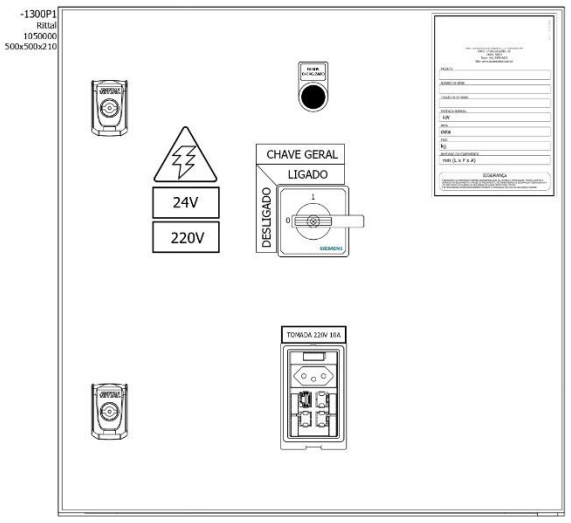


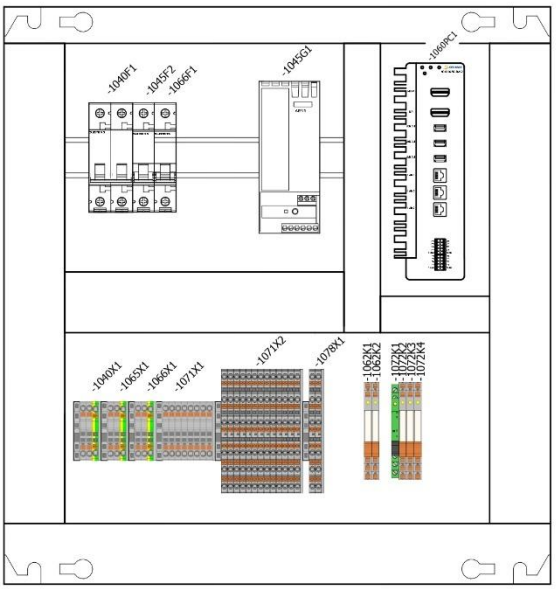
Nivel do pedal →	-2	-1	0	½	1	2	3	4	5	6	7	8	9	10	11	12
Entrada A	L	L	H	H	H	L	L	H	H	L	L	H	H	L	L	H
Entrada B	L	H	H	L	L	L	H	H	H	H	L	L	L	L	H	H
Entrada C	H	H	H	H	L	L	L	L	L	L	L	L	H	H	H	H
Entrada D	H	H	H	H	H	H	H	H	L	L	L	L	L	L	L	L

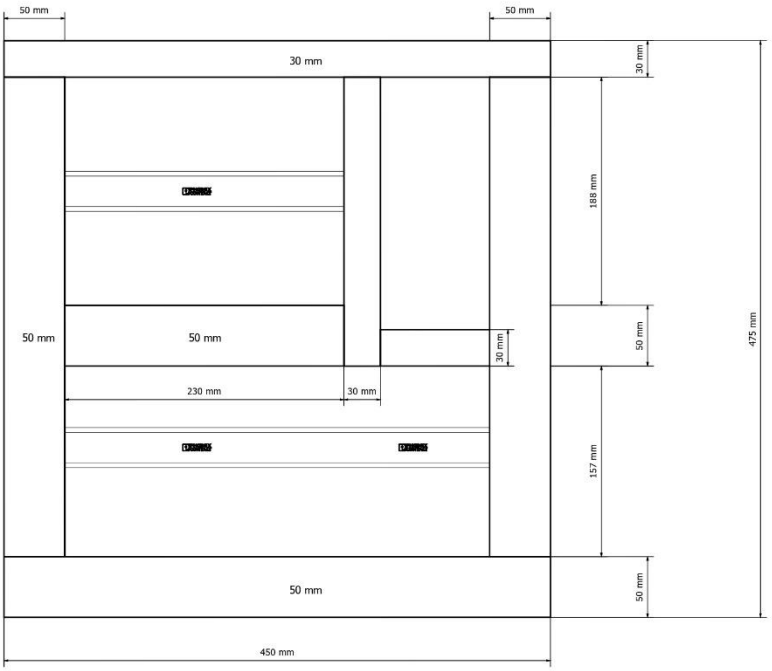

+COSTURA
Pedal de costura
Interface A Sub d-9
Conector Fêmea
-1078W1
Cabo manga
Double Ended
-1078XT1
Db9
Fêmea
-1078W2
Cabo manga
8x28AWG
-1078X1
-1062K1
/1062.1
Contato
-1078X1
-1078W3
Cabo manga
8x28AWG
CZ
BR
VD
AM
MR
1078.2 / 1078.4 / 1078.5 / 1078.6 / 1078.7

