UNIVERSIDADE FEDERAL DE SANTA CATARINA
CENTRO TECNOLÓGICO
DEPARTAMENTO DE AUTOMAÇÃO E SISTEMAS
ENGENHARIA DE CONTROLE E AUTOMAÇÃO

**Guilherme Silva Andrade**

**Projeto de Fim de Curso**

# Protocolo e software para TAF em paineis de automação

Florianópolis

2019

**Guilherme Silva Andrade**

# Protocolo e software para TAF em paineis de automação

Projeto de Fim de Curso de Graduação em Engenharia de Controle e Automação da Universidade Federal de Santa Catarina como requisito para a obtenção do título de Bacharel em Engenharia de Controle e Automação. Orientador: Prof. Marcelo De Lellis Costa de Oliveira

Florianópolis

2019

Ficha de identificação da obra elaborada pelo autor,
através do Programa de Geração Automática da Biblioteca Universitária da UFSC.

Andrade, Guilherme Silva
Protocolo e software para TAF em paineis de automação / Guilherme Silva Andrade ; orientador, Marcelo De Lellis Costa de Oliveira, orientador, Rafael Gonçalves d'Ávila da Silva, 2019.
76 p.

Trabalho de Conclusão de Curso (graduação) - Universidade Federal de Santa Catarina, Centro Tecnológico, Graduação em Engenharia de Controle e Automação, Florianópolis, 2019.

Inclui referências.

1. Engenharia de Controle e Automação. 2. Teste de Aceitação de Fábrica. 3. TIA Portal. 4. Painel de automação. 5. Teste em painel de automação. I. Oliveira, Marcelo De Lellis Costa de. II. Silva, Rafael Gonçalves d'Ávila da III. Universidade Federal de Santa Catarina. Graduação em Engenharia de Controle e Automação. IV. Título.

Guilherme Silva Andrade

# Protocolo e software para TAF em paineis de automação

Este Projeto de Fim de Curso foi julgado adequado para obtenção do Título de "Engenheiro de Controle e Automação" e aprovado em sua forma final pelo Curso Engenharia de Controle e Automação.

Florianópolis, 12 de dezembro de 2019

**Banca Examinadora:**

---

**Eng. Rafael Gonçalves d'Ávila da Silva**
Orientador na empresa
GreyLogix Brasil

---

**Prof. Marcelo De Lellis Costa de Oliveira**
Orientador no Curso
Universidade Federal de Santa Catarina

---

**Ph.D. Eng. Rodrigo Tacla Saad**
Avaliador
Universidade Federal de Santa Catarina

---

**Gustavo Vicente Cordeiro**
Debatedor
Universidade Federal de Santa Catarina

---

**Murilo Rodrigues Padilha Leite**
Debatedor
Universidade Federal de Santa Catarina

# Agradecimentos

Primeiramente gostaria de agradecer imensamente aos meus pais, Adilson e Marilá-dia, que sempre me incentivaram, dando apoio incondicional e me proporcionando ótimos momentos ao longo da minha vida.

Agradecer também aos meus irmãos, André e Gabriel, que junto aos meus pais, sempre estiveram presentes me apoiando e me dando forças para que eu continuasse lutando até concluir essa etapa da minha vida.

Gostaria de agradecer a minha namorada Ariany, por toda compreensão e com-panheirismo demonstrada durante todo período de graduação, sempre me apoiando em todas as decisões.

Aos meus colegas de apartamento por toda a ajuda na graduação e sem dúvidas, nos momentos de distração.

Aos meus colegas da graduação e professores do departamento, por todos esses longos anos juntos, em especial ao meu orientador Marcelo e todo apoio prestado pela Universidade Federal de Santa Catarina.

À empresa GreyLogix Brasil, pela oportunidade de estágio e possibilidade da aplicação do conhecimento teórico adquirido ao longo da faculdade. Agradecimento em especial ao Rafael, por me supervisionar e possibilitar que o projeto fosse o mais proveitoso possível.

*"A menos que modifiquemos a nossa maneira de pensar, não seremos capazes de resolver os problemas causados pela forma como nos acostumamos a ver o mundo".*

*(Albert Einstein)*

# Resumo

Todo painel de automação deve ser testado antes de sair da área de montagem, a fim de validar todas as conexões elétricas. Desde o surgimento da GreyLogix Brasil, todo teste de conexões de paineis de automação sempre foi executado por dois funcionários, que são compartilhados com outra área da empresa. Esses funcionários têm como função verificar a conexão dos cartões com o ponto mais distante do painel, para assim validar a exatidão de todas as conexões, ou seja, enquanto um funcionário força um sinal, o outro verifica a existência desse sinal na outra extremidade, assim ponto a ponto.

Esse documento detalha a solução para melhoria do processo, por meio da criação de um *software* genérico para teste de paineis de automação, que necessite de apenas um funcionário, para testar qualquer painel de automação. A execução desse teste passará por uma migração de tecnologias, buscando tornar o processo mais produtivo e mais ágil.

Buscou-se como motivação a diminuição do quadro de funcionários necessário para os testes, evitando o desfalque de outra área da empresa, a diminuição significativa na interpretação elétrica do projeto, visto que o próprio *software* fará a interpretação e ganho de tempo na execução do teste.

**Palavras-chave:** TAF. Teste de Aceitação de Fábrica. Painel de automação. Teste em painel de automação.

# Abstract

The entire automation panel must be tested before leaving the mounting area in order to validate all electrical connections. Since the advent of GreyLogix Brazil, the automation panel connection testing has always been performed by two employees, who are shared with another area of the company. These employees are responsible for checking the card connections to the farthest point of the panel to validate the accuracy of all connections. Generally, while an employee forces a signal, or another checks for the presence of that signal at the other end, so peer to peer.

This document details a solution for the best process by creating generic automation testing software that requires only one employee to test any automation panel. The execution of this test will go through a review of possible technologies, seeking to make the process more productive and more agile.

The motivation was to reduce the staff required for testing, to prevent blurring from another area of the company,to reduce the time tken to interprett the test results, to visualize what the software itself uses to interpret and gain time in execution of the test.

**Key-words**: FAT. Factory Acceptance Test. Automation panel. Automation panel test.

# Lista de ilustrações

# Sumário

# 1 INTRODUÇÃO

Todo painel de automação, antes de ser enviado para o cliente, recebe o TAF (Teste de Aceitação de Fábrica) conforme a norma IEC-62381, para avaliar de acordo com a norma, se todos os pré-requisitos de montagem foram seguidos. O TAF tem como finalidade validar o projeto em relação à especificação. Caso alguma inconformidade seja encontrada durante os testes, deve ser tomada uma ação para corrigi-la. O equipamento só poderá ser liberado e entregue quando não existirem mais inconformidades. A instalação, comissionamento e validação garantem que o sistema seja instalado e iniciado de acordo com as especificações propostas.

O processo de produção de um painel de automação é composto por diversas etapas, desde o recebimento do pedido pela equipe comercial, do projeto elétrico e projeto de automação pela equipe técnica, até separação dos itens do estoque e montagem do painel pela equipe de montagem. Para agilidade entre o setor comercial, o setor de elétrica e o setor de automação, é indispensável um fluxo bem definido das etapas e a integração entre *softwares* evitando o retrabalho e a perda de informações.

O presente trabalho trata da validação da integração desenvolvida pela equipe de *software* da GreyLogix e da criação de um protocolo (procedimento-padrão) e *software* para TAF em paineis de automação, visando uma maior segurança nos cumprimentos dos pré-requisitos, um menor quadro de funcionários para execução dos testes e um menor tempo de execução.

## 1.1 As Empresas

- **Bilfinger GreyLogix GmbH**

A empresa de engenharia para a execução do projeto teve origem na alemã Grey-Logix GmbH, que foi fundada no ano de 2000 por Gerd Witzel, Sven Karsten e Lars Malter. Inicialmente especializada em soluções tecnológicas em automação para gestão de tratamento de água, sua gama de serviços foi ampliada a partir de investimentos e aquisições de empresas como Wolfgang Wiezorrek GmbH e Sepa GmbH & Co. KG, passando a oferecer soluções em automação para indústria dos mais diversos segmentos, incluindo: farmacêutico, bebidas, química, papel e celulose, automobilística, óleo e gás, entre outras. Em 2013 a GreyLogix foi adquirida pela Bilfinger Industrial Technologies, empresa especializada em projetos e construção de plantas industriais que emprega mais de 60 mil funcionários e possui faturamento superior a 7,5 bilhões de euros. Atualmente

a Bilfinger GreyLogix conta com mais de 700 funcionários e 23 escritórios em diferentes países da Europa (GLX-ALEMANHA, 2019).

- **GreyLogix Brasil**

A empresa para a execução do projeto, GreyLogix Brasil, foi fundada pelos ex-alunos do curso de Engenharia de Controle e Automação Renato Leal e Rafael Gonçalves no ano de 2007, após ficarem dois anos em estágio na empresa alemã Bilfinger GreyLogix GmbH. Com o modelo de negócio descentralizado proveniente da empresa alemã, a GreyLogix Brasil conta hoje com mais de 90 colaboradores e 7 escritórios em regiões estratégicas de Santa Catarina e do Paraná. Cada escritório possui uma equipe especializada em diferentes segmentos da indústria, nos setores alimentício e de bebidas, papel e celulose, metal mecânica, automobilístico, químico, farmacêutico, água e efluentes, energia e meio ambiente, entregando soluções em diversas disciplinas, como elétrica, instrumentação e controle, engenharia de *software*, instalação e supervisão de montagens e gestão de projetos. A expertise adquirida em 10 anos de história e a constante capacitação de seus colaboradores possibilita a entrega de soluções de alto desempenho para seus clientes (GLX-BRASIL, 2019).

## 1.2 Motivação

Atualmente na GreyLogix Brasil, todo TAF é executado por dois funcionários qualificados que são compartilhados com a equipe de montagem de paineis. Com o auxílio do projeto, os dois funcionários têm como função verificar a conexão dos cartões com o ponto mais distante, para assim validar a conexão. Enquanto um funcionário força um sinal, o outro verifica a existência desse sinal na outra extremidade, assim ponto a ponto, para todas conexões do painel, desse modo, cita-se diversos motivos para realizar a automação dessa função, dentre eles:

- disposição de dois funcionários, causando desfalque na montagem;
- interpretação do projeto, podendo ocorrer erro na leitura;
- produção manual de relatórios;
- demora na execução.

Outro ponto importante é o desenvolvimento do projeto de paineis de automação, que atualmente é executado com base em três *softwares* de engenharia. Pela equipe comercial é utilizado o TIA Selection Tool, pelo equipe de elétrica é utilizado o COMOS e pela equipe de automação é utilizado o TIA Portal. Todos *softwares* serão explicados no

capítulo seguinte. A utilização desses *softwares* sem uma determinada integração pode gerar alguns problemas, dentre eles:

- trabalho repetido entre as áreas;
- comunicação limitada entre as áreas;
- possíveis perdas de informações durante o projeto.

## 1.3 Objetivo Geral

O objetivo desse projeto é a validação entre *softwares* internos e a criação de um *software*, que necessite de apenas um funcionário, para teste de qualquer painel de automação. A execução do TAF passará por uma migração de tecnologias, buscando tornar o processo mais produtivo, mais ágil, com menos funcionários envolvidos e com a geração do relatório final de forma automática.

### 1.3.1 Objetivos Específicos

Para atingir o sucesso na solução como um todo, tem-se como objetivo geral do projeto especificar, modelar e implementar um protocolo e *software* para TAF em paineis de automação.

Como objetivos específicos do trabalho, cita-se:

- aprendizado e aperfeiçoamento no uso de novas ferramentas de engenharia de controle e automação industrial;
- aprendizado das normas de testes de paineis de automação;
- desenvolver a programação do protocolo e *software*;
- dar suporte aos funcionários responsáveis pelo testes de paineis.

## 1.4 Metodologia

A metodologia do trabalho consiste em validar a integração desenvolvida na Grey-Logix para fim de documentação e criar um protocolo e *software* de teste para paineis de automação, com o objetivo de agilizar a execução dos testes de paineis executados na empresa.

O seguinte projeto comporá da validação da integração de dois *softwares* de automação e enfatizará no teste ponto a ponto em paineis de automação.

A validação da integração foi executada nos *softwares* COMOS e TIA Portal, que serão abordados no próximo capítulo. Os *softwares* são da empresa Siemens, mas a integração foi desenvolvida pela equipe da GreyLogix, necessitando apenas a validação completa da integração.

O teste de paineis focará no teste ponto a ponto de cada conexão dos cartões de entradas e saídas, analógicas e digitais de um painel de automação. Para isso, será desenvolvido um *hardware*, composto por dispositivos padrões de automação, que será utilizada quando o painel não dispor de tais itens em sua aplicação, caso contrário, apenas a programação desenvolvida será necessária para o teste.

Primeiramente mostra-se os passos da validação da integração dos *softwares*, por meio de várias simulações de importação e exportação de arquivos, sempre verificando a integridade de cada passo, ou seja, nenhuma informação pode ser perdida no processo.

Em seguida inicia-se a programação do *software*, o qual foi criado no ambiente do TIA Portal e basta ser importado para realizar os testes. Caso o painel não possua uma CPU, a empresa contará com um *hardware* de teste, que disporá de uma CPU, cartões I/O com entradas e saídas, digitais e analógicas, módulos de rede e uma IHM para execução dos testes e geração dos protocolos.

# 2 FUNDAMENTAÇÃO E CONCEITOS

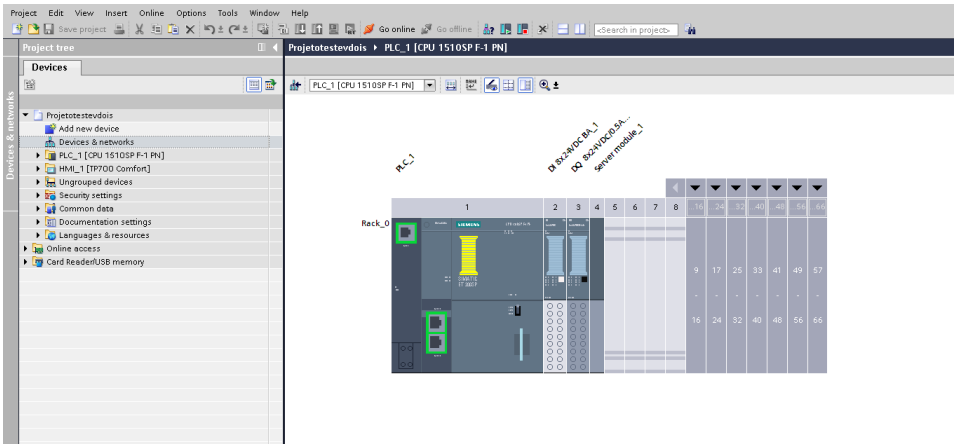
A metodologia utilizada para desenvolver o presente projeto foi baseada nas ferramentas de engenharia utilizadas pela empresa GreyLogix Brasil e todas contribuíram para execução do projeto.

Assim, neste capítulo será explicitado brevemente as principais ferramentas utilizadas. Todas as ferramentas têm em comum o mesmo fabricante, a empresa Siemens, porém cada uma delas foi desenvolvida para atuar em partes específicas do processo de criação de uma solução de engenharia de controle e automação. Por serem da mesma empresa obtém-se um certo grau de integração entre elas, facilitando o desenvolvimento.

## 2.1 TIA Portal

O TIA Portal, *Totally Integrated Automation Portal*, Figura 1, disponibiliza acesso irrestrito a gama completa de serviços de automação digitalizada da Siemens, desde planejamento digital, engenharia integrada até operação transparente. O TIA Portal aumenta a flexibilidade dos construtores de máquinas e operadores da planta através do fluxo de trabalho digital. Com soluções de nuvem flexíveis, o comissionamento virtual e interfaces abertas para maior conectividade permite um trabalha aberto, virtual e em rede. (SIEMENS, 2019a).

Figura 1 – Interface inicial TIA Portal




Fonte: Arquivo pessoal.

O TIA possibilita o desenvolvimento do algoritmo, o projeto de telas IHM, simulação de *hardware* e *software*, criação da arquitetura de redes e declaração da topologia de redes,

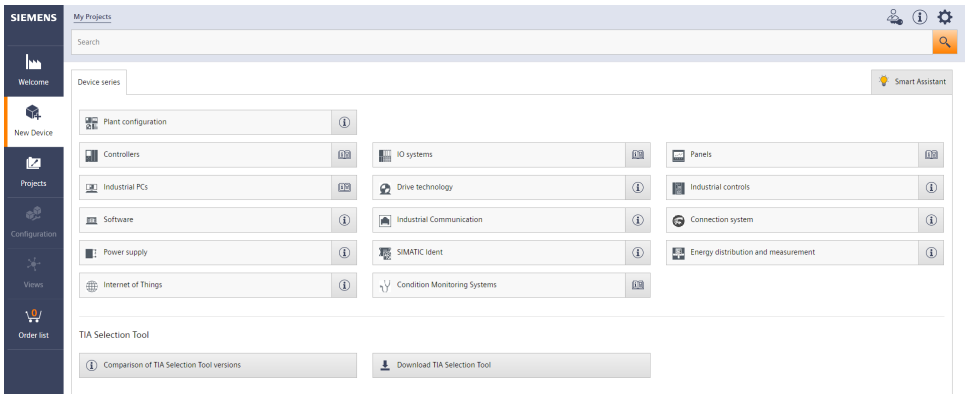
sendo uma ferramenta muito importante na área de automação.

## 2.2 TIA Selection Tool

O TIA Selection Tool, produto da empresa Siemens, serve de apoio para todos os planejadores de projeto, iniciantes e especialistas, pois não é necessário nenhum conhecimento detalhado do portfólio. A ferramenta é disponibilizada na versão *desktop* e também em nuvem (SIEMENS, 2019b).

Na configuração de um novo projeto, sem o mínimo conhecimento é possível montar um *hardware* completo, apenas inserindo o nome ou código de cada equipamento. Na Figura 2, observa-se as diversas opções de *hardware*, como configuração de plantas, controles, PCs industriais, *softwares*, fontes de energia, sistemas IOs, entre outros.

Figura 2 – Interface de novo dispositivo - TIA Selection Tool (Versão nuvem)




Fonte: Arquivo pessoal.

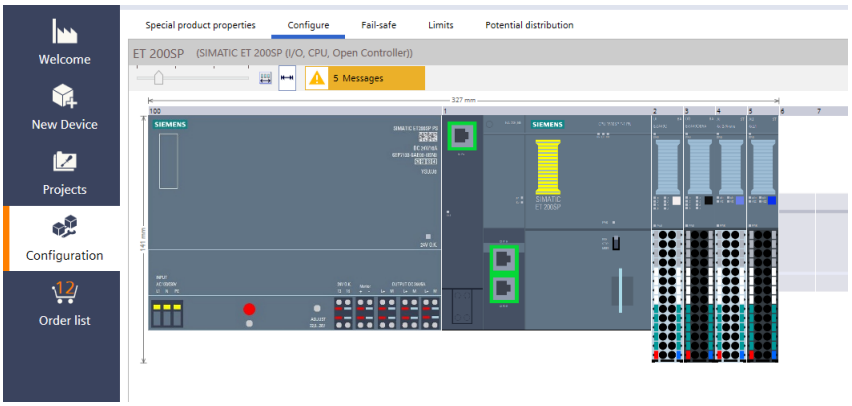
Conforme o TIA Selection Tool (SIEMENS, 2019b), o *software* é disponibilizado com as vantagens de ser rápido, fácil e inteligente.

- **Rápido** pois um projeto pode ser configurado a partir de algumas entradas, sem a necessidade de manual ou algum conhecimento específico. Importação e exportação simplificada da configuração de *hardware* para o TIA Portal ou outros sistemas e apresenta uma visualização ideal dos projetos a serem configurados.
- **Fácil** pois oferece as opções de *download* da ferramenta na versão *offline* para *desktop* e na nuvem com base na *web*. A Siemens garante que a ferramenta está tecnicamente sempre atualizada sobre o portfólio de produtos, como também as abordagens inovadoras e que o trabalho em equipe é altamente flexível e seguro.

- **Inteligente** por oferecer um assistente de seleção para configuração e pedidos sem erros, sendo que as configurações podem ser testadas e simuladas com antecedência. A ferramenta dispõe também de uma biblioteca para arquivar configurações.

Na Figura 3, observa-se o exemplo de um pequeno projeto pronto, composto por uma fonte de energia à esquerda, uma CPU no centro e quatro sistemas IO na direita, entrada digital, saída digital, entrada analógica e saída analógica, respectivamente. Nota-se então que a partir do nome ou código do dispositivo, é possível montar e visualizar o projeto, servindo também como um aprendizado e familiarização com os equipamentos disponíveis.

Figura 3 – Configuração de projeto - TIA Selection Tool (Versão *desktop*)




Fonte: Arquivo pessoal.

## 2.3 COMOS

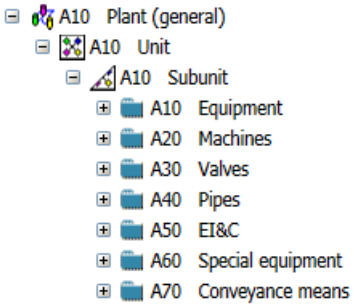
O COMOS, produto da empresa Siemens, é um sistema que contém componentes CAE (Computer Assisted Engineering), CMMS (Computerized Maintenance Management System) e DMS (Document Management System), sendo todo orientado a objetos, o que permite associar o mesmo objeto a diferentes disciplinas, como por exemplo um motor que está em um P&ID e em um diagrama elétrico. Desse modo, alterações realizadas nos dados do objeto são atualizadas em todos os documentos em que ele estiver inserido (SIEMENS, 2019c).

O *software* unifica e indexa grande parte dos documentos industriais como, por exemplo, descritivos de equipamentos, diagramas elétricos, diagrama de processos P&ID, lista de materiais, entre outros. As ferramentas do *software* compõem basicamente todas etapas de um processo, como projeto, montagem, manutenção e gerenciamento da fábrica. Para estruturação das diversas informações o *software* conta com dois grandes conceitos-base, o primeiro conceito é o uso de um banco de dados interno e o segundo está relacionado

como as informações são criadas, usando conceitos de programação orientada a objetos. Para ilustrar a operabilidade do COMOS, pode-se usar como exemplo a avaria de um motor ou qualquer outro equipamento em uma linha de produção. Para localizar as informações do equipamento do modo clássico, faz-se necessário a procura dos documentos dos armários de acionamento do equipamento, o fluxograma do processo, o *datasheet* do equipamento entre outras informações. Utilizando-se o COMOS o processo torna-se mais ágil, visto que para acessar essas informações, basta digitar o nome que foi declarado o equipamento e todas informações relacionadas ao equipamento estarão indexadas junto ao objeto (SIEMENS, 2019c).

Na interface do *software* o projeto é dividido em duas principais visões, a *unit* e a *location*, além de um terceira visão, *base objects*, que relaciona-se ao projeto indiretamente.

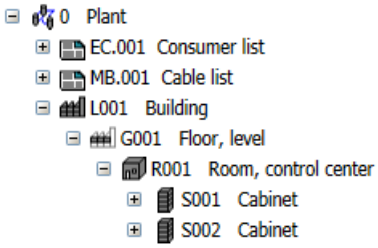
Figura 4 – Unit - COMOS




Fonte: Arquivo pessoal.

Na Figura 4, tem-se a visão da *unit*, onde insere-se as informações de processo. Observa-se que o *software* divide a hierarquia dos processos em Projeto > Planta > Unidade > Subunidade > Pastas divididas em categorias, que são subdivididas em equipamentos, motores, válvulas, malhas de instrumentação e controle, equipamentos de transporte e outros. Nesta aba declara-se os diagramas de P&ID.

Figura 5 – Location - COMOS

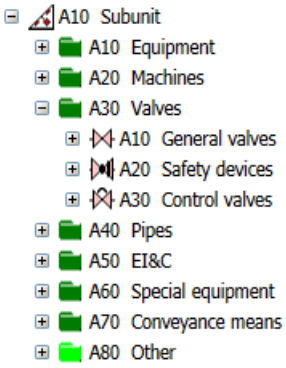



Fonte: Arquivo pessoal.

Na Figura 5, tem-se a visão de *location*, onde insere-se os diagramas elétricos e de automação. Nessa aba a hierarquia é dividida em Projeto > Planta > Prédio > Andar > Sala > Painel, ainda sendo possível a criação de mais um nível, uma gaveta para o painel. Percebe-se que essa aba está mais atrelada com a parte física, tanto que a hierarquia é separada em prédio, andar, sala, etc.

Na visão *documentations*, segue-se a mesma lógica das abas *location* e *unit*, porém para armazenar-se toda a documentação. E na visão de *base objects*, Figura 6, onde localiza-se a base de dados dos objetos, incluindo os modelos de desenhos, tabelas, modelos de documentação elétrica, de processo e afins.

Figura 6 – Base Objects - COMOS




Fonte: Arquivo pessoal.

## 2.4 SIMATIC WinCC

O WinCC foi projetado de modo a não ser específico para indústria e tecnologia, ele é modular e pode ser estendido de forma flexível, podendo ser usado tanto em aplicações para um único usuário como em soluções complexas para múltiplos usuários ou até sistemas distribuídos incluindo vários servidores e clientes. O WinCC estabeleceu-se como um padrão da indústria no campo de visualização de processo, independente de ser um Sistema SCADA autônomo (Supervisory Control and Data Acquisition) ou componente IHM (Interface Homem-Máquina) de sistemas de controle para gerenciamento de energia. Um grande número de Opcionais e Add-ons para WinCC permitem criar soluções específicas para cada tipo de indústria, por exemplo na indústria farmacêutica ou aplicações de tratamento de água (SIEMENS, 2019d).

Todos os canais de comunicação essenciais para conectar aos controladores, como para AllenBradley Ethernet IP, Modubus TCP/IP e canais de outros fabricantes como PROFIBUS/PROFINET e OPC já estão inclusos na licença WinCC. Devido ao fato de cada fabricante de controlador também oferecer respectivos servidores OPC, praticamente não

existem limites na capacidade de comunicação do WinCC. Também inclui um arquivamento poderoso e escalonável de dados baseado em Microsoft SQL Server, que pode ser usado como um *hub* central de informações. Interfaces abertas e opcionais para efetiva integração com a área de negócios e de TI formam a base do Plant Intelligence, isso também permite conexão com sistemas MES (Manufacturing Execution Systems) e sistemas ERP (Enterprise Resource Planning) (SIEMENS, 2019d).

## 2.5 SIMATIC IHM

A Siemens projetou a tecnologia SIMATIC IHM para atender a processos cada vez mais complexos de suas máquinas e sistemas. Essa ferramenta é otimizada para atender as necessidades de segurança e produtivas, usando interfaces abertas e padronizadas em *hardware* e *software*, que permitem uma integração eficiente em sistemas de automação (SIEMENS, 2019e).

Os SIMATIC IHM são adequados para a integração nas redes PROFINET e PROFIBUS e oferecem interfaces para a conexão de aparelhos periféricos USB. A partir de 7", os dispositivos dispõem de um *switch* de Ethernet com 2 portas, a partir de 15", possui uma interface PROFINET de 1 gigabit.

## 2.6 Arquitetura e topologia de redes

Arquitetura de redes faz referência às camadas e aos protocolos utilizados pela mesma. Para desenvolver-se uma tecnologia, seja *hardware* ou *software*, é necessário que conheça-se a arquitetura para construir corretamente a comunicação do projeto com a camada e o protocolo. A arquitetura apresenta de forma simples, a conexão representativa de todos os *hardwares* e *softwares* envolvidos do projeto.

Segundo (TORRES, 2010), a topologia de rede é a estrutura na qual o meio de rede está conectado aos computadores e outros componentes de uma rede, assim deve-se conter exatamente em qual porta do equipamento o cabo de rede está conectado. A topologia pode ser descrita de forma física ou lógica. A topologia física é a real aparência da rede, enquanto que a lógica descreve-se o fluxo dos dados através da rede. A topologia física representa como as redes estão conectadas e o meio de conexão dos dispositivos de redes. A topologia lógica refere-se à maneira como os sinais agem sobre os meios de rede, ou a maneira como os dados são transmitidos através da rede a partir de um dispositivo para o outro, sem ter em conta a interligação física dos dispositivos.

## 2.7 Linguagens de programação

No desenvolvimento da programação do projeto, usou-se as linguagens SCL e LADDER, que serão brevemente apresentadas nessa seção, sendo exemplificadas com mais detalhes no capítulo seguinte.

- **SCL**

SCL (*Structured Control Language*) é uma linguagem de programação de alto nível que se orienta por Pascal e que possibilita uma programação estruturada. A linguagem corresponde à linguagem de programação ST (*Structured Text*), especificada na Norma DIN EN-61131-3 (IEC 61131-3) (SIEMENS, 2019f).

Com SCL não é necessário criar cada função, pode-se recorrer a módulos previamente elaborados como funções do sistema ou módulos de funções do sistema que existem no sistema operacional do módulo central.

Figura 7 – Instruções da linguagem SCL previamente elaborados

| Program control operations | |
|---|---|
| IF ... THEN ... | Run conditionally |
| IF ... THEN ... ELSE ... | Branch conditionally |
| IF ... THEN ... ELSIF ... | Branch conditionally multiple times |
| CASE ... OF ... | Create multiway branch |
| FOR ... TO ... DO ... | Run in counting loop |
| FOR ... TO ... BY ... DO ... | Run in counting loop with step width |
| WHILE ... DO ... | Run if condition is met |
| REPEAT ... UNTIL ... | Run if condition is not met |
| CONTINUE | Recheck loop condition |
| EXIT | Exit loop immediately |
| GOTO ... | Jump |
| RETURN | Exit block |
| (* *) | Insert a comment section |
| REGION | Organize source code |

Fonte: Arquivo pessoal.

Conforme a Siemens, o SCL é um editor de texto com o qual podem ser editados textos de livre escolha. Sua tarefa principal é a criação de módulos para programas STEP 7. Durante a introdução dos dados ocorre uma verificação básica de sintaxe, que simplifica a programação sem erros. Erros de sintaxe são representados em diferentes cores.

O editor oferece as seguintes possibilidades:

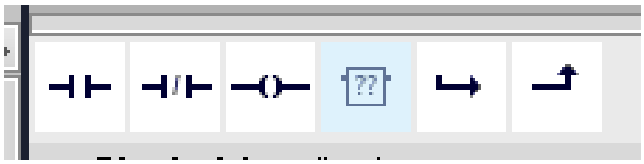
- programação de um módulo na linguagem SCL;
- inserção confortável de elementos de linguagem e chamadas de módulos;

- verificação direta de sintaxe durante a programação;
- ajuste do editor conforme as suas necessidades;
- verificação do módulo concluído mediante compilação;
- indicação de todos os erros e advertências que ocorrem durante a compilação;
- localização dos pontos errados no módulo, opcionalmente com descrição dos erros e indicações para a eliminação dos erros.

- **LADDER**

A linguagem LADDER é amplamente utilizada em todos os campos da automação de sistemas realizada por meio do uso de CLPs, a mesma constitui-se por símbolos gráficos, representando contatos e bobinas (CAETANO, 2018). A principal vantagem de utilizar as lógicas de controle por meio de diagramas LADDER, baseia-se na facilidade com que engenheiros e técnicos de campo desenvolvem "códigos" sem conhecimento anterior de outras lógicas de programação, devido à familiaridade com a lógica a relés, Figura 8.

Figura 8 – Elementos da linguagem LADDER



Fonte: Arquivo pessoal.

Os principais elementos na linguagem LADDER, são:

- Entradas (ou contatos) que podem ler o valor de uma variável booleana;
- Saídas (ou bobinas) que podem escrever o valor de uma variável booleana;
- Blocos funcionais que permitem realizar funções avançadas.

## 2.8 TAF

TAF compreende os serviços de testes realizados em fábrica, destinados a avaliar a troca mensagens entre os protocolos e as demais funcionalidades do sistema (SANTANA, 2008).

A norma IEC-62381 define procedimentos e especificações para o TAF, que são realizados para provar que o sistema de automação está de acordo com a especificação. Segundo a norma, a complexidade dos sistemas de automação vem crescendo cada vez mais, devido ao grande número de sistemas conectados e o uso de novas tecnologias. A experiência demonstrou que o proprietário, o contratado e o fornecedor têm longas e extensas discussões para estabelecer o escopo das atividades e responsabilidades, a fim de alcançar a entrega e aceitação oportunas de sistemas de automação. Esta norma deve levar a uma melhoria e aceleração da fase de negociação e a um entendimento mútuo sobre o escopo das atividades de cada parte (IEC-62381, 2006).

Antes de iniciar o TAF, o fornecedor deve concluir todos os testes internos e os relatórios de teste devem estar disponíveis para inspeção. Todos os documentos relevantes devem ser preparados para uso durante o TAF.

A seguir, mostra-se a lista de documentos normalmente preparados pelo fornecedor, podendo ser adaptada para um projeto específico.

- Documentação do sistema;
- Manuais, folhas de dados do sistema e certificados;
- *Layout* do sistema;
- *Layout* de *hardware*;
- Descrição das interfaces;
- Convenções de lista de I/O e nome de *tag*;
- Impressões gráficas;
- Impressão de configuração;
- Relatórios de testes internos;
- Lista típica de *loops* (*hardware* e *software*) de entregas (*hardware*, *software*, aplicativos e licenças);
- Plano de teste.

O TAF é realizado pelo fornecedor, o comprador deve testemunhar as atividades de teste e será considerado completo quando o fornecedor comprovar com êxito todas as funções necessárias, de acordo com os procedimentos e especificações do TAF. Após a conclusão dos testes, as duas partes de compra e venda devem assinar o certificado.

### 2.8.1 Programação do TAF

Um cronograma de testes deve ser mutuamente acordado entre comprador e fornecedor e deve-se incluir, entre outras, as seguintes atividades:

- **Item 1:** Reunião inicial (revisão de documentos, agendamento etc.);
- **Item 2:** Verificação da documentação do fornecedor;
- **Item 3:** Verificação de estoque de *hardware* e *software*;
- **Item 4:** Inspeção mecânica;
- **Item 5:** Inspeção de fiação e terminação;
- **Item 6:** Teste inicial;
- **Item 7:** Funções gerais do sistema;
- **Item 8:** Visualização/operação;
- **Item 9:** Teste de funcionalidades básicas;
- **Item 10:** Funcionalidade complexa e modos de operação;
- **Item 11:** Teste de interface do subsistema;
- **Item 12:** Retrabalhos TAF;
- **Item 13:** Reunião de encerramento do TAF.

### 2.8.2 Procedimento de teste

O teste é feito para todas as conexões do painel. Na Figura 9, exemplifica-se um teste genérico de apenas uma conexão. Nota-se na imagem o CLP, os cartões de I/O e a conexão, fio na cor azul escuro, entre o cartão e os bornes de saída do painel.

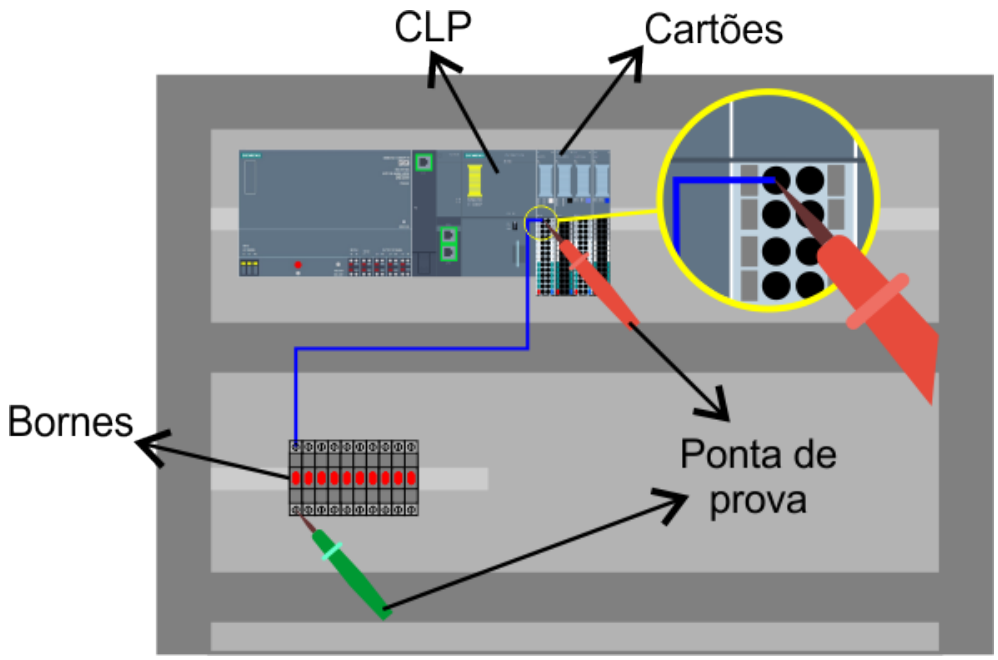
O teste tem como objetivo validar se a conexão do cartão I/O e o borne de saída do painel está bem feita, para isto basta injetar um sinal em uma extremidade e verificar a existência desse sinal na outra extremidade. A injeção e verificação do sinal é demonstrado por meio das pontas de prova.

De acordo com um ou mais dos seguintes cenários típicos, um teste completo deve ser realizado. O cenário comum de teste resume-se a:

- forçar I/O por meio de dispositivos de simulação conectados aos módulos de I/O.

Os cenários listados abaixo dependem dos requisitos de contrato/especificação:

Figura 9 – Esquema de teste




Fonte: Arquivo pessoal.

- forçar I/O por meio de simulação de *software* no nível de processador;
- forçar I/O por meio de simulação de *software* nos módulos I/O;
- forçar I/O por meio de dispositivos de simulação conectados aos terminais de campo.

Para as interfaces de barramento deve-se realizar um teste genérico para cada tipo especificado de dispositivo de campo que seja compatível como padrão relevante. Este ensaio deve abranger a interoperabilidade do sistema de automação e do dispositivo.

- Um segmento deve ser construído e testado com todos os dispositivos associados a ele;
- No caso da funcionalidade de controle distribuído, todos os segmentos envolvidos devem ser testados;
- Sinais relacionados a segmentos não construídos devem ser simulados;
- Todos os documentos relevantes, folhas de dados e figuras devem ser revisados para todos os segmentos.