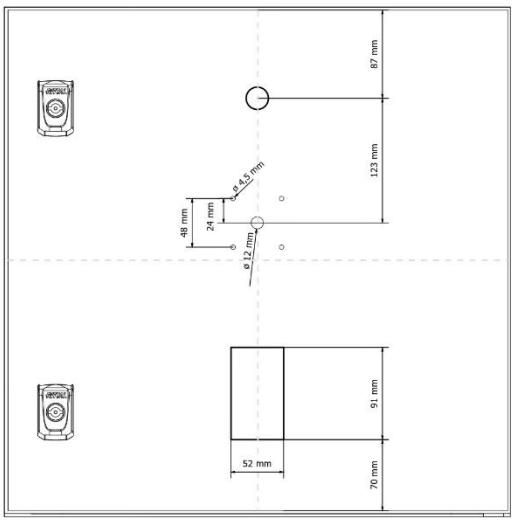

1078.2 / 1078.4 / 1078.5 / 1078.6 / 1078.7
-1078W3
/1078.2
Cabo manga
8x28AWG
CZ
BR
VD
AM
MR
-1079XT1
Db9
Fêmea
-1078W4
Cabo manga
Double Ended
+COSTURA
-1071U1
/1071.0
/1078.0
Controlador Servo Drive
EFKA dc 1500 - DA321G321
Interface A Sub d-9
Conector Macho

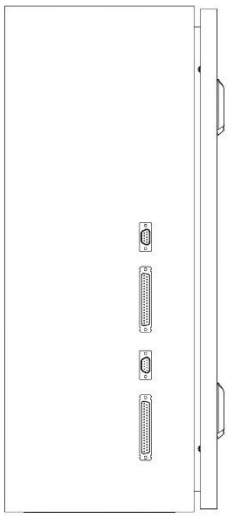

Controlador Servo Drive
Efka
Servo + Drive
Painel elétrico
Pedal
CABO
FÊMEA-MACHO
-1079XT1
OUT
DB9
FÊMEA
DB9
FÊMEA
CABO
MACHO-MACHO
-1073XT1
DB37
FÊMEA
B80
EB...
A
Controlador Costura
-1078XT1
IN
DB9
MACHO
-1071XT1
DB37
FÊMEA
DB37
MACHO
EFKA dc 1500 - DA321G321


-1300P1
Rittal
1050000
500x500x210
CHAVE GERAL
LIGADO
DESLIGADO
24V
220V
TOMADA 220V 10A


-1040F1
-1045F2
-1066F1
-1045G1
-1069PC1
-1040X1
-1065X1
-1066X1
-1071X1
-1071X2
-1078X1
1062K1
1062K2
1072K1
1072K2
1072K3
1072K4


50 mm
50 mm
30 mm
30 mm
188 mm
50 mm
50 mm
475 mm
30 mm
230 mm
30 mm
157 mm
50 mm
50 mm
450 mm


87 mm
123 mm
ø 4,5 mm
48 mm
24 mm
ø 12 mm
91 mm
52 mm
70 mm



APÊNDICE 3 – CRONOGRAMA

SEMANAS 1 e 2 – PLANEJAMENTO E DEFINIÇÃO DO TEMA

- Definir tema e objetivos do TCC.
- Estabelecer escopo do projeto.
- Elaborar o cronograma inicial.
- Definir ferramentas e tecnologias (PC industrial, sensores, câmera Cognex, SQL Server, etc.).
- Reunir requisitos do sistema com base nas necessidades do setor de costura.

SEMANAS 3 e 4 – REVISÃO BIBLIOGRÁFICA

- Identificar artigos e publicações relevantes sobre sistemas embarcados e automação.
- Analisar tecnologias existentes para automação de máquinas de costura.
- Estudar protocolos de comunicação e integração de sensores e câmeras.
- Resumir as principais descobertas para inclusão na introdução do TCC.