Segundo a norma, o teste do próprio *link* e dos *loops* selecionados deve ser realizado por meio de um dispositivo de simulação. O valor do sinal é forçado/monitorado na

simulação do dispositivo/sistema de automação. Outros cenários, como os listados abaixo, dependem dos requisitos de contrato/especificação.

- O subsistema é emulado no sistema de automação e os sinais são forçados/monitorados no sistema de automação;
- Subsistema com configuração limitada (apenas dispositivos de processador e *link*) estão disponíveis para verificar a comunicação real e os sinais são simulados no subsistema;
- Subsistema completo, dispositivos de *link* e sistema de automação estão disponíveis, forçando/monitorando I/O no sistema de subsistema/automação.

A maneira de testar deve ser definida para cada subsistema individualmente e após a consideração dos requisitos do projeto.

As atividades de teste podem ser divididas em verificação de recursos do sistema e escopo de projeto relacionado ao fornecimento e aplicação. Inicialmente verifica-se as listas de verificação para o teste de recursos do sistema, de seguinte forma:

- teste inicial;
- funções gerais do sistema, incluindo redundância de *hardware* e verificação de diagnóstico.

Após, confere-se as listas de verificação para o escopo e fornecimento relacionado ao projeto, seguindo a lista abaixo:

- verificação da documentação;
- verificação de estoque de *hardware* e *software*;
- inspeção mecânica;
- inspeção de fiação e terminação.

E por fim, executa-se o teste de funcionalidades de acordo com os documentos listados abaixo:

- P&ID;
- plano de funções;
- narrativa de controle;

- causa e efeitos, listas de intertravamentos;
- diagramas de lógica de função;
- esquemas de controle complexos;
- documentos de interface;
- informações sobre código de cores (cores do fluxo do processo, cores do status do bloco etc.);
- definição das unidades da planta e filosofia de alarme;
- atribuição de unidades da planta às estações de trabalho do operador;
- filosofia operacional (acesso a partir de *displays* gráficos, *displays* de grupo ou paineis frontais);
- definição de coleção de histórico.

Antes do teste orientado ao *loop*, as partes estáticas dos *displays* da IHM devem ser testadas. As seguintes funcionalidades de exibição devem ser verificadas:

- símbolos para embarcações, linhas de processo, válvulas, transmissores, motores, bombas, etc;
- cores para itens estáticos, por exemplo, válvulas manuais, linhas de processo etc;
- a direção e o caminho do fluxo do processo, isto é, as setas da linha de processo;
- a ligação correta dos esquemas de controle de faixa dividida;
- hierarquias e *links* de *displays*;
- as alterações dinâmicas de cores, sub-imagens e pontos de entrada de dados.

Um documento-mestre deve ser identificado antes do TAF para garantir a cobertura completa de todas as *tags*, assim devem ser testadas da seguinte forma: o painel frontal, por exemplo, funcionalidade, texto de serviço, faixa, unidades etc. Deve ser verificado:

- *link* para o nível de I/O;
- exibição do grupo relacionado;
- tendências relacionadas.

Deve verificar-se também se o destino da *tag* está no local correto e se as mudanças de cor para alvos dinâmicos estão de acordo, por exemplo, válvulas, motores, gráficos de barras, etc.

O teste de funcionalidade complexa e intertravamentos deve ser realizado após o teste orientado para as *tags* relacionadas. A simulação dos sinais deve ser realizada de acordo com o cenário escolhido. As funções relacionadas devem ser verificadas de acordo com a especificação de teste orientada para etiquetas. Além do teste relacionado ao aplicativo, recursos do sistema como: recuperação de falha, redundância e modos alternativos de operação devem ser verificados.

Além do teste relacionado ao aplicativo, verifica-se também os recursos do sistema, como:

- recuperação de falha;
- redundância;
- estratégia e níveis de *login*;
- estratégia de processamento de alarmes e reconhecimento;
- desempenho garantido do sistema (taxa de atualização etc.).

Toda as retificações e subsequentes verificações devem ser executadas durante o TAF, caso não seja possível, deve ser realizado após, com base em um acordo mútuo. Para a execução do retrabalho segue-se a lista:

- identificação de retrabalho;
- plano de ação/cronograma;
- execução de retrabalho;
- notificação de conclusão.

Após o teste ser concluído com êxito, executa-se os passos de finalização, sendo eles:

- imprimir e assinar os planos de funções testados;
- verificar data e assinatura de todos os documentos gerados durante o TAF;
- revisar a lista de opções;
- documentar o *hardware* e *software* testados;

- documentar a reposição e a carga do sistema.
- fornecer um índice e cópias coloridas de todas as telas gráficas aplicáveis.

## 2.9 CLP

Controlador Lógico Programável (CLP) ou do inglês PLC (*programmable logic controller*) é um dos controladores mais utilizados na indústria. Conceitualmente, CLP é uma ferramenta para comandar e monitorar equipamentos e processos industriais. possuindo um processador com *software* de controle e um *hardware* que suporta operação em ambientes industriais. O *software* possui um sistema operacional essencial para o controle de processos, onde exigem um curto tempo de resposta, prezando pela confiabilidade. Já o *hardware* suporta condições extremas, as quais um computador de uso doméstico não suportaria.

Controlador Lógico Programável segundo a ABNT (ABNT, 2019), é um equipamento eletrônico digital com *hardware* e *software* compatíveis com aplicações industriais. Segundo a NEMA (*National Electrical Manufacturers Association*) (NEMA, 2019), é um aparelho eletrônico digital que utiliza uma memória programável para armazenar internamente instruções e para implementar funções específicas, tais como lógica, sequenciamento, temporização, contagem e aritmética, controlando, por meio de módulos de entradas e saídas, vários tipos de máquinas ou processos. Um CLP é o controlador indicado para lidar com sistemas caracterizados por eventos discretos (SEDs), ou seja, com processos em que as variáveis assumem valores zero ou um (ou variáveis ditas digitais, que só assumem valores dentro de um conjunto finito). Podem ainda lidar com variáveis analógicas definidas por intervalos de valores de corrente ou tensão elétrica. As entradas e/ou saídas digitais são os elementos discretos, as entradas e/ou saídas analógicas são os elementos variáveis entre valores conhecidos de tensão ou corrente.

# 3 DESENVOLVIMENTO

O desenvolvimento será dividido em duas etapas, inicialmente a validação da integração entre os *softwares* COMOS e TIA Portal e posteriormente o detalhamento do projeto de criação do *software* para TAF.

## 3.1 Integração COMOS e TIA Portal

A integração entre os *softwares* COMOS e TIA Portal foi desenvolvida por dois funcionários da empresa, sob demanda da empresa Siemens. A integração foi criada por demanda do solicitante, porém na GreyLogix ainda não havia essa demanda. Como houve o interesse da integração geral dos projetos da empresa, a validação dos *softwares* fez-se necessária e será exemplificada nessa seção.

A empresa utiliza três *softwares* de engenharia que serão objetivos da integração. O *software* COMOS é utilizado pela equipe de elétrica para desenvolvimento dos circuitos elétricos unifilares e trifilares e o *software* TIA Portal é utilizado pela equipe de automação para desenvolvimento da lógica e programação. Assim, para uma integração completa, adicionou-se ao processo a validação da integração dos dois *softwares* junto ao *software* TIA Selecion Tool, utilizado pela equipe comercial para organizar os pedidos, tanto para empresa, como para o cliente. Como os três *softwares* são da empresa Siemens, supõe-se mais útil a validação da integração de todos esses *softwares*.

Inicialmente a proposta de projeto é estruturada no TIA Selection Tool pela equipe comercial em contato com o cliente. Após a confirmação do projeto, o arquivo do TIA Selection Tool deve ser exportado e importado no TIA Portal pela equipe de automação e exportado através do TIA Portal para o COMOS, da equipe de elétrica. Posteriormente a importação e exportação entre os *softwares* COMOS e TIA Portal deve ser de forma ágil e sem perda de informações.

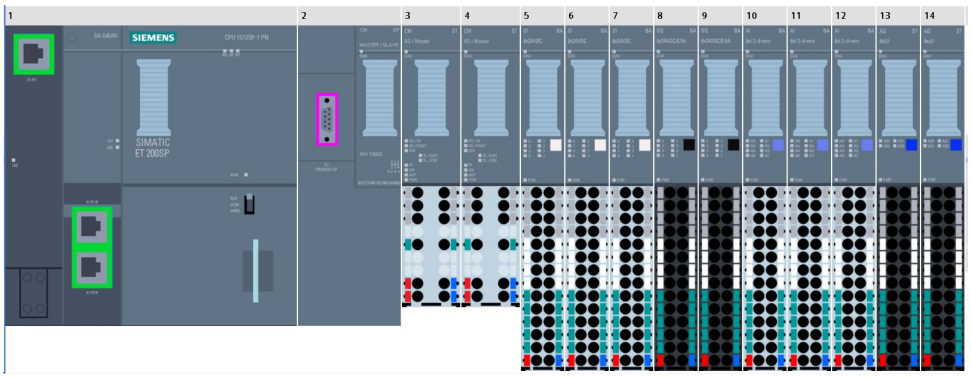
- **Primeira etapa**

A primeira etapa para validação foi a obtenção de um projeto real que a empresa executou, a fim de criá-lo no TIA Selection Tool para simular uma demanda real. Esse projeto foi desenvolvido para uma empresa de café liofilizado e é composto por treze paineis de automação, sendo 4 paineis de remotas e 9 paineis de comando de motores.

O projeto proposto é formado por dois grupos de equipamentos, o grupo 1 composto basicamente por CLPs, remotas e módulos e o grupo 2 composto por remotas, partida de motores e módulos.

A seguir detalha-se todos os equipamentos que compõem o **Grupo 01**. Na Figura 10, visualiza-se os componentes do CLP-01, que servirá de exemplo para todos os itens desse grupo, que são:

Figura 10 – CLP-01 - TIA Selection Tool




Fonte: Arquivo pessoal.

- CLP-01:
  - CPU ET200SP;
  - cartão de memória de 256MB;
  - adaptador de rede 2xRJ45;
  - módulo SIMATIC PROFIBUS DP, mestre DP para até 125 escravos;
  - 2 módulos mestre AS-i ST;
  - 3 módulos digitais de 8 entradas;
  - 2 módulos digitais de 8 saídas;
  - 3 módulos analógicos de 8 entradas;
  - 2 módulos analógicos de 4 saídas.
- CLP-02:
  - CPU ET200SP;
  - cartão de memória de 256MB;
  - adaptador de rede 2xRJ45;
  - módulo SIMATIC PROFIBUS DP, mestre DP para até 125 escravos;
  - 4 módulos digitais de 8 entradas;
  - 3 módulos digitais de 8 saídas;
  - 1 módulo analógico de 8 entradas;

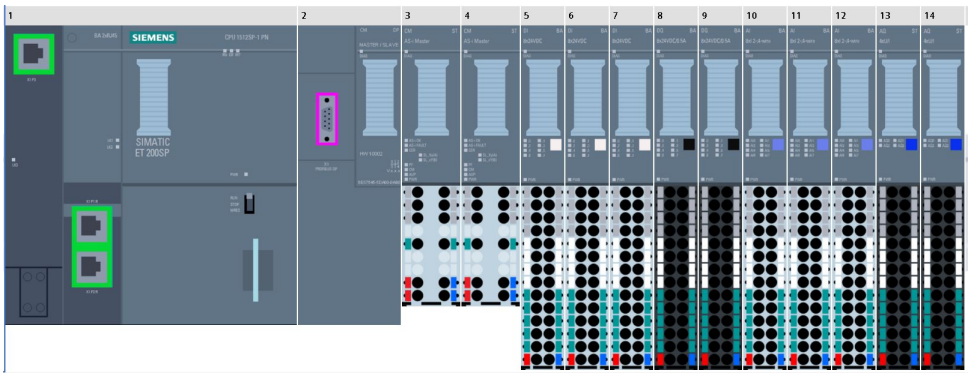
  - 1 módulo analógico de 4 saídas.

- IHM KTP1200 Basic para o CLP-02;
- Remota HVAC:
  - remota ET200SP;
  - adaptador de rede 2xRJ45;
  - 4 módulos digitais de 8 entradas;
  - 2 módulos digitais de 8 saídas;
  - 4 módulos analógicos de 4 entradas;
  - 3 módulos analógicos de 4 saídas.
- Remota Painel-Caixa:
  - remota ET200SP;
  - 1 módulo digital de 8 entradas;
  - 2 módulos digitais de 8 saídas;
- Remota-Enchedeira:
  - remota ET200SP;
  - adaptador de rede 2xRJ45;
  - 6 módulos digitais de 8 entradas;
  - 3 módulos digitais de 8 saídas;
  - 1 módulo analógico de 4 saídas.
- IHM KTP1200 Basic para a Remota-Enchedeira;

O **Grupo 02** é composto exclusivamente por CCM (Centro de Controle de Motores) e assim como no grupo 01, segue uma estrutura com componentes padrões. Na Figura 11, tem-se os componente do *CCM - Motores Embalagem* e a seguir cita-se os componentes dos demais itens do grupo:

- CCM - Motores Embalagem:
  - remota ET200SP;
  - adaptador de rede 2xRJ45;
  - 1 módulo digital de 8 entradas;
  - 1 módulos digital de 8 saídas;

Figura 11 – CCM - Motores Embalagem - TIA Selection Tool




Fonte: Arquivo pessoal.

  - 1 módulo analógico de 8 entradas;
  - 1 módulo analógico de 4 saídas;
  - 6 dispositivos de partida direta de motor;
  - 6 dispositivos de partida com reversão de motor.

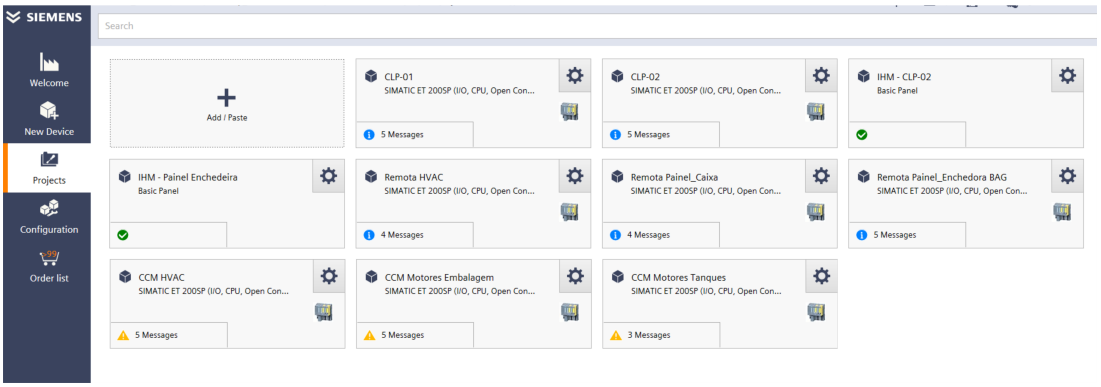
- CCM - Motores Tanque:
  - remota ET200SP;
  - 11 dispositivos de partida direta de motor.

- CCM - HVAC
  - remota ET200SP;
  - adaptador de rede 2xRJ45;
  - 2 módulos digitais de 8 entradas;
  - 2 módulos digitais de 8 saídas;
  - 2 módulos analógicos de 8 entradas;
  - 2 módulos analógicos de 4 saídas;
  - 1 dispositivo de partida direta de motor.

Após criar todo o projeto, tem-se no total os seguintes equipamentos, Figura 12. Todos itens foram validados com a equipe comercial e então exportados para seguir o fluxo. Exportou-se o documento como um arquivo na extensão *.TIA*.

- **Segunda etapa**

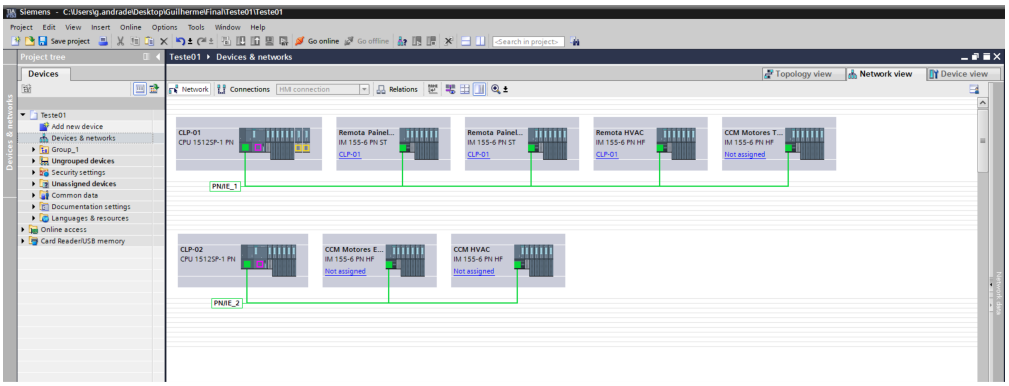
Figura 12 – Equipamentos TIA Selection Tool




Fonte: Arquivo pessoal.

Após todas tratativas comercias e burocráticas de um projeto, faz-se a importação no TIA Portal. Na Figura 13, nota-se que todos os componentes foram importados corretamente, exceto pela comunicação entre os equipamentos, que não é importada, pois não projetou-se no TIA Selection Tool. Assim, o primeiro passo a executar-se no TIA Portal, é a criação da comunicação entre os equipamentos de acordo com as especificações de projeto, visualiza-se essa comunicação na Figura 13.

Figura 13 – 1ª Importação TIA Portal




Fonte: Arquivo pessoal.

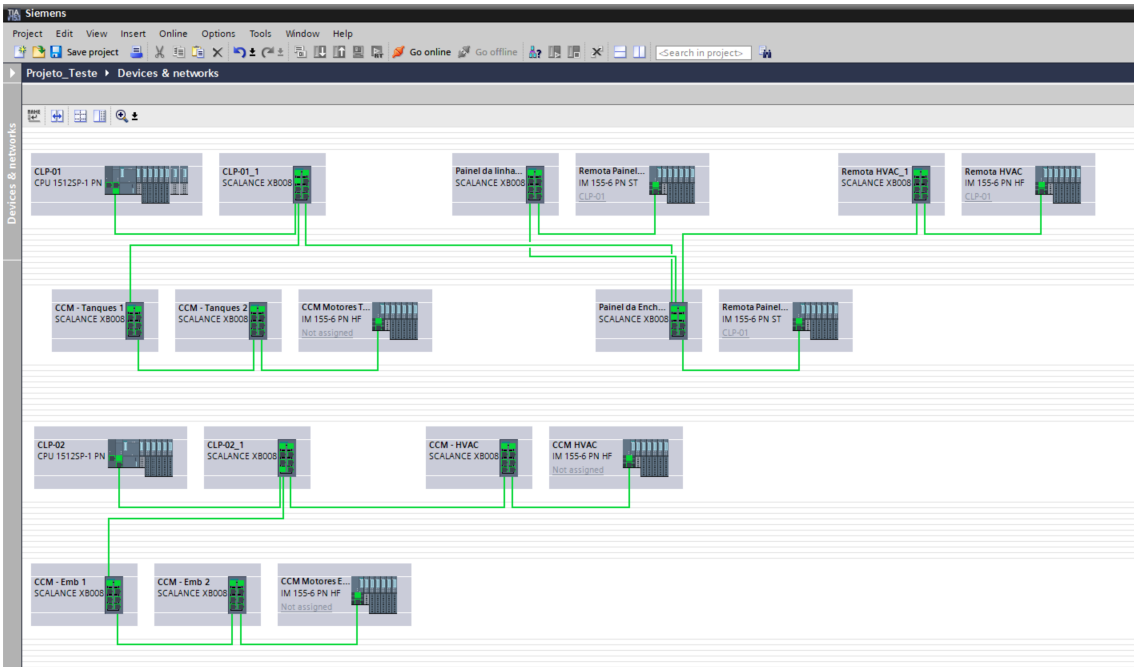
Nessa etapa torna-se possível a visualização de todos equipamentos do projeto, dispensando a partir desse ponto o uso do TIA Selection Tool, utilizado apenas para fechar pedido com o cliente, devido a sua facilidade e agilidade na montagem de paineis. Então qualquer mudança no *hardware* será executada diretamente no TIA Portal.

Nessa mesma etapa desenvolve-se também a arquitetura de redes, tornando a visualização geral do projeto muito mais sistemática e intuitiva. A especificação de uma arquitetura contém informações suficientes para permitir que o implementador desenvolva o

programa ou construa o *hardware* de cada camada, de forma que ela obedeça corretamente ao protocolo adequado. Nesse projeto conectou-se Remota Painel da Enchedora, Remota Painel de Caixa, Remota HVAC, CCM Motores dos Tanques ao CLP-01 e CCM HVAC e CCM Motores Embalagem ao CLP-02.

Logo após, desenvolve-se a topologia de rede, que vai além das conexões entre equipamentos, especificando-se a conexão de cada porta de rede de cada equipamento. Na Figura 14, observa-se a topologia de rede desse caso, já com especificação de alguns módulos SCALANCE, que são *switches* de rede para comunicação industrial, utilizados na lógica da topologia.

Figura 14 – Topologia de rede - TIA Portal



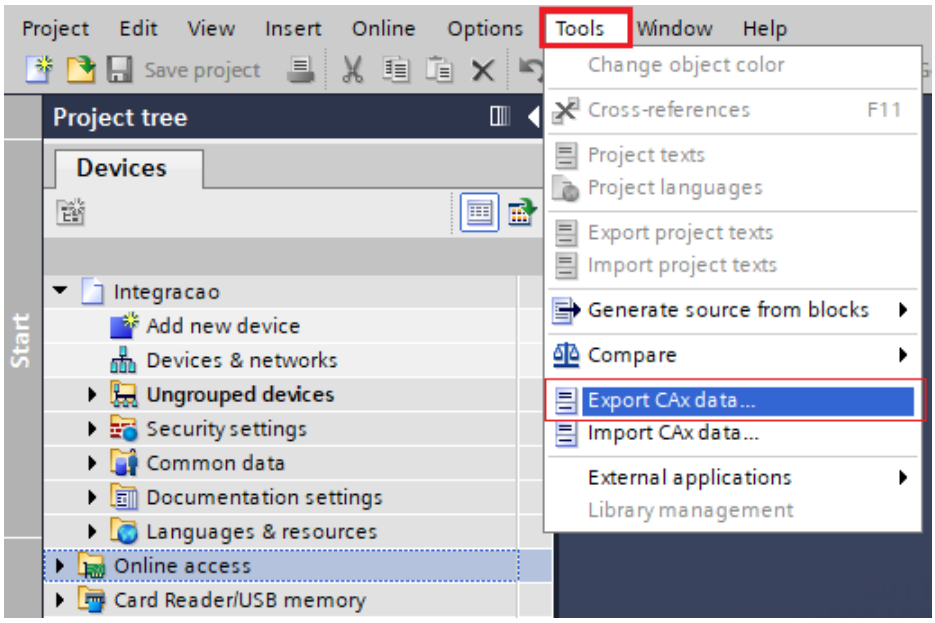

Fonte: Arquivo pessoal.

Após importar o arquivo e configurar a rede, a equipe de automação já está apta a iniciar a criação da lógica de programação de acordo com o que foi solicitado. Mas antes, deve-se exportar o programa para ser importado no COMOS, para todo desenvolvimento da parte elétrica e de acionamentos do projeto.

A exportação deve ser feita no formato *CAx*, extensão utilizada pela GreyLogix para integrar os *softwares*. De acordo com a Figura 15, para executar a função no TIA Portal, acessa-se a aba **Tool**, no menu principal no cabeçalho do *software* e seleciona-se a opção **Export Cax data**.

- **Terceira etapa**

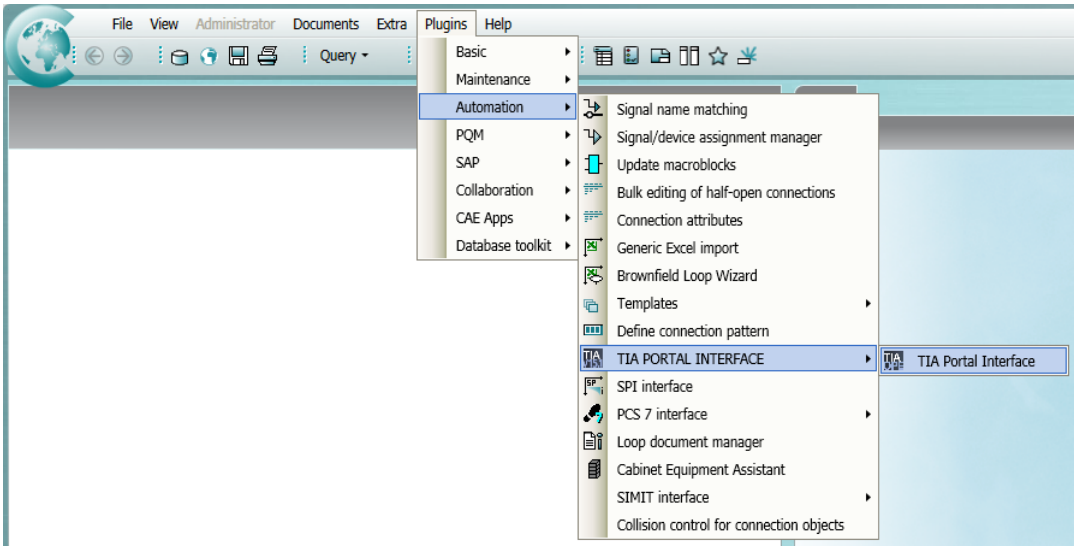
Figura 15 – Formato de exportação TIA Portal




Fonte: Arquivo pessoal.

Após o projeto ser exportado no TIA Portal, o arquivo deve ser importado no COMOS, assim fechando o primeiro ciclo de integração. No COMOS, onde criou-se o projeto de integração, fazendo-o ser compatível com o TIA Portal. A seguir será explicado o processo de importação.

Figura 16 – Formato de importação COMOS



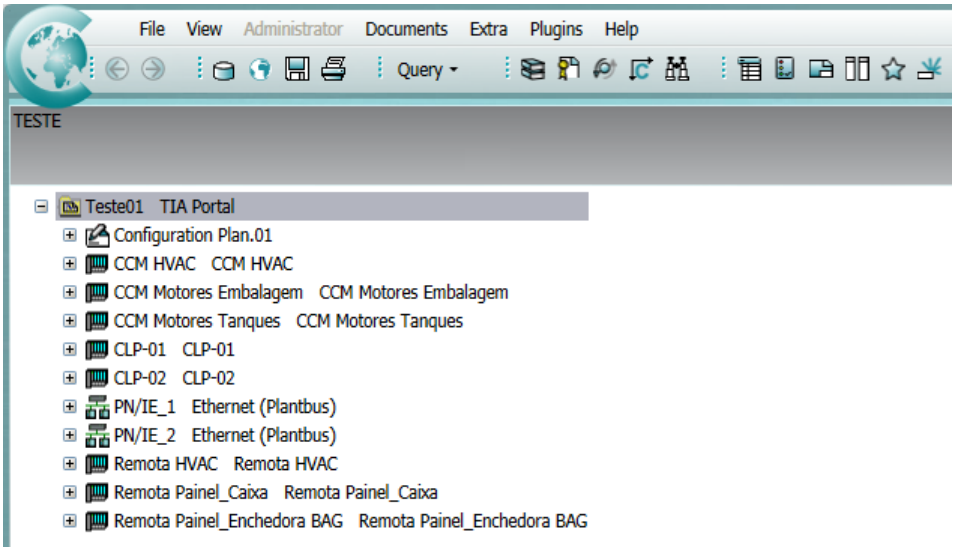

Fonte: Arquivo pessoal.

Inicialmente no COMOS, de acordo com a Figura 16, para importar o projeto,

acessa-se a aba **Plugins**, no menu principal no cabeçalho do *software*, seleciona-se a aba **Automation**, posteriormente **TIA PORTAL INTERFACE** e por fim a opção **TIA Portal Interface**.

Na Figura 17, observa-se a importação no COMOS e valida-se a importação de todos os equipamentos do projeto, incluindo as conexões de rede, observando-as como sendo **PN/IE_1** e **PN/IE_2**.

Figura 17 – Arquivos importados no COMOS




Fonte: Arquivo pessoal.

A partir dessa etapa pode-se criar qualquer outro equipamento ou conexão de rede respeitando as camadas atuais e posteriormente exportar no COMOS e importar novamente no TIA Portal.

- **Quarta etapa**

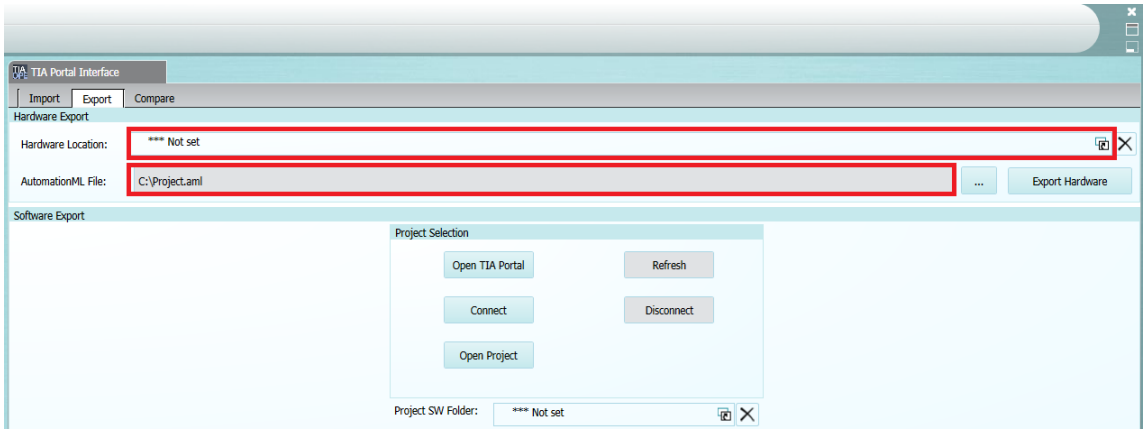
Essa etapa é caracterizada como a última, pois qualquer importação ou exportação executada entre os três *softwares*, já terá sido explicitada.

O projeto elétrico final sairá do COMOS e o projeto de programação final sairá do TIA Portal, então não será todos os casos que a exportação do COMOS será útil. A exportação do COMOS torna-se útil quando alguém da equipe de elétrica observa alguma incongruência no projeto, assim adiciona-se, altera-se ou exclui-se tais incongruências e faz-se a exportação comunicando a equipe de automação.

Para executar a exportação do COMOS, segue-se a Figura 16, porém ao contrário da importação, seleciona-se a aba **Export**. Assim, como exposto na Figura 18, insere-se no primeiro campo, grifado em vermelho, a localização do *hardware*, basta arrastar o bloco

da localização, insere-se no segundo campo a pasta destino do arquivo e clica-se no botão **Export hardware**.

Figura 18 – Formato de exportação COMOS




Fonte: Arquivo pessoal.

A execução da importação no TIA Portal segue como explicitado na Figura 15, porém clica-se em **Import CAx data** e executa-se os mesmos procedimentos explicitados na segunda etapa.

Finalizando com êxito a validação da integração entre os *softwares* COMOS e TIA Portal, criada pela equipe de *software* da GreyLogix.

## 3.2 Software para TAF

Após a validação dos *softwares*, iniciou-se o desenvolvimento do *software* de TAF para paineis de automação.

Primeiramente, iniciou-se o processo de entendimento do que deve-se testar, que no âmbito desse projeto será o teste de cartões de entradas e saídas digitais e entradas e saídas analógicas de um painel. Após, estudou-se sobre os cartões a serem testados, ou seja, como funcionam, sob quais condições estão inseridos e qual a magnitude caso ocorra um erro nesses cartões.

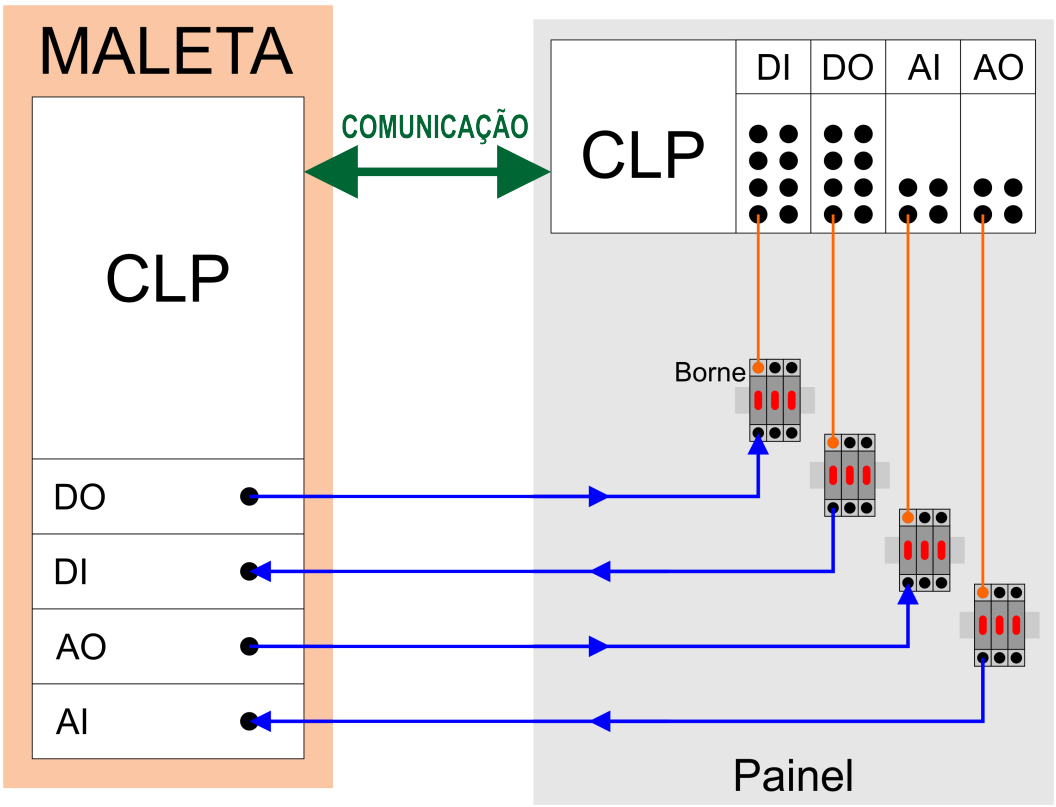
Assim, definiu-se que os cartões de entradas e saídas digitais possuem comercialmente os modelos com 8, 16 e 32 entradas ou saídas e que cada entrada ou saída pode receber uma variável do tipo booleana, ou seja, a variável possui apenas dois estados, 0 ou 1. Utiliza-se comumente esses cartões em aplicações como sensor de presença, onde há apenas dois estados, detectou ou não algo, cita-se também um botão, que está pressionado ou não está pressionado, possuindo também apenas dois estados.

Os cartões de entradas e saídas analógicas possuem comercialmente os modelos com 2, 4 ou 8 entradas ou saídas e têm um funcionamento diferente dos cartões digitais, podendo cada entrada ou saída analógica receber uma variável do tipo inteiro. Utiliza-se comumente esses cartões em aplicações como sensor de temperatura, onde a temperatura varia entre um valor máximo e uma valor mínimo e cita-se também um sensor de nível de um líquido, onde o nível também varia entre um valor máximo e um valor mínimo.

Após o estudo do CLP e participação em várias reuniões com o orientador na empresa, chegou-se à ideia do desenvolvimento de um *hardware* de teste, idealizado inicialmente como uma maleta, que tem como propósito testar qualquer painel de automação, sem nenhuma restrição.

Em um painel, os fios conectam os cartões aos bornes de saída do painel, podendo ser uma conexão direta ou uma conexão que passe por outros equipamentos ou dispositivos. O teste tem como objetivo validar essas conexões, então o que importa para o teste é o último borne ao qual o fio é conectado, pois a partir desse ponto a conexão continuará em campo.

Figura 19 – Esboço do *hardware* de teste




Fonte: Arquivo pessoal.

Na Figura 19, idealiza-se a consolidação do *hardware* de teste. Tem-se à direita uma representação genérica de um painel de automação, que é composto por:

- **CLP**: Pode ser com CPU ou sem CPU (Remota);
- **DI**: Entrada digital, *Digital Input*;
- **DO**: Saída digital, *Digital Output*;
- **AI**: Entrada Analógica, *Analog Input*;
- **AO**: Saída Analógica, *Analog Output*;
- **bornes**: Representando o final da conexão dentro de um painel;
- **conexões em laranja**: Ilustram a conexão entre o cartão e o borne final.

Partindo desse pressuposto de um painel genérico, estrutura-se o *hardware* de teste com os seguintes objetivos:

- **CLP**: CPU para processar o código de testes;
- **DO**: Tem como função gerar um sinal para ser lido pela DI do painel;
- **DI**: Tem como função ler um sinal gerado pela DO do painel;
- **AO**: Tem como função gerar um sinal para ser lido pela AI do painel;
- **DI**: Tem como função ler um sinal gerado pela AO do painel.

Por fim, define-se uma comunicação entre os dois CLPs, fator importante para a associação das CPUs e dos cartões do *hardware* de teste e do painel.

Assim, o *hardware* de teste funcionará da seguinte forma:

- para o teste de DI, o *hardware* de teste forçará um sinal digital através da DO;
- para o teste de DO, o *hardware* de teste esperará um sinal digital através da DI;
- para o teste de AI, o *hardware* de teste forçará um sinal analógico através da AO;
- para o teste de AO, o *hardware* de teste esperará um sinal analógico através da AI.

Definidas as funções mecânicas do *hardware* de teste, fica a cargo da CPU executar o algoritmo necessário para que as saídas e entradas sejam validadas.

### 3.2.1 Algoritmo preliminar

Após definidos os equipamentos do *hardware* de teste e o princípio de funcionamento, inicia-se a parte mais importante do projeto, o algoritmo do *software*.

Por familiaridade e por visualizar um melhor potencial para a solução do problema, escolheu-se SCL como sendo a ferramenta da programação para o *software* de teste.