SEMANAS 5 e 6 – ELABORAÇÃO DO PROJETO

- Escrever os objetivos, justificativa e metodologia do projeto.
- Elaborar o escopo detalhado do sistema.
- Definir arquitetura geral do sistema (hardware e software).
- Planejar integração entre sensores, câmera e PC industrial.
- Criar fluxograma inicial do software.

SEMANAS 7 e 8 – APROVAÇÃO DO PROJETO

- Revisar o projeto com o orientador.
- Ajustar o projeto conforme o feedback.
- Formalizar a aprovação do projeto.

SEMANAS 9 e 10 – PROJETO ELÉTRICO

- Selecionar componentes eletrônicos e módulos I/O.
- Desenhar diagramas elétricos e blocos funcionais do sistema.

- Verificar compatibilidade dos componentes (GPIO, entradas rápidas, etc.).
- Validar o design do circuito com o orientador.

SEMANAS 11 e 12 – DESENVOLVIMENTO DO BACKEND

- Configurar ambiente de desenvolvimento (Visual Studio, .NET MAUI, etc.).
- Criar classes para comunicação com hardware (sensores, câmera).
- Desenvolver lógica de controle e processamento de dados.
- Testar módulos individualmente (sensores, entradas digitais, etc.).

SEMANAS 13 e 14 – DESENVOLVIMENTO DO FRONTEND

- Criar interfaces gráficas para operação do sistema.
- Implementar telas de configuração, operação e monitoramento.
- Testar a integração do frontend com o backend.

Semanas 15 e 16 – INTEGRAÇÃO DE HARDWARE E SOFTWARE

- Montar o protótipo físico do sistema.
- Conectar módulos I/O, sensores e câmera ao PC industrial.
- Realizar testes iniciais de integração.

SEMANAS 17 e 18 – CONFIGURAÇÃO E TESTES DOS SENSORES

- Ajustar parâmetros dos sensores (Keyence).
- Testar leitura precisa de sinais digitais e analógicos.
- Validar tempo de resposta e precisão.

SEMANAS 19 e 20 – CONFIGURAÇÃO E TESTE DA CAMERA

- Configurar comunicação com a câmera Cognex.
- Testar captura de imagens e reconhecimento de objetos.
- Validar integração com o software principal.

SEMANAS 21 e 22 – IMPLEMENTAÇÃO DO BANCO DE DADOS

- Configurar banco de dados SQL Server.
- Implementar lógica de armazenamento e consulta de dados.

- Validar persistência e integridade dos dados.

SEMANAS 23 e 24 – TESTES E VALIDAÇÃO DO SISTEMA

- Testar o sistema em condições reais.
- Verificar a robustez e estabilidade do software.
- Ajustar parâmetros conforme resultados dos testes.

SEMANAS 25 e 26 – DOCUMENTAÇÃO DO TCC

- Escrever relatório técnico detalhado.
- Incluir diagramas, fluxogramas e resultados de testes.
- Revisar documentação com o orientador.

SEMANAS 27 – PREPARAÇÃO DA APRESENTAÇÃO

- Criar slides para a apresentação.
- Preparar material de apoio e possíveis demonstrações.

SEMANAS 28 – REVISÃO E AJUSTES FINAIS

- Revisar o relatório e a apresentação.
- Fazer ajustes finais com base no feedback do orientador.

SEMANAS 29 – ENTREGA DO TRABALHO FINAL

- Submeter o TCC para a banca examinadora.
- Confirmar entrega de todos os documentos necessários.

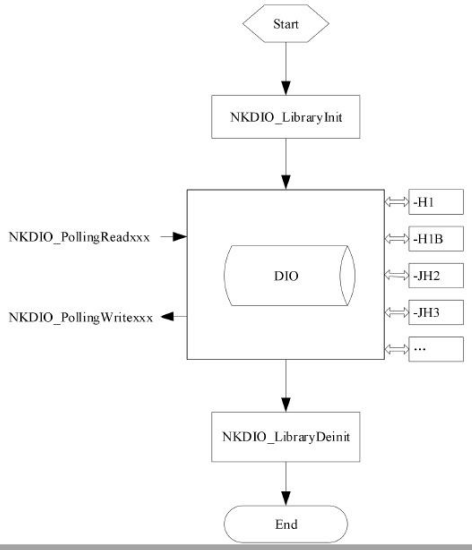
SEMANAS 30 – APRESENTAÇÃO DO TCC

- Realizar a apresentação para a banca examinadora.
- Responder perguntas e discutir os resultados do trabalho.