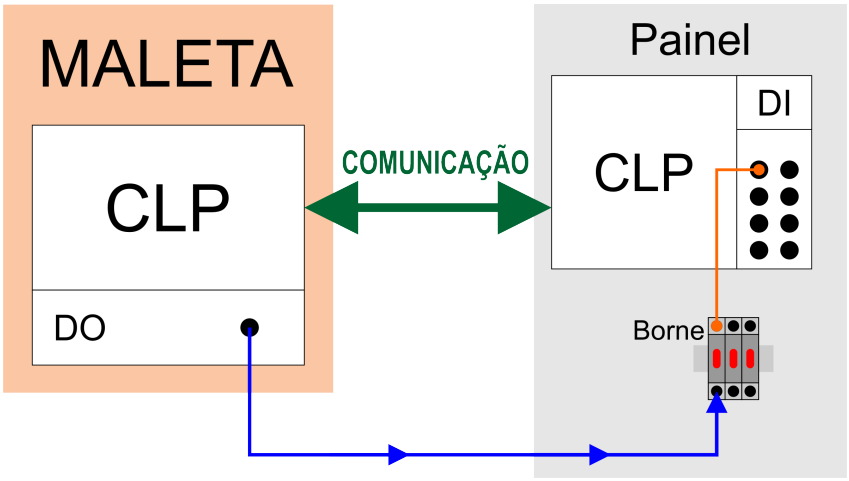
- **Entradas Digitais**

Iniciou-se a definição do algoritmo pelos cartões de entradas digitais, por serem os primeiros cartões convencionalmente alocados na hora da montagem dos equipamentos.

Essa seção limita-se ao objetivo de explicar a estrutura de código, assim, o código consolidado será explicado posteriormente.

Na Figura 20 tem-se um esquema do teste de um cartão DI. Ilustra-se com essa figura o teste da primeira entrada digital do cartão, assim, conecta-se a saída digital do *hardware* de teste ao último ponto da conexão no painel da entrada digital a ser testada. A função do algoritmo será forçar um sinal na saída digital do *hardware* de teste e verificar se o sinal chegou no entrada digital do painel, caso seja verdadeiro, o teste da primeira entrada digital é validado e testa-se a próxima entrada, caso o sinal não chegue, cria-se algumas tratativas para corrigir o problema.

Figura 20 – Esboço do *hardware* de teste - Cartão DI




Fonte: Arquivo pessoal.

Para criação do algoritmo inicial de DI, dividiu-se a lógica em três blocos, o primeiro bloco verifica se está chegando sinal em algumas das entradas do cartão DI, caso esteja chegando energia, executa-se o segundo bloco, que verifica se a conexão está correta, caso

a conexão não esteja correta, o terceiro bloco verifica em qual entrada está chegando sinal. A seguir, explica-se a função de cada bloco:

Na Figura 21 exemplifica-se o código do bloco 1, porém anteriormente deve-se explicar o que significa cada condição.

Figura 21 – Bloco 1 do código preliminar para teste de DI

```
IF "Var_Sequencial" = 0 THEN
    IF NOT "DI00" AND NOT "DI01" AND NOT "DI02" AND NOT "DI03" THEN
        ;// NADA CONECTADO (FUNÇÃO A DIFINIR)
    ELSE
        "Var_Sequencial" := 1;
    END_IF;
END_IF;
```

Fonte: Arquivo pessoal.

A condição **"Var_Sequencial"** tem como objetivo deixar a programação totalmente sequencial, ou seja, nessa figura observa-se que a condição é igual a 0 e só sairá dessa função quando a condição deixar de ser igual a 0, forçando assim um sequenciamento amarrado.

As condições **"DI00"**, **"DI01"**, **"DI02"** e **"DI03"** são as entradas do cartão digital, **"DI00"** sendo a primeira, **"DI01"** sendo a segunda e assim por diante. Esse bloco limita-se até **"DI03"** para uma melhor visualização, mas o código deve abranger todas as entradas do cartão.

Deve-se atentar que a DO do *hardware* de teste está conectada com a DI do painel, assim o algoritmo da Figura 21 têm como função verificar se alguma entrada do cartão DI está recebendo sinal lógico igual a 1. Se alguma entrada estiver recebendo sinal 1, o segundo bloco é chamado, caso contrário, a função é executada novamente até que alguma entrada receba sinal lógico 1.

Na Figura 22 exemplifica-se o código do bloco 2, porém também deve-se explicar o que significa a condição **"Var_Tentativas"**, que tem como objetivo limitar o número de tentativas caso a função não seja verdadeira. Foi estabelecido que tem-se apenas duas tentativas, porém pode-se mudar para quantos for necessário.

Assim, o código da Figura 22 possui a função de verificar se a entrada digital a ser testada está recebendo sinal lógico 1 e se todas as outras entradas estejam recebendo sinal lógico 0. Caso a entrada a ser testada esteja correta, a próxima entrada é testada, caso contrário, chama-se o terceiro bloco.

Caso o bloco 1 denote que alguma entrada possui sinal lógico igual a 1, mas o bloco 2 denote que não é a entrada correta que possui sinal lógico igual a 1, chama-se o bloco 3, que tem como função verificar qual entrada está recebendo sinal lógico igual a 1. Na

Figura 22 – Bloco 2 do código preliminar para teste de DI

```
IF "Var_Sequencial" = 1 THEN
    IF "DI00" AND NOT "DI01" AND NOT "DI02" AND NOT "DI03" THEN
        ;// LIGAÇÃO CORRETA (FUNÇÃO A DEFINIR)
    ELSE
        "Var_Tentativas" := "Var_Tentativas" + 1;
     IF "Var_Tentativas" < 2 THEN
         "Var_Sequencial" := 1;
     ELSE
         "Var_Tentativas" := 0;
         "Var_Sequencial" := 2;
     END_IF;
   END_IF;
END_IF;
```

Fonte: Arquivo pessoal.

Figura 23 explica-se o código e nota-se que todas entradas são testadas a fim de descobrir onde está o erro.

Figura 23 – Bloco 3 do código preliminar para teste de DI

```
IF "Var_Sequencial" = 2 THEN
    IF "DI00" THEN
        "Var_Sequencial" := 1;
        // LIGAÇÃO INCORRETA (FUNÇÃO A DEFINIR)
        // EXIBE ONDE O FIO ESTÁ CONECTADO
    END_IF;
    IF "DI01" THEN
        "Var_Sequencial" := 1;
        // LIGAÇÃO INCORRETA (FUNÇÃO A DEFINIR)
        // EXIBE ONDE O FIO ESTÁ CONECTADO
    END_IF;
END_IF;
```

Fonte: Arquivo pessoal.

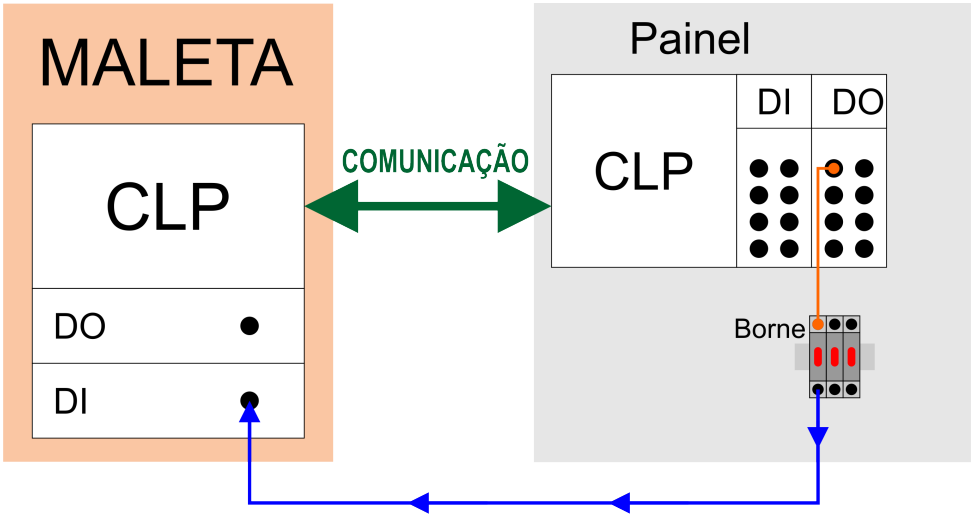
Os três blocos que foram exemplificados compõem o teste da primeira entrada do cartão DI, sendo assim, o teste das outras entradas possui a mesma estrutura, apenas com valores diferentes para as condições.

- **Saídas Digitais**

A segunda definição do algoritmo foram os cartões de saídas digitais e essa seção limita-se ao objetivo de explicar a estrutura de código, assim, o código consolidado será explicado posteriormente.

Na Figura 24 tem-se o esquema do teste de um cartão DO. Ilustra-se com essa figura o teste da primeira saída digital do cartão, assim, conecta-se a entrada digital do *hardware* de teste ao último ponto da conexão no painel da saída digital a ser testada. A

Figura 24 – Esboço do *hardware* de teste - Cartão DO




Fonte: Arquivo pessoal.

função do algoritmo será forçar um sinal na saída digital do painel e verificar se o sinal chegou na entrada digital do *hardware* de teste, caso seja verdadeiro, o teste da primeira saída digital é validado e testa-se a próxima saída, caso o sinal não chegue, cria-se algumas tratativas para corrigir o problema.

Para criação do algoritmo inicial de DO, utilizou-se apenas um bloco, pois ao contrário do teste de DI, não é possível verificar qual saída possui sinal lógico 1, para isso necessitaria que todos saídas digitais estivessem conectadas em todas as entradas digitais. Assim, o teste é composto por apenas um bloco que verifica se a conexão está correta.

Na Figura 25 exemplifica-se o código, porém anteriormente deve-se explicar o que significa cada condição.

A condição **"tag_TESTE_DI"** é a *tag* de teste DI do *hardware* de teste e tem como função verificar o recebimento do sinal. As condições **"DO00"** e **"DO01"** são as saídas do cartão digital, **"DO00"** sendo a primeira, **"DO01"** sendo a segunda e assim por diante. Esse bloco limita-se até **"DO00"** para uma melhor visualização, mas o código deve abranger todas as saídas do cartão.

Esse bloco possui também os valores lógicos TRUE que significa o mesmo que sinal lógico 1 e FALSE que significa o mesmo que sinal lógico 0.

Deve-se atentar que a DI do *hardware* de teste está conectada com a DO do painel, assim o algoritmo da Figura 25 têm como função verificar se na entrada digital do *hardware* de teste está chegando sinal lógico igual a 1, caso contrário, o algoritmo mantém-se nesse bloco até que o sinal lógico 1 seja verificado.

Figura 25 – Código preliminar para teste de DO

```
IF "Var_Sequencial" = 0 THEN
    "D000" := TRUE;
    IF "TAG_TESTE_DI" THEN
        "Var_Sequencial" := 1;
        "D000" := FALSE;
         //FUNÇÃO A DEFINIR
    ELSE
        ;//FUNÇÃO A DEFINIR
    END_IF;
END_IF;
```

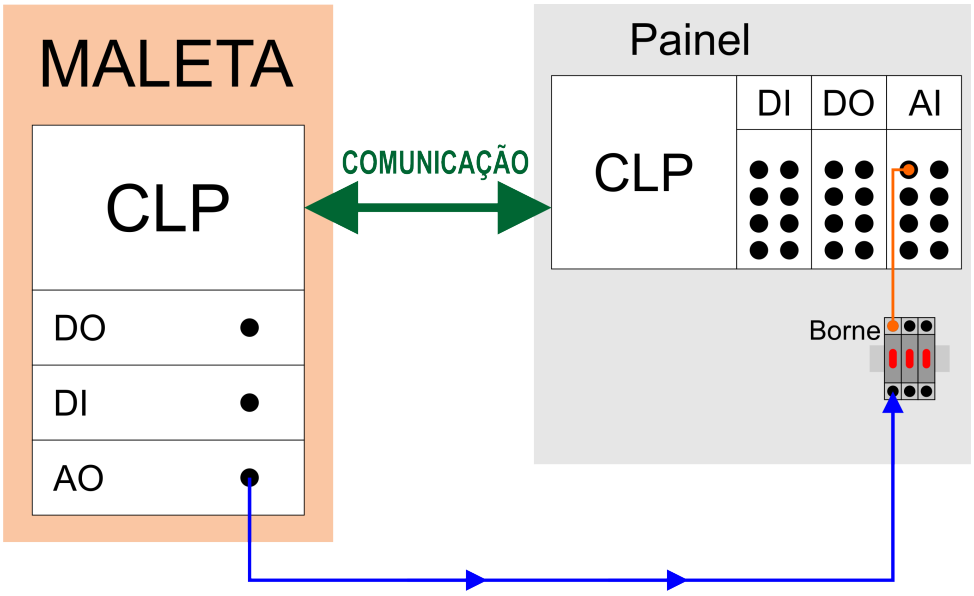
Fonte: Arquivo pessoal.

O bloco que exemplificou-se compõe o teste da primeira saída do cartão DO, sendo assim, o teste das outras saídas possui a mesma estrutura, apenas com valores diferentes para as condições.

- **Entradas Analógicas**

Após a definição do algoritmo para os cartões digitais, iniciou-se o desenvolvimento do algoritmo para as entradas analógicas.

Figura 26 – Esboço do *hardware* de teste - Cartão AI




Fonte: Arquivo pessoal.

Na Figura 26 tem-se um esquema do teste de um cartão AI. Ilustra-se com essa figura o teste da primeira entrada analógica do cartão, assim, conecta-se a saída analógica do *hardware* de teste ao último ponto da conexão no painel da entrada analógica a ser testada. A função do algoritmo será forçar um sinal na saída analógica do *hardware* de teste e verificar se o sinal chegou na entrada analógica do painel, caso seja verdadeiro, o teste da primeira entrada analógica é validado e testa-se a próxima entrada, caso contrário, cria-se algumas tratativas para corrigir o problema.

Para criação do algoritmo inicial de AI, dividiu-se a lógica em três blocos, o primeiro bloco verifica se está chegando sinal em algumas das entradas do cartão AI, caso esteja chegando sinal, executa-se o segundo bloco, que verifica se a conexão está correta, caso a conexão não esteja correta, o terceiro bloco verifica em qual entrada está chegando o sinal. A seguir, explica-se a função de cada bloco:

Na Figura 27 exemplifica-se o código do bloco 1, porém anteriormente deve-se explicar que a as condições **"AI0"** e **"AI1"** são as entradas do cartão analógico. Esse bloco têm como função verificar se ao menos alguma das entradas está recebendo um valor diferente de 0, se sim, o algoritmo continua no bloco 2.

Figura 27 – Bloco 1 do código preliminar para teste de AI

```
IF "Var_Sequencial" = 0 THEN
    IF "AI0" = 0 AND "AI1" = 0 THEN
        ;// NADA CONECTADO (FUNÇÃO A DEFINIR)
    ELSE
        "Var_Sequencial" := 1;
    END_IF;
END_IF;
```

Fonte: Arquivo pessoal.

Na Figura 28 exemplifica-se o código do bloco 2, porém também deve-se explicar o que significa a condição **"Var_Tentativas"**, que tem como objetivo limitar o número de tentativas caso a função não seja verdadeira e a condição **"Var_Erro"**, que tem como objetivo verificar a ocorrência de erros no teste. Deve-se explicar também as variáveis #"0%", #"25%", #"50%", #"75%", #"100%"e #erro. Uma entrada ou saída analógica possui 16 bits, ou seja, o valor varia entre -32767 e 32767, porém utilizou-se apenas entre 0 e 32767, assim, #"0%"é igual ao valor 0, #"100%"igual ao valor 32767 e as outras variáveis são iguais as devidas porcentagens do valor e por fim a variável #erro é igual a 327, ou seja aproximadamente um por cento do valor máximo.

O bloco 2 da Figura 28 têm como função verificar se o sinal está chegando corretamente na entrada analógica, assim, divide-se o sinal enviado em porcentagens e verifica-se

Figura 28 – Bloco 2 do código preliminar para teste de AI

```
IF "Var_Sequencial" = 1 THEN
    IF "AI0" < (#"0%" + #erro) AND "AI0" > (#"0%" - #erro) THEN
        "TAG_TESTE_AO" := #"25%";
    ELSE
        "Var_Erro" := TRUE;
    END_IF;
    IF NOT "Var_Erro" AND "AI0" < (#"25%" + #erro) AND "AI0" > (#"25%" - #erro) THEN
        "TAG_TESTE_AO" := #"50%";
    END_IF;

    IF "Var_Erro" THEN
        "Var_Tentativas" := "Var_Tentativas" + 1;
        IF "Var_Tentativas" < 2 THEN
            "Var_Sequencial" := 1;
        ELSE
            "Var_Tentativas" := 0;
            "Var_Sequencial" := 2;
        END_IF;
    ELSE
        ;//LIGAÇÃO CORRETA (FUNÇÃO A DEFINIR)
    END_IF;
END_IF;
```

Fonte: Arquivo pessoal.

se esse mesmo sinal está chegando corretamente na entrada analógica, caso não esteja, o algoritmo continua no bloco 3. Foi estabelecido que tem-se apenas duas tentativas, porém pode-se mudar para quantos for necessário.

Caso o bloco 1 denote que alguma entrada possui sinal diferente de 0, mas o bloco 2 denote que não é a entrada correta que possui o sinal correto, chama-se o bloco 3, que tem como função verificar qual entrada está recebendo sinal. Na Figura 29 explica-se o código e nota-se que todas entradas são testadas a fim de descobrir onde está o erro.

Figura 29 – Bloco 3 do código preliminar para teste de AI

```
IF "Var_Sequencial" = 2 THEN
    IF "AI0" <> 0 THEN
        "Var_Sequencial" := 1;
        //LIGAÇÃO INCORRETA (FUNÇÃO A DEFINIR)
        // EXIBE ONDE O FIO ESTÁ CONECTADO
    END_IF;
    IF "AI1" <> 0 THEN
        "Var_Sequencial" := 1;
        //LIGAÇÃO INCORRETA (FUNÇÃO A DEFINIR)
        // EXIBE ONDE O FIO ESTÁ CONECTADO
    END_IF;
END_IF;
```

Fonte: Arquivo pessoal.

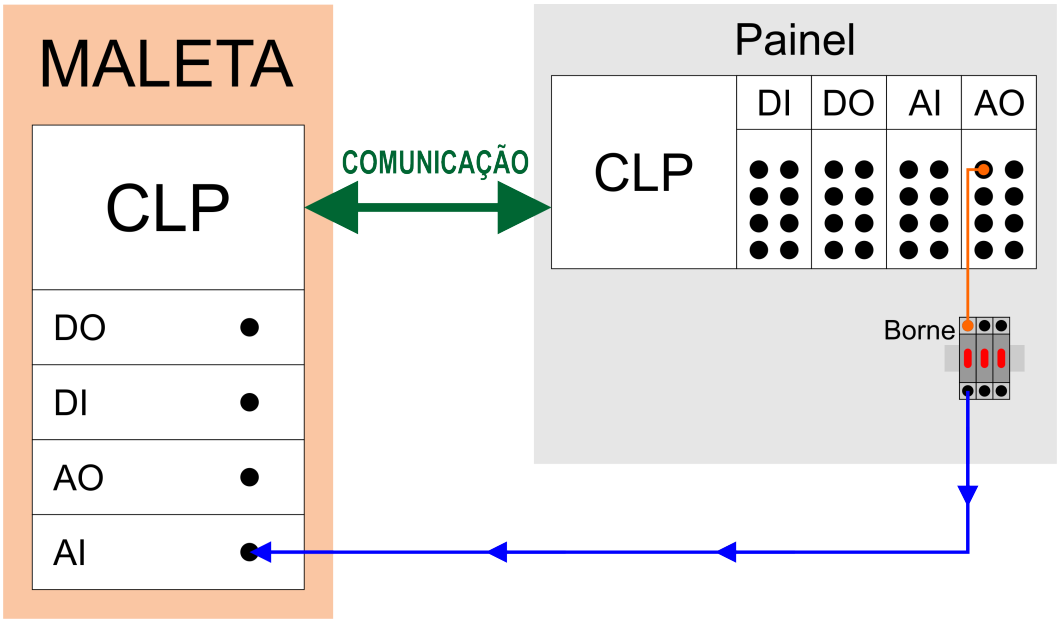
Os três blocos que foram exemplificados compõem o teste da primeira entrada do cartão AI, sendo assim, o teste das outras entradas possui a mesma estrutura, apenas com valores diferentes para as condições.

- **Saídas Analógicas**

A última definição do algoritmo foram os cartões de saídas analógicas e essa seção limita-se ao objetivo de explicar a estrutura de código, assim, o código consolidado será explicado posteriormente.

Na Figura 30 tem-se o esquema do teste de um cartão AO. Ilustra-se com essa figura o teste da primeira saída analógica do cartão, assim, conecta-se a entrada analógica do *hardware* de teste ao último ponto da conexão no painel da saída analógica a ser testada. A função do algoritmo será forçar um sinal na saída analógica do painel e verificar se o sinal chegou na entrada analógica do *hardware* de teste, caso seja verdadeiro, o teste da primeira saída analógica é validado e testa-se a próxima saída, caso contrário, cria-se algumas tratativas para corrigir o problema.

Figura 30 – Esboço do *hardware* de teste - Cartão AO




Fonte: Arquivo pessoal.

Para criação do algoritmo inicial de AO, utilizou-se apenas um bloco. A seguir, na Figura 31 exemplifica-se o código, porém anteriormente deve-se explicar o que significa cada condição.

A condição **"tag_TESTE_AI"** é a *tag* de teste AI do *hardware* de teste e tem como função verificar o recebimento do sinal. As condições **"AO0"** e **"AO1"** são as saídas do cartão analógico, **"AO0"** sendo a primeira, **"A01"** sendo a segunda e assim por diante.

Deve-se atentar que a AI do *hardware* de teste está conectada com a AO do painel, assim o algoritmo da Figura 31 têm como função verificar se na entrada analógica do

Figura 31 – Código preliminar para teste de AO

```
IF "Var_Sequencial" = 0 THEN
    "A00" := 0;
    IF "TAG_TESTE_AI" < (#"0%" + #erro) AND "TAG_TESTE_AI" > (#"0%" - #erro) THEN
        "A00" := #"25%";
    ELSE
        "Var_Erro" := TRUE;
    END_IF;
    IF NOT "Var_Erro" AND "TAG_TESTE_AI" < (#"25%" + #erro) AND "TAG_TESTE_AI" > (#"25%" - #erro) THEN
        "A00" := #"50%";
    ELSE
        "Var_Erro" := TRUE;
    END_IF;

    IF "Var_Erro" THEN
        ;// FUNÇÃO A DEFINIR
    ELSE
        ;// FUNÇÃO A DEFINIR
    END_IF;
END_IF;
```

Fonte: Arquivo pessoal.

*hardware* de teste está chegando sinal enviado, caso contrário, o algoritmo mantém-se nesse bloco até que o sinal validado.

O bloco que exemplificou-se compõe o teste da primeira saída do cartão AO, sendo assim, o teste das outras saídas possui a mesma estrutura, apenas com valores diferentes para as condições.

### 3.2.2 Desenvolvimento das telas da IHM

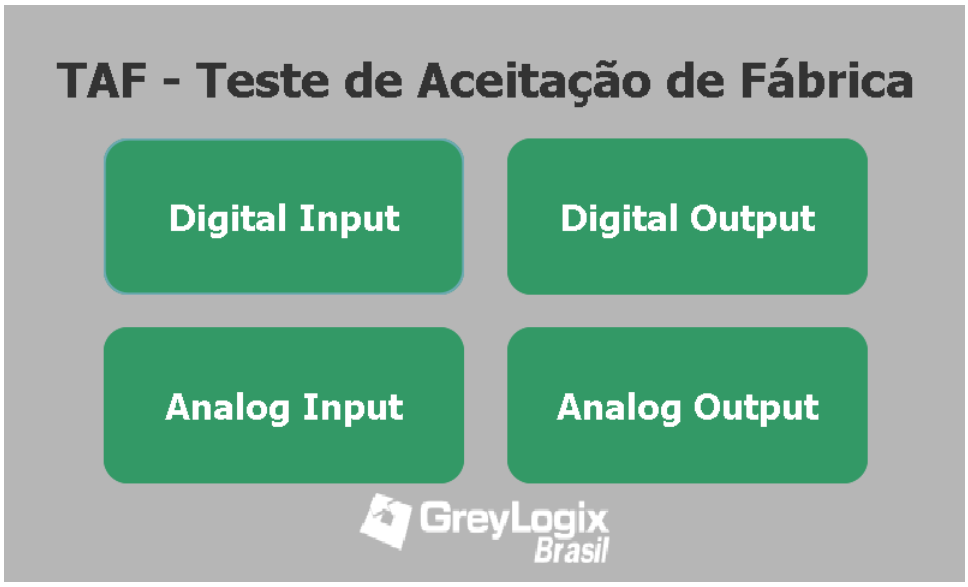
As telas IHM têm como objetivo guiar o testador durante os testes, instruindo-o para o que deve ser conectado e exibindo mensagens de validação ou invalidação dos testes.

Essa seção demonstrará as telas criadas e explicará as funções de cada objeto criado e qual a conexão com o algoritmo. A explicação será no teste de entradas digitais, replicando-se para os demais testes.

Na Figura 32, apresenta-se a primeira tela da IHM, que possui quatro botões referentes aos quatro testes possíveis. A IHM iniciará com essa tela e entre cada teste essa tela volta a ser exibida.

Ao clicar no botão **Digital Input**, que significa entrada digital, determinou-se a ocorrência de alguns eventos. O principal evento atribui o sinal lógico igual a 1 para a variável "Habilita_DI", que tem como função, em conexão com o algoritmo, dar permissão para a execução do bloco de teste de DI. Os outros eventos atribuem o sinal lógico igual a 0 para as variáveis "Habilita_DO", "Habilita_AI" e "Habilita_AO", garantindo que nenhum desses quatro algoritmos executem em paralelo. O último evento é responsável por apenas mudar para a próxima tela.

Figura 32 – Primeira tela da IHM



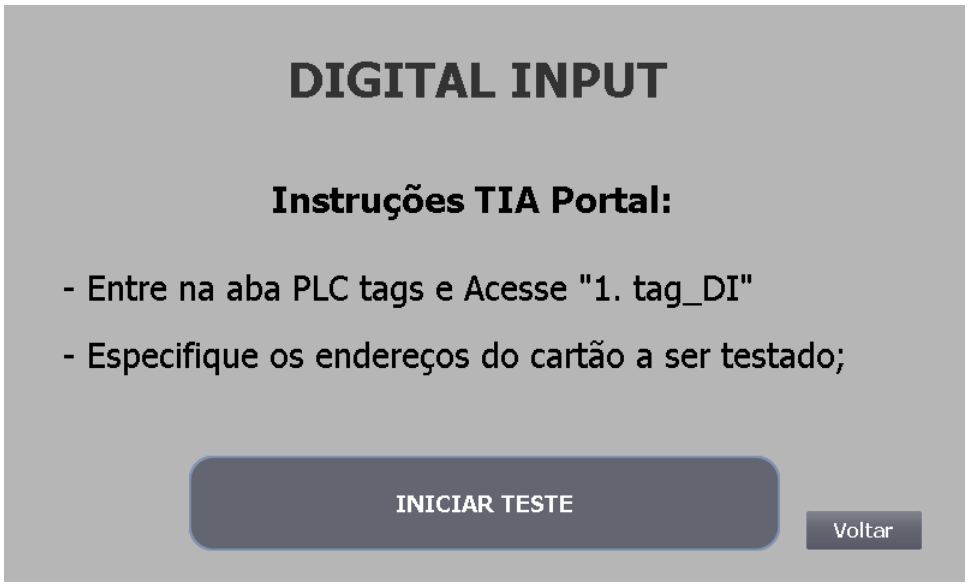

Fonte: Arquivo pessoal.

Na Figura 33, apresenta-se a segunda tela da IHM, que constitui-se de uma instrução e dois botões, um botão para iniciar o teste e outro botão para voltar para a tela anterior. A instrução indica que o testador deve entrar na aba PLC *tags* e acessar "1. tag_DI" e especificar os endereços do cartão que será testado, caso o teste fosse para saídas digitais acessaria-se "2. tag_DO", para entradas analógicas "3. tag_AI" e para saídas analógicas "4. tag_AO". Na Figura 34, mostra-se a tela que deve ser testada e onde encontram-se os endereços, que nesse exemplo vão de I.0 até I2.0.

Ainda na Figura 33, ao clicar no botão **INICIAR TESTE**, determinou-se a ocorrência de apenas um evento, que é a mudança para a próxima tela sem nenhuma conexão com o algoritmo. Ao clicar no botão **Voltar**, determinou-se a ocorrência de dois eventos, o primeiro evento é a atribuição do valor lógico igual a 0 para a variável "Habilita_DI", pois entende-se que clicou-se acidentalmente no botão Digital Input na primeira tela e o segundo evento é a mudança para a tela anterior. Caso estivesse acessado o teste de saídas digitais o primeiro evento seria a atribuição do valor lógico igual a 0 para a variável "Habilita_DO", para entradas analógicas seria a atribuição do valor lógico igual a 0 para a variável "Habilita_AI" e para saídas analógicas seria a atribuição do valor lógico igual a 0 para a variável "Habilita_AO".

Na Figura 35 apresenta-se a terceira tela da IHM para cartões digitais, que constitui-se de três botões, o botão "8" caso o cartão possua 8 entradas ou saídas, o botão "16" caso o cartão possua 16 entradas ou saídas e o botão "32" caso o cartão possua 32 entradas ou saídas. Ao clicar no botão "8" determinou-se a ocorrência de quatro eventos, o primeiro

Figura 33 – Segunda tela da IHM




Fonte: Arquivo pessoal.

Figura 34 – *Tag* de teste DI

1. tag_DI

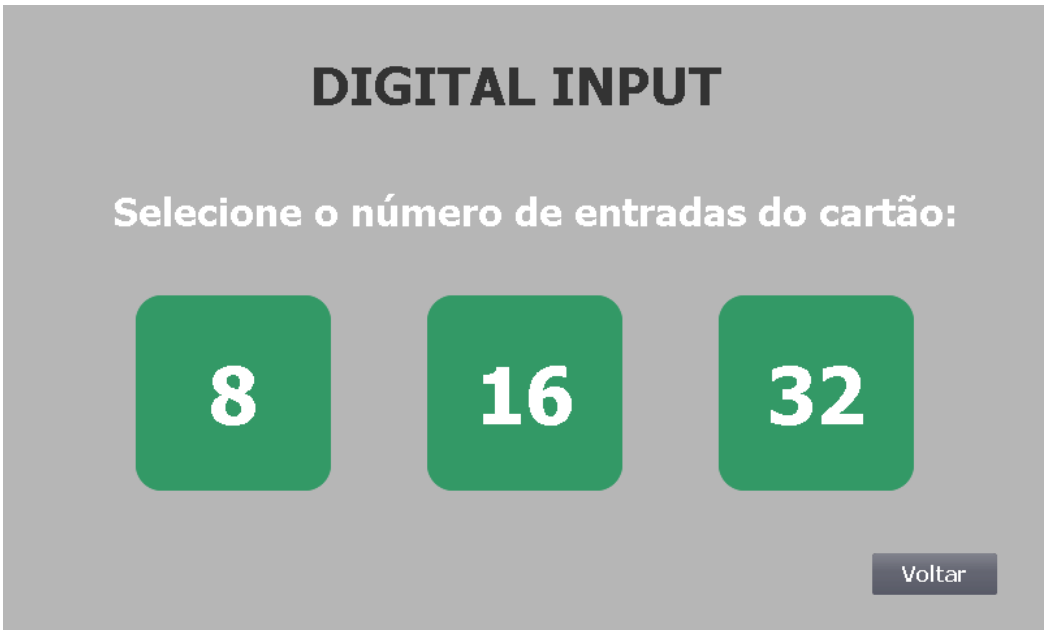
| | | Name | Data type | Address |
|---|---|---|---|---|
| 1 | | DI00 | Bool | %I1.0 |
| 2 | | DI01 | Bool | %I1.1 |
| 3 | | DI02 | Bool | %I1.2 |
| 4 | | DI03 | Bool | %I1.3 |
| 5 | | DI04 | Bool | %I1.4 |
| 6 | | DI05 | Bool | %I1.5 |
| 7 | | DI06 | Bool | %I1.6 |
| 8 | | DI07 | Bool | %I1.7 |
| 9 | | DI08 | Bool | %I2.0 |

Fonte: Arquivo pessoal.

evento é a atribuição do valor lógico igual a 1 para a variável "Dig_8", que tem como função, em conexão com o algoritmo, dar permissão para a execução do bloco de teste do cartão com 8 entradas no bloco de teste de DI, o segundo e terceiro evento é a atribuição do sinal lógico 0 para as variáveis "Dig_16" e "Dig_32", garantindo que nenhum desses três algoritmos executem em paralelo e por fim, o quarto e último evento é a mudança para a tela seguinte.

Ainda na Figura 35, os botões "16" e "32" seguem a mesma estrutura de eventos do botão "8". Para o botão "16" os eventos atribuem o sinal lógico 1 para a variável "Dig_16" e 0 para as variáveis "Dig_8" e "Dig_32". Para o botão "32" os eventos atribuem o sinal lógico 1 para a variável "Dig_32" e 0 para as variáveis "Dig_8" e "Dig_16". O evento de

Figura 35 – Terceira tela da IHM para cartões digtais




Fonte: Arquivo pessoal.

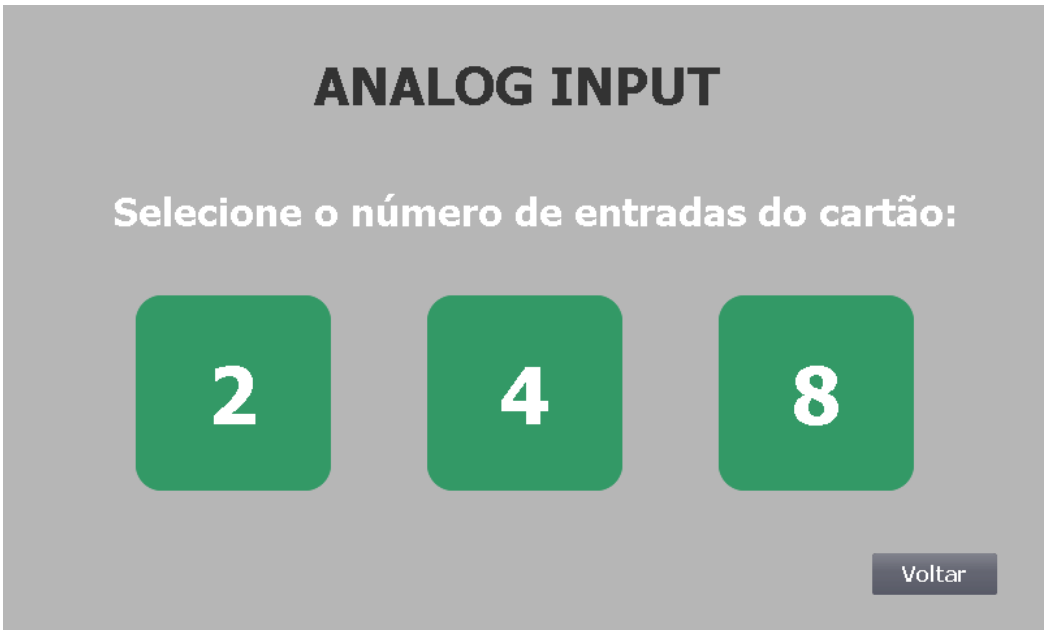
mudança de tela segue igual para os três botões. Ao clicar no botão **Voltar**, determinou-se a ocorrência de quatro eventos, os três primeiros eventos são a atribuição do valor lógico igual a 0 para as variáveis "Dig_8", "Dig_16", "Dig_32" e o quarto evento é a mudança para a tela anterior.

Na Figura 36, apresenta-se a terceira tela da IHM para cartões analógicos, que constitui-se de três botões, o botão "2" caso o cartão possua 2 entradas ou saídas, o botão "4" para 4 entradas ou saídas e o botão "8" para 8 entradas ou saídas. Os botões "2" "4" e "8" seguem a mesma estrutura de eventos da tela da IHM para para cartões digitais. Para o botão "2" os eventos atribuem o sinal lógico 1 para a variável "Ana_2" e 0 para as variáveis "Ana_4" e "Ana_8". Para o botão "4" os eventos atribuem o sinal lógico 1 para a variável "Ana_4" e 0 para as variáveis "Ana_2" e "Ana_8". Para o botão "8" os eventos atribuem o sinal lógico 1 para a variável "Ana_8" e 0 para as variáveis "Ana_2" e "Ana_4". O evento de mudança de tela segue igual ao utilizado na tela da IHM para para cartões digitais.

Ainda na Figura 36, ao clicar no botão **Voltar**, determinou-se a ocorrência de quatro eventos, os três primeiros eventos são a atribuição do valor lógico igual a 0 para as variáveis "Ana_2", "Ana_4", "Ana_8" e o quarto evento é a mudança para a tela anterior.

Na Figura 37, apresenta-se a quarta e última tela da IHM, que será utilizada pelo testador para acompanhar os testes. Essa tela constitui-se de quatro caixas de texto e dois botões, que serão exemplificados a seguir.

Figura 36 – Terceira tela da IHM para cartões analógicos




Fonte: Arquivo pessoal.