APENDICE 4 – BIBLIOTECA NODKA

4API usage

The dynamic library can be used to access many add-on boards, the library must be initialized before connecting to the device and accessing the data, and library process function must be called cyclically to execute the command and communicate with the devices. There is a profile file which stored the configure information for the board, which must be imported during the library initialization. The blow is the logic schematic for your reference.



4.1API function list

Function Name	Description
DIO Library	
DIO Library Base	
NKDIO_LibraryInit	Init the DIO library and the drivers
NKDIO_LibraryDeinit	Deinitialize the DIO library and release the resources
DIO Functions	
NKDIO_PollingReadDiByte	Read the DI port in the polling mode
NKDIO_PollingReadDiWord	Read two DI ports in the polling mode
NKDIO_PollingWriteDoByte	Write the DO port in the polling mode
NKDIO_PollingWriteDoWord	Write two DO ports in the polling mode
NKDIO_PollingReadDoByte	Read back the set value of the digital output port(8 bits).
NKDIO_PollingReadDoWord	Read back the set value of the digital output port(16 bits).

4.2DIO Library

4.2.1DIO Library Base

4.2.1.1NKDIO_LibraryInit

- **Description**
  DIO Library initialized.
- **Prototype**

FONTE: NKIOLIB API user manual_V5.0.0.

int NKDIO_LibraryInit(const char * configFile)

- **Parmeters**
  - Input
    - configIniFile：the name and path of the configure file. The configuration for the IO for each add-on board is stored in the ini format file, which can be got in the according folder after the SDK is installed.
  - Output
    - None
- **Return**
  Return 0 if success; other return error codes.
- **Other**
  The function should be called firstly but only once to allocate the resources before using the I/O access functions in the library.
- **Usage**
  int ret = NKDIO_LibraryInit ("./nkio_config.ini");

4.2.1.2NKDIO_LibraryDeinit

- **Description**
  The function to be used to release the DIO library resources before the application exit.
- **Prototype**
  void NKDIO_LibraryDeinit(void)
- **Parmeters**
  - Input
    - None
  - Output
    - None
- **Return**
  None.
- **Other**
  The function should be called before the application exit.
- **Usage**

```
int ret = NKDIO_LibraryInit("nkio_config.ini");
if( ret == NKIO_ENOERR)
{
    While(1)
    {
       ……
      Sleep(1);
    }

}
NKDIO_LibraryDeinit();
```

The library provide the functions to access the digital I/O in the polling mode, but in the case of multi-thread access the independent I/O channel, the critical mutex must be used to protect the resources.

4.2.2DIO Functions

4.2.2.1NKDIO_PollingReadDiByte

- **Description**
  Read digital input port value(8 channels) .
- **Prototype**
  int NKDIO_PollingReadDiByte(unsigned char diByteIndex,unsigned char *pByteValue)
- **Parmeters**
  - Input
    - diByteIndex：The byte index of the digital input channel located, for the first 8 channel, the diByteIndex is set to 0, while the 8~15 channel is set to 1.
  - Output
    - pByteValue**:** Pointer to store the read data

FONTE: NKIOLIB API user manual_V5.0.0.

- **Return**
  Return 0 if success; other return error codes.
- **Other**
  This function must be used after the DIO port is initialized.
- **Usage**

```
unsigned char port0Value = 0;
unsigned char port0Index = 0;
int ret = NKIO_ENOERR;
if(NKIO_ENOERR == NKDIO_LibraryInit("./nkio_config.ini"))
{
    ret = NKDIO_PollingReadDiByte(port0Index, &port0Value);  // Read data from port0.
    if(ret != NKIO_ENOERR )
    {
        printf("Read Data Error!, ErrorCode: %d\n\r", ret);
        return ret;
    }
    else
    {
        printf("Read Data Success!, Port value: %d\n\r", port0Value);
        return ret;

    }
    ……
}
else
{
      printf("DIO library initialized Error!, ErrorCode: %d\n\r", ret);
      return ret;
}
```

4.2.2.2NKDIO_PollingReadDiWord

- **Description**
  Read digital input port value(16 channels) .
- **Prototype**
  int NKDIO_PollingReadDiWord(unsigned char diWordIndex, unsigned short *pWordValue)
- **Parmeters**
  - Input
    - diWordIndex：Index of the first DI port of both DI input ports, the first two ports set to 0.
  - Output
    - pWordValue**:** Pointer to store the read data.
- **Return**
  Return 0 if success; other return error codes.
- **Other**
  This function must be used after the DIO port is initialized.
- **Usage**

```
unsigned short wordValue = 0;
unsigned char wordIndex = 0;
int ret = NKIO_ENOERR;
if(NKIO_ENOERR == NKDIO_LibraryInit("./nkio_config.ini"))
{
    ret = NKDIO_PollingReadDiWord(wordIndex, & wordValue);  // Read data from the first 2 ports.
    if(ret != NKIO_ENOERR )
    {
        printf("Read Data Error!, ErrorCode: %d\n\r", ret);
        return ret;
    }
    else
    {
        printf("Read Data Success!, Word value: %d\n\r", wordValue);
        return ret;
```


FONTE: NKIOLIB API user manual_V5.0.0.

```
    }
    ……
}
else
{
      printf("DIO library initialized Error!, ErrorCode: %d\n\r", ret);
      return ret;
}
```

4.2.2.3NKDIO_PollingWriteDoByte

- **Description**
  Write the output port value (8 channels).
- **Prototype**
  int NKDIO_PollingWriteDoByte(BYTE doByteIndex, BYTE doByteValue)
- **Parmeters**
  - Input
    - doByteIndex：The byte index of the digital output channel located, for the first 8 channel, the doByteIndex is set to 0, while the 8~15 channel is set to 1.
    - doByteValue: the port status to be set, from 0 to 0xFF. 0x0 to turn on all of the port, and set to 0xFF to reset the port.
  - Output
    - None
- **Return**
  Return 0 if success; other return error codes.
- **Other**
  This function must be used after the DIO port is initialized.
- **Usage**

```
unsigned char portIdx = 0;
unsigned char portValue = 0x0F; // turn on the bit4~8 in the first port.
if(NKIO_ENOERR == NKDIO_LibraryInit("./nkio_config.ini"))
{
    ret = NKDIO_ PollingWriteDoByte (portIdx, portValue);
    if(ret != NKIO_ENOERR )
    {
       printf("Write Data Error!, ErrorCode: %d\n\r", ret);
       return ret;
    }
    else
    {
       printf("Write Data Success!\n\r");
       return ret;

    }
    ……
}
else
{
      printf("DIO library initialized Error!, ErrorCode: %d\n\r", ret);
      return ret;
}
```

4.2.2.4NKDIO_PollingWriteDoWord

- **Description**
  Write the output port value (16 channels).
- **Prototype**
  int NKDIO_PollingWriteDoWord(unsigned char doWordIndex, unsigned short doWordValue)

FONTE: NKIOLIB API user manual_V5.0.0.

- **Parmeters**
  - Input
    - doWordIndex：index of the first DO port of both DO input ports, the first two ports set to 0.
    - doWordValue:the port status to be set,, from 0x0000 to 0xFFFF. Open DDO channel according to bit set to 0, close DO channel set to 1, and the default value of DO port is 0 xFFFF.
  - Output
    - None
- **Return**

  Return 0 if success; other return error codes.
- **Other**

  This function must be used after the DIO port is initialized.
- **Usage**

```
unsigned char Idx = 0;  // for the first two ports.
unsigned short wordValue = 0xFFFE; // turn on bit 0 in the first port.
if(NKIO_ENOERR == NKDIO_LibraryInit("./nkio_config.ini"))
{
    ret = NKDIO_ PollingWriteDoWord(Idx, wordValue);
    if(ret != NKIO_ENOERR )
    {
       printf("Write Data Error!, ErrorCode: %d\n\r", ret);
       return ret;
    }
    else
    {
       printf("Write Data Success!\n\r");
       return ret;

    }
    ……
}
else
{
      printf("DIO library initialized Error!, ErrorCode: %d\n\r", ret);
       return ret;
}
```


FONTE: NKIOLIB API user manual_V5.0.0.