A **caixa de texto 1** mostrará a *tag* de teste do *hardware* de teste, definida no algoritmo e terá as seguintes opções:

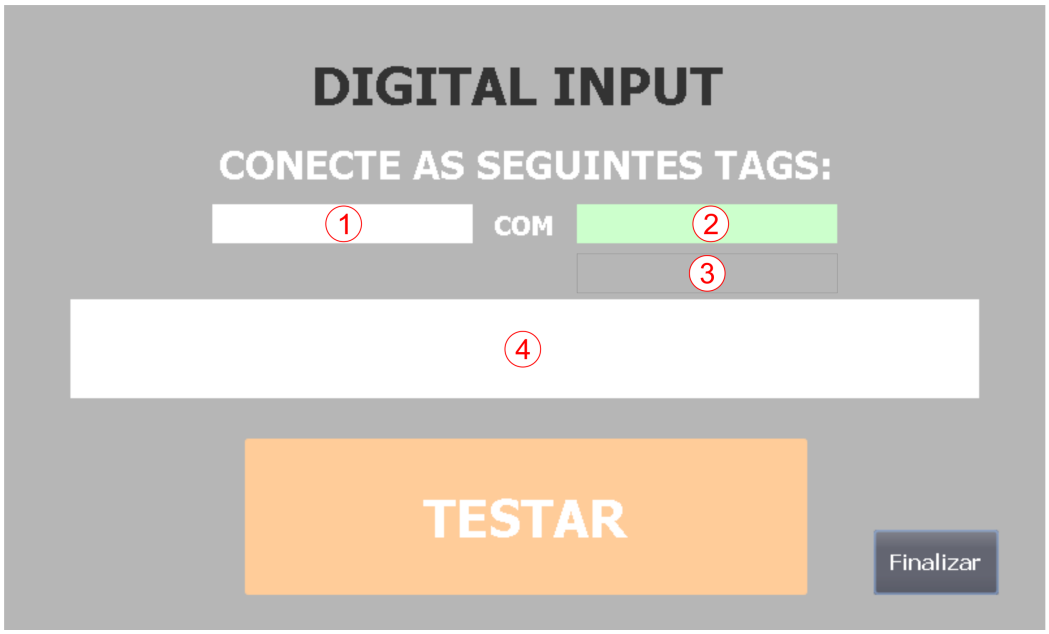
- *tag* de teste AI
- *tag* de teste AO
- *tag* de teste DI
- *tag* de teste DO

A **caixa de texto 2** mostrará a *tag* atual a ser testada no painel, definida no algoritmo e terá as seguintes opções:

- DI00, DI01, DI02 ... quando o teste atual for para o cartão de entradas digitais;
- DO00, DO01, DO02 ... quando o teste atual for para o cartão de saídas digitais;
- AI00, AI01, AI02 ... quando o teste atual for para o cartão de entradas analógicas;
- AO00, AO01, AO02 ... quando o teste atual for para o cartão de saídas analógicas.

A **caixa de texto 3** mostrará a *tag* que está recebendo sinal lógico igual a 1 quando a *tag* atual do teste não estiver recebendo sinal lógico igual a 1. As opções são as mesmas citadas anteriormente para a caixa de texto 2.

Figura 37 – Quarta tela da IHM




Fonte: Arquivo pessoal.

A **caixa de texto 4** mostrará a mensagem sobre a situação atual do teste, definida no algoritmo e terá as seguintes opções:

- **Nada conectado, verifique a conexão**: Quando não estiver chegando o sinal esperado em nenhuma entrada ou saída do cartão;
- **Conexão correta**: Indicando que a conexão está correta e após, nota-se que a mensagem de texto 2 atualizará para a próxima *tag* a ser testada;
- **Conexão incorreta, reconecte as ponteiras**: Indicando que conexão está incorreta e possibilitando a segunda tentativa de teste;
- **Conexão errada**: Quando a conexão incorreta extrapola as duas tentativas e após, nota-se que a mensagem de texto 3 atualizará para a *tag* onde está chegando o sinal lógico igual a 1;
- **Teste executado com sucesso**: Mensagem que aparecerá após executar o teste de todas saídas ou entradas estabelecidas para o teste.

Ainda na Figura 37, ao clicar no botão **TESTAR**, determinou-se a ocorrência de apenas um evento, que é a atribuição do sinal lógico igual a 1 na variável "Botao_OK", que tem como função, em conexão com o algoritmo, dar permissão para se executar determinadas funções. O outro botão, denominado como **Finalizar**, gera dois eventos,

o primeiro evento é a atribuição do sinal lógico igual a 1 na variável "Reset", que tem como função, em conexão com o algoritmo, atribuir a todas as variáveis e condições o sinal lógico igual a 0 e o segundo evento é a mudança para a tela inicial da IHM.

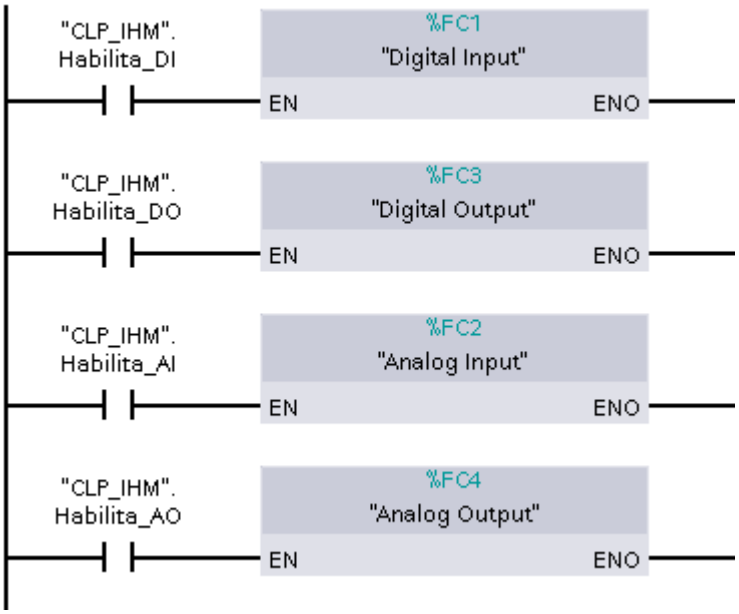
### 3.2.3 Algoritmo consolidado

Após as telas da IHM finalizadas, volta-se para o algoritmo e incrementa-se as variáveis da IHM no código, para que o teste fique completo e funcional. Para comunicação entre algoritmo e IHM criou-se um banco de dados "CLP_IHM", assim, qualquer variável que seja para a comunicação terá esse sufixo.

- **Estrutura principal**

A estrutura principal é desenvolvida no Main, programa que é processado ciclicamente e é a maneira normal pela qual os programas são executados em CLPs. O Main foi programado na linguagem LADDER, por ser uma execução simples e objetiva.

Figura 38 – Main TIA Portal




Fonte: Arquivo pessoal.

Na Figura 38, observa-se os quatro blocos de função correspondentes às quatro modalidades de teste, que serão detalhadas na próxima seção. Observa-se também quatro contatos abertos para acionamento de cada um dos blocos, por exemplo, para acionamento do bloco de entradas digitais, aciona-se o contato fechado "CLP_IHM".Habilita_DI, sendo "CLP_IHM" o banco de dados e Habilita_DI o evento que acionará o contato ao

pressionar o botão **Digital Input** na primeira tela, (como visto na Figura 32). O acionamento dos outros blocos, segue a mesma estrutura do bloco de entradas digitais, ou seja, para executar o bloco de saídas digitais, aciona-se o contato "CLP_IHM".Habilita_DO através da IHM, para executar o bloco de entradas analógicas, aciona-se o contato "CLP_IHM".Habilita_AI através da IHM e para executar o bloco de saídas analógicas, aciona-se o contato "CLP_IHM".Habilita_AO através da IHM.

- **Entradas Digitais**

Assim como no algoritmo preliminar, inicia-se a definição do algoritmo pelos cartões de entradas digitais, por serem os primeiros cartões convencionalmente alocados na hora da montagem dos equipamentos.

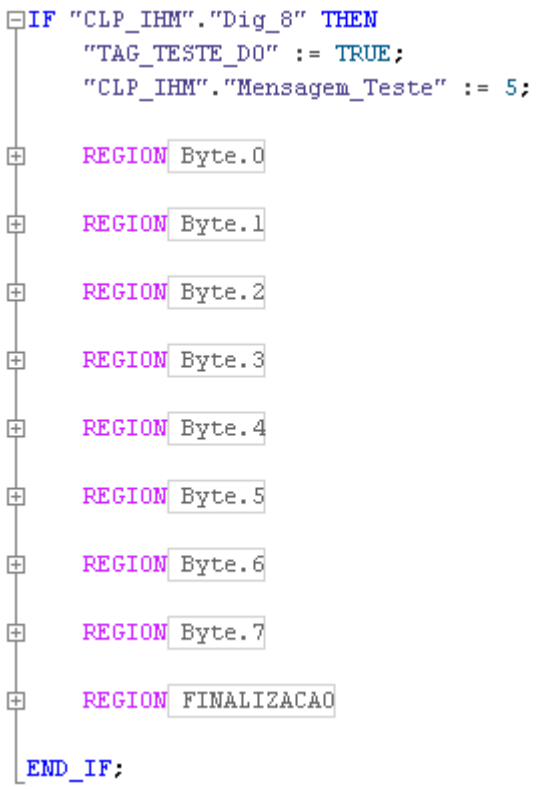
Figura 39 – Bloco de função de entradas digitais

```
// Cartão com 8 DI
⊞IF "CLP_IHM"."Dig_8" THEN ... END_IF;

// Cartão com 16 DI
⊞IF "CLP_IHM"."Dig_16" THEN ... END_IF;

// Cartão com 32 DI
⊞IF "CLP_IHM"."Dig_32" THEN ... END_IF;
```

Fonte: Arquivo pessoal.

Na Figura 39, observa-se o bloco de função principal de entradas digitais, divididas em 8, 16 e 32 entradas. Nota-se que cada um dos IF é executado quando o sinal lógico da condição é igual a 1 e nesse caso a condição é acionada na terceira tela da IHM, (como visto na Figura 35). Assim, executa-se o algoritmo de teste de cartões com 8 entradas digitais quando a variável "CLP_IHM"."Dig_8" possui sinal lógico igual a 1, executa-se o algoritmo para cartões com 16 entradas digitais quando a variável "CLP_IHM"."Dig_16" possui sinal lógico igual a 1 e executa-se o algoritmo para cartões com 32 entradas digitais quando a variável "CLP_IHM"."Dig_32" possui sinal lógico igual a 1. Expandindo-se algumas das três opções possíveis, observa-se com maior clareza a estrutura do algoritmo e as informações que precedem o código de teste de cada entrada.

Figura 40 – Bloco de função de entradas digitais expandido




Fonte: Arquivo pessoal.

Na Figura 40, observa-se a expansão do código de 8 entradas digitais e nota-se a separação em REGION, que traduzindo significa região e é utilizado para organizar melhor o código sem influenciar no mesmo. Cada REGION será explicada a seguir, sendo que "REGION Byte.0" agrega o código composto pelos 3 blocos de programação explicados anteriormente e "REGION FINALIZACAO" agrega a finalização do teste do cartão. Após definidas as informações iniciais, exemplifica-se então a programação consolidada do teste da primeira entrada digital do cartão que está dentro da "REGION Byte.0.

Na Figura 41, observa-se a bloco 1 citado anteriormente, mas agora com a inserção das variáveis de comunicação do algoritmo com a IHM. Como teve-se a abordagem de parte do código no algoritmo preliminar, adicionou-se "CLP_IHM"."Botao_OK", que é a variável acionada quando pressiona-se o botão **TESTAR**, (visto na Figura 37). Caso nenhuma das entradas recebe sinal lógico igual a 0, uma mensagem é exibida na IHM e o bloco inicia novamente.

Caso o bloco 1 identifique que ao menos uma das entradas está recebendo um sinal, executa-se então o bloco 2. Na Figura 42, nota-se que se a conexão está correta, exibe-se uma mensagem na IHM e o código inicia o teste da próxima entrada, caso contrário, exibe-se outra mensagem na IHM e o algoritmo continua no bloco 3.

Figura 41 – Bloco 1 do código consolidado para teste de DI

```
IF "Var_Sequencial" = 0 THEN
    "CLP_IHM"."tag_atual_DI" := 1;
    IF "CLP_IHM"."Botao_OK" THEN
        IF NOT "DI00" AND NOT "DI01" AND NOT "DI02" AND NOT "DI03" THEN
            "CLP_IHM"."Mensagem_Feedback" := 3;
            "CLP_IHM"."Botao_OK" := FALSE;
        ELSE
            "Var_Sequencial" := 1;
        END_IF;
    END_IF;
END_IF;
```

Fonte: Arquivo pessoal.

E por fim, caso o bloco 1 denote que alguma entrada possui sinal lógico igual a 1, mas o bloco 2 denote que não é a entrada correta que possui sinal lógico igual a 1, chama-se o bloco 3, que tem como função verificar qual entrada está recebendo sinal lógico igual a 1. Na Figura 43, nota-se que no bloco 3 adicionou-se uma mensagem na IHM e uma nova variável chamada "CLP_IHM"."tag_trocada_DI" que tem como função identificar e exibir na IHM em qual entrada está conectada a DO do *hardware* de teste.

Figura 42 – Bloco 2 do código consolidado para teste de DI

```
IF "Var_Sequencial" = 1 THEN
    IF "CLP_IHM"."Botao_OK" THEN
        IF "DI00" AND NOT "DI01" AND NOT "DI02" AND NOT "DI03" THEN
            "CLP_IHM"."Mensagem_Feedback" := 1;
            "Var_Sequencial" := 3;
            "CLP_IHM"."tag_trocada_DI" := 0;
            "CLP_IHM"."Botao_OK" := FALSE;
        ELSE
            "Var_Tentativas" := "Var_Tentativas" + 1;
            IF "Var_Tentativas" < 2 THEN
                "CLP_IHM"."Mensagem_Feedback" := 2;
                "CLP_IHM"."Botao_OK" := FALSE;
            ELSE
                "Var_Tentativas" := 0;
                "Var_Sequencial" := 2;
            END_IF;
        END_IF;
    END_IF;
END_IF;
```

Fonte: Arquivo pessoal.

Assim, após testadas todas as entradas, o algoritmo chega em "REGION FINALI-

Figura 43 – Bloco 3 do código consolidado para teste de DI

```
IF "Var_Sequencial" = 2 THEN
    IF "DI00" THEN
        "CLP_IHM"."Mensagem_Feedback" := 4;
        "Var_Sequencial" := 1;
        "CLP_IHM"."tag_trocada_DI" := 1;
    END_IF;
    IF "DI01" THEN
        "CLP_IHM"."Mensagem_Feedback" := 4;
        "Var_Sequencial" := 1;
        "CLP_IHM"."tag_trocada_DI" := 2;
    END_IF;
    "CLP_IHM"."Botao_OK" := FALSE;
    ;
END_IF;
```

Fonte: Arquivo pessoal.

ZACAO" que é o último bloco desse teste e têm como função finalizar o teste. Na Figura 44, nota-se que após finalizar o teste, uma mensagem é exibida na IHM e a variável de teste das 8 entradas digitais é desabilitada, possibilitando que outro teste seja iniciado.

Figura 44 – Código consolidado de finalização para teste de DI

```
IF "Var_Sequencial" = 24 THEN
    "CLP_IHM"."Mensagem_Feedback" := 5;
    "CLP_IHM"."Dig_8" := FALSE;
END_IF;
```

Fonte: Arquivo pessoal.

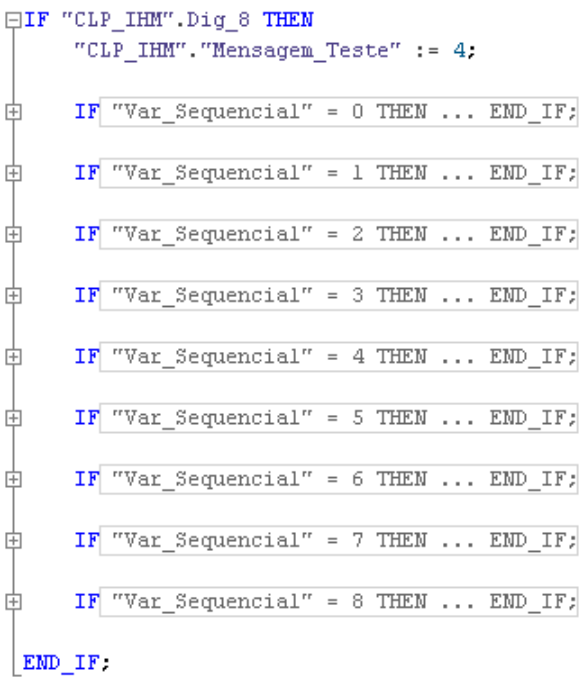
- **Saídas Digitais**

A segunda definição do algoritmo consolidado foram os cartões de saídas digitais, que serão explicados nessa seção.

Na Figura 39, observa-se o bloco de função principal de entradas digitais, que aplica-se ao bloco de função principal de saídas digitais, divididos em 8, 16 e 32 saídas.

Expandindo-se algumas das três opções possíveis, observa-se com maior clareza a estrutura do algoritmo e as informações que precedem o código de teste de cada saída.

Figura 45 – Bloco de função de saídas digitais




Fonte: Arquivo pessoal.

Na Figura 45, observa-se a expansão do código de 8 saídas digitais e nota-se uma informação que precede o código de teste de cada saída, o teste das 8 saídas e o último bloco que denota a finalização do teste.

A informação que precede o código atribui o valor igual a 4 para a variável "CLP_IHM"."Mensagem_Teste" e esse valor corresponderá à mensagem **tag_TESTE_DI** na caixa de texto 1 da Figura 37.

Na Figura 46, observa-se o código citado anteriormente, mas agora com a inserção das variáveis de comunicação do algoritmo com a IHM. Como teve-se a abordagem de parte do código no algoritmo preliminar, será explicado apenas as informações adicionais de comunicação. Assim, nota-se a inserção do botão **TESTAR**, explicado na seção anterior e da mensagem na IHM quando obtém-se êxito no teste e também quando não obtêm-se.

Por fim após executadas todas as entradas do teste, deve-se executar o último bloco da Figura 45, porém a estrutura do código é a mesma do teste de entradas digitais exemplificado na Figura 44, apenas com diferentes variáveis e condições.

- **Entradas Analógicas**

Após faz-se o algoritmo dos cartões de entradas analógicas. Na Figura 47, observa-se o bloco de função principal de entradas analógicas, divididas em 2, 4 e 8 entradas. As

Figura 46 – Código consolidado para teste de DO

```
IF "Var_Sequencial" = 0 THEN
    "CLP_IHM"."tag_atual_DO" := 1;
    "D000" := TRUE;

    IF "CLP_IHM"."Botao_OK" THEN
        IF "TAG_TESTE_DI" THEN
            "Var_Sequencial" := 1;
            "CLP_IHM".Mensagem_Feedback := 1;
            "CLP_IHM"."Botao_OK" := false;
            "D000" := FALSE;
        ELSE
            "CLP_IHM".Mensagem_Feedback := 3;
            "D000" := FALSE;
            "CLP_IHM"."Botao_OK" := false;
        END_IF;
    END_IF;
END_IF;
```

Fonte: Arquivo pessoal.

três opções de códigos possíveis são acionadas na terceira tela da IHM, (como visto na Figura 36).

Figura 47 – Bloco de função de entradas analógicas

```
//Cartão com 2 AO

IF "CLP_IHM".Ana_2 THEN ... END_IF;

//Cartão com 4 AO

IF "CLP_IHM".Ana_4 THEN ... END_IF;

//Cartão com 8 AO

IF "CLP_IHM"."Ana_8" THEN ... END_IF;
```

Fonte: Arquivo pessoal.

Ainda na Figura 47, executa-se o algoritmo de teste de cartões com 2 entradas analógicas quando a variável "CLP_IHM"."Ana_2" possui sinal lógico igual a 1, executa-se o algoritmo para cartões com 4 entradas analógicas quando a variável "CLP_IHM"."Ana_4" possui sinal lógico igual a 1 e o algoritmo para cartões com 8 entradas analógicas quando a variável "CLP_IHM"."Ana_8" possui sinal lógico igual a 1. Expandindo-se algumas das três opções possíveis, observa-se com maior clareza a estrutura do algoritmo e as

informações que precedem o código de teste de cada entrada. Assim como nas entradas digitais, as entradas analógicas são dividas em 3 blocos.

Figura 48 – Bloco 1 do código consolidado para teste de AI

```
IF "Var_Sequencial" = 0 THEN
    "CLP_IHM"."tag_atual_AI" := 1;
    "TAG_TESTE_AO" := #"50%";
    IF "CLP_IHM"."Botao_OK" THEN
        IF "AIO" = 0 AND "AI1" = 0 THEN
            "CLP_IHM"."Mensagem_Feedback" := 3;
            "CLP_IHM"."Botao_OK" := FALSE;
        ELSE
            "Var_Sequencial" := 1;
        END_IF;
    END_IF;
END_IF;
```

Fonte: Arquivo pessoal.

Na Figura 48, observa-se a bloco 1 citado anteriormente, mas agora com a inserção das variáveis de comunicação do algoritmo com a IHM. Adicionou-se a variável "CLP_IHM"."Botao_OK", que é a variável acionada quando pressiona-se o botão **TESTAR**, (visto na Figura 37). Caso nenhuma das entradas recebe sinal igual a 0, uma mensagem é exibida na IHM e o bloco inicia novamente. É declarado uma saída analógica com valor de 50% do valor total de um número inteiro para teste da entrada analógica.

Figura 49 – Bloco 2 do código consolidado para teste de AI

```
IF "Var_Sequencial" = 1 THEN

    IF "CLP_IHM"."Botao_OK" THEN
        "TAG_TESTE_AO" := 0;

        IF "AIO" < (#"0%" + #erro) AND "AIO" > (#"0%" - #erro) THEN
            "TAG_TESTE_AO" := #"25%";
        ELSE
            "Var_Erro" := TRUE;
        END_IF;

        IF NOT "Var_Erro" AN ... THEN ... END_IF;

        IF NOT "Var_Erro" AN ... THEN ... END_IF;

        IF NOT "Var_Erro" AN ... THEN ... END_IF;

        IF NOT "Var_Erro" AN ... THEN ... END_IF;

        IF "Var_Erro" THEN ... END_IF;
    END_IF;
END_IF;
```

Fonte: Arquivo pessoal.

Caso o bloco 1 identifique que ao menos uma das entradas está recebendo um sinal,

executa-se então o bloco 2. Na Figura 49, nota-se que o teste da conexão é executada em vários níveis de entrada, variando a saída analógica em 0%, 25%, 50%, 75% e 100%, caso a conexão esteja correta, exibe-se uma mensagem na IHM e o código inicia o teste da próxima entrada, caso contrário, exibe-se outra mensagem na IHM e o algoritmo continua no bloco 3.

E por fim, caso o bloco 1 denote que alguma entrada possui sinal, mas o bloco 2 denote que não é a entrada correta que possui sinal, chama-se o bloco 3, que tem como função verificar qual entrada está recebendo esse sinal.

Na Figura 50, nota-se que no bloco 3 adicionou-se uma mensagem na IHM e uma nova variável chamada "CLP_IHM"."tag_trocada_AI" que tem como função identificar e exibir na IHM em qual entrada analógica está conectada a AO do *hardware* de teste. Assim, após testadas todas as entradas, o algoritmo no bloco final do teste, uma mensagem é exibida na IHM e a variável de teste das 2 entradas analógicas é desabilitada, possibilitando que outro teste seja iniciado.

Figura 50 – Código consolidado para teste de DO

```
IF "Var_Sequencial" = 2 THEN
    IF "AI0" <> 0 THEN
        "CLP_IHM"."Mensagem_Feedback" := 4;
        "Var_Sequencial" := 1;
        "CLP_IHM"."tag_trocada_AI" := 1;
    END_IF;
    IF "AI1" <> 0 THEN
        "CLP_IHM"."Mensagem_Feedback" := 4;
        "Var_Sequencial" := 1;
        "CLP_IHM"."tag_trocada_AI" := 2;
    END_IF;
    "CLP_IHM"."Botao_OK" := FALSE;
END_IF;
```

Fonte: Arquivo pessoal.

- **Saídas Analógicas**

A última definição do algoritmo consolidado foram os cartões de saídas analógicas, que serão explicados nessa seção. Na Figura 47, observa-se o bloco de função principal de entradas analógicas, que aplica-se ao bloco de função principal de saídas analógicas, divididos em 2, 4 e 8 saídas.

Na Figura 51, observa-se a expansão do código de 2 saídas analógicas e nota-se que o código assemelha-se bastante com o código de entradas analógicas, porém a saída possui apenas um bloco. Ocorreu a inserção de variáveis de comunicação do algoritmo com a IHM, em relação ao código preliminar e como teve-se a abordagem de parte do código no algoritmo preliminar, será explicado apenas as informações adicionais de comunicação.

Assim, nota-se a inserção do botão **TESTAR**, explicado na seção anterior e da mensagem na IHM quando obtém-se êxito no teste e também quando não obtêm-se, assim nota-se o padrão de teste de entradas, focando em porcentagem de sinal.

Figura 51 – Código consolidado para teste de DO

```
IF "Var_Sequencial" = 0 THEN
    "CLP_IHM".tag_atual_A0 := 1;
    IF "CLP_IHM"."Botao_OK" THEN
        "A00" := 0;
        IF "TAG_TESTE_AI" < (#"0%" + #erro) AND "TAG_TESTE_AI" > (#"0%" - #erro) THEN
            "A00" := #"25%";
        ELSE
            "Var_Erro" := TRUE;
        END_IF;
        IF NOT "Var_Erro" AND "TAG_TESTE_AI" < (#"25%" + #erro) AND "TAG_TESTE_AI" > (#"25%" - #erro) THEN
            "A00" := #"50%";
        ELSE
            "Var_Erro" := TRUE;
        END_IF;
        IF NOT "Var_Erro" AN ... THEN ... END_IF;
        IF NOT "Var_Erro" AN ... THEN ... END_IF;
        IF NOT "Var_Erro" AN ... THEN ... END_IF;

        IF "Var_Erro" THEN ... END_IF;
    END_IF;
END_IF;
```

Fonte: Arquivo pessoal.

### 3.2.4 Geração de relatórios

Após finalizada toda a programação e as telas, iniciou-se o algoritmo para geração de relatórios. O relatório têm como objetivo ser apresentado para o cliente e garantindo que o teste foi executado e obteve-se sucesso.

Figura 52 – Quinta tela da IHM


Fonte: Arquivo pessoal.

Para geração de relatório, acrescentou-se mais uma tela para IHM, Figura 52, exemplificando em uma tela de saída digital, porém aplica-se para todos os outros testes. O campo branco central ao ser selecionado abre um teclado, onde o executor do teste deve inserir a *tag* correspondente ao cartão a ser testado. O botão verde gera um relatório e o botão vermelho vai para tela inicial sem a geração do relatório.

Figura 53 – Relatório de TAF

**TAF - TESTE DE ACEITAÇÃO DE FÁBRICA**

SAÍDAS DIGITAIS

CARTÃO:

| TAG | SITUAÇÃO | TAG | SITUAÇÃO |
|---|---|---|---|
| DO00 | **Não Aplicado** | DO16 | **Não Aplicado** |
| DO01 | **Não Aplicado** | DO17 | **Não Aplicado** |
| DO02 | **Não Aplicado** | DO18 | **Não Aplicado** |
| DO03 | **Não Aplicado** | DO19 | **Não Aplicado** |
| DO04 | **Não Aplicado** | DO20 | **Não Aplicado** |
| DO05 | **Não Aplicado** | DO21 | **Não Aplicado** |
| DO06 | **Não Aplicado** | DO22 | **Não Aplicado** |
| DO07 | **Não Aplicado** | DO23 | **Não Aplicado** |
| DO08 | **Não Aplicado** | DO24 | **Não Aplicado** |
| DO09 | **Não Aplicado** | DO25 | **Não Aplicado** |
| DO10 | **Não Aplicado** | DO26 | **Não Aplicado** |
| DO11 | **Não Aplicado** | DO27 | **Não Aplicado** |
| DO12 | **Não Aplicado** | DO28 | **Não Aplicado** |
| DO13 | **Não Aplicado** | DO29 | **Não Aplicado** |
| DO14 | **Não Aplicado** | DO30 | **Não Aplicado** |
| DO15 | **Não Aplicado** | DO31 | **Não Aplicado** |

Fonte: Arquivo pessoal.

Na Figura 53, observa-se como é a estrutura do relatório, que tem como resultado as seguintes situações:

- **Não Aplicado:** O teste não foi aplicado na *tag*;
- **Aprovado:** O teste da *tag* foi executado com sucesso e êxito;
- **Reprovado:** O teste da *tag* foi executado, porém não obteve-se êxito.

Por padrão todos as *tags* iniciam-se com a atribuição de não aprovado, pois usa-se o mesmo *layout* para todas as *tags* passíveis de teste. Na programação inseriu-se uma variável denominada como "CLP_IHM"."Re_00" para a primeira *tag* a ser testada, "CLP_IHM"."Re_01" para a segunda *tag* a ser testada e assim por diante. Inicialmente a variável corresponde a situação de não aprovado, logo no início de cada teste a situação da *tag* é atribuida como reprovado, para caso o teste seja iniciado e não seja finalizado e ao

final do teste de cada *tag* atribui-se a situação de aprovado e reiniciá-se o ciclo na próxima *tag*.

A geração de relatórios de entradas e saídas analógicas segue a mesma estrutura de programação, porém o relatório possui menos *tags*, pois têm apenas 8 *tags* a serem testadas.

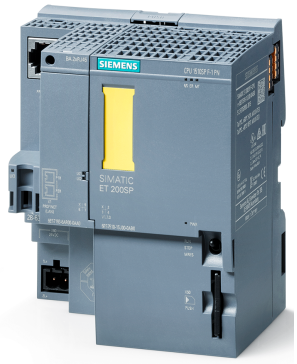
### 3.2.5 Hardware para TAF

Para desenvolvimento do *hardware* de teste, a GreyLogix efetuou a compra dos seguintes materiais necessários para execução do TAF.

- ET 200SP

A CPU utilizada foi uma remota com CPU ET200SP com uma CPU 1510SP F-1 PN, unidade central de processamento com memória de trabalho 150 KB para programa e 750 KB para dados, com um adaptador de rede com mais 2 portas RJ45, Figura 54.

Figura 54 – ET 200SP - CPU 1510SP F-1 PN




Fonte: Disponível em <mall.industry.siemens.com>.

- Fonte SITOP PSU100S

A fonte utilizada foi SITOP PSU100S 24V/10A, denominada como fonte de alimentação estabilizada com entrada de 120/230VCA e saída 24VDC/10A, Figura 55.

- Unidade base para ET 200SP

4 Unidades base para ET 200SP, do tipo A0 com 16 entradas/saídas e 10 conexões auxiliares, Figura 56.

Figura 55 – Fonte SITOP PSU100S


Fonte: Disponível em <mall.industry.siemens.com>.

Figura 56 – Unidade base para ET 200SP

Fonte: Disponível em <mall.industry.siemens.com>.

- Cartão DI

Módulo de entrada digital para ET 200SP, DI 8x24VDC Basic, unidade base do tipo A0, Figura 57.

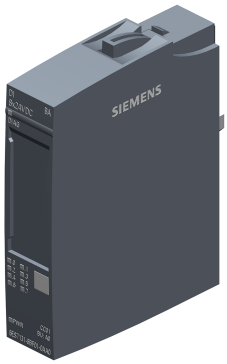
- Cartão DO

Módulo de saída digital para ET 200SP, DQ 8x24VDC Basic, unidade base do tipo A0, imagem muito semelhante ao cartão DI na Figura 57.

- Cartão AI

Módulo de entrada analógica para ET 200SP, AI 4XI 2-/4-fios Padrão, unidade base do tipo A0, imagem muito semelhante ao cartão DI na Figura 57.

Figura 57 – Cartão DI para ET 200SP




Fonte: Disponível em <mall.industry.siemens.com>.

- Cartão AO

Módulo de saída analógica para ET 200SP, AQ 4XU/I Padrão, unidade base do tipo A0, imagem muito semelhando ao cartão DI na Figura 57.

- IHM TP700 Comfort

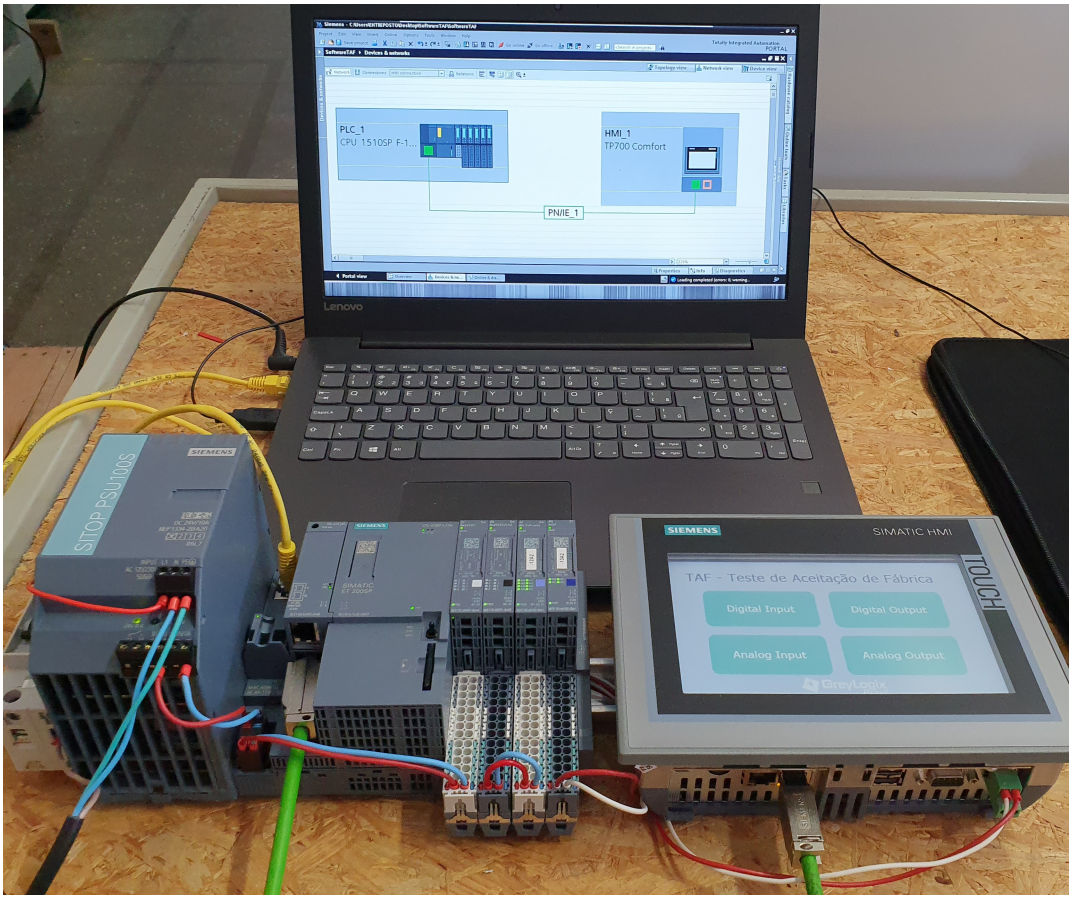
IHM TP700 Comfort, operação por toque, tela TFT widescreen de 7", 16 milhões de cores, interface PROFINET, interface MPI/PROFIBUS DP, 12 MB de memória de configuração e Windows CE 6.0, Figura 58.

Figura 58 – IHM TP700 Comfort


Fonte: Arquivo pessoal.

Assim, após todas alimentações executadas e todas conexões estabelecidas, obteve-se o *hardware* da Figura 59. A fonte é alimentada com uma tensão de 220VCA e alimenta o CLP, os cartões e a IHM com 24VDC. As conexões de rede foram estabelecidas entre CLP e IHM e entre CLP e computador para descarregando da programação e das telas de IHM.

Figura 59 – *Hardware* de teste




Fonte: Arquivo pessoal.

### 3.2.6 Execução completa de um teste

Antes de executar o *software* proposto por esse trabalho, deve-se executar um TAF ponto a ponto nas conexões elétricas do painel, pois para executar o TAF proposto, todos os componentes do painel devem estar energizados.

Na Figura 60, observa-se um documento elaborado para preenchimento do TAF inicial do projeto, que contempla a inspeção mecânica e elétrica do painel. O primeiro quadro aborda um teste visual e o segundo um teste de continuidade, que é executado com auxílio de um multímetro.

Figura 60 – Relatório inicial de TAF

INSPEÇÃO MECÂNICA/ELÉTRICA

| Verificar | Aprovado | Reprovado | Observações |
|---|---|---|---|
| Montagem conforme layout | | | |
| Identificação dos componentes | | | |
| Aterramento do painel | | | |
| Fixação dos cabos | | | |
| Organização dos cabos | | | |
| Identificação dos cabos | | | |
| Plaquetas e etiquetas | | | |

*Marque com um **X** a opção que melhor descreve o teste.

INSPEÇÃO ELÉTRICA DO CIRCUITO

| Verificar | Aprovado | Reprovado | Observações |
|---|---|---|---|
| Teste de continuidade | | | |
| Circuito de comando | | | |
| Circuito de alimentação | | | |
| Circuito de proteção | | | |

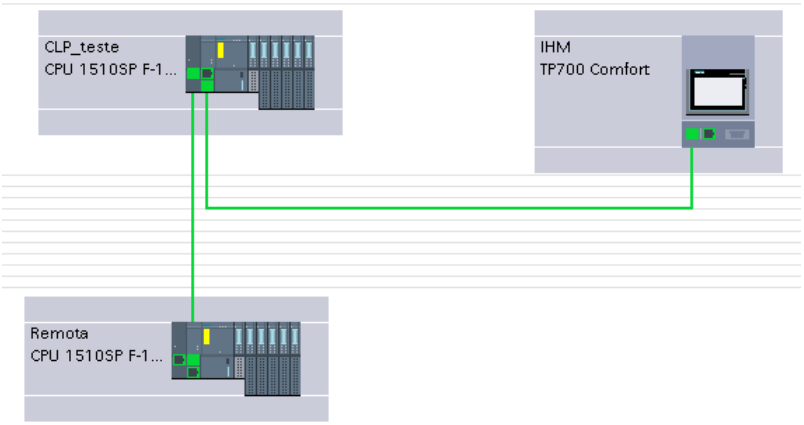
*Marque com um **X** a opção que melhor descreve o teste.

Fonte: Arquivo pessoal.

Após esses testes, inicia-se de fato a execução do *software* desenvolvido. Com o *hardware* montado, o primeiro passo é conectá-lo via cabo de rede com o CLP ou remota do painel. Após deve-se conectar também via rede o *hardware* de teste com o computador que possui o *software* do TIA Portal.

Assim que todos os *hardwares* encontram-se conectados fisicamente, deve-se acessar no TIA Portal a opção **Online** > **Hardware detection** > **PROFINET devices from network** e então, procurar na rede o CLP ou remota do painel. Após encontrado, basta executar via *software* as conexões de rede nas mesmas portas que foram conectadas fisicamente. Na Figura 61, observa-se um exemplo de conexão.

Figura 61 – Configuração dos *softwares* na rede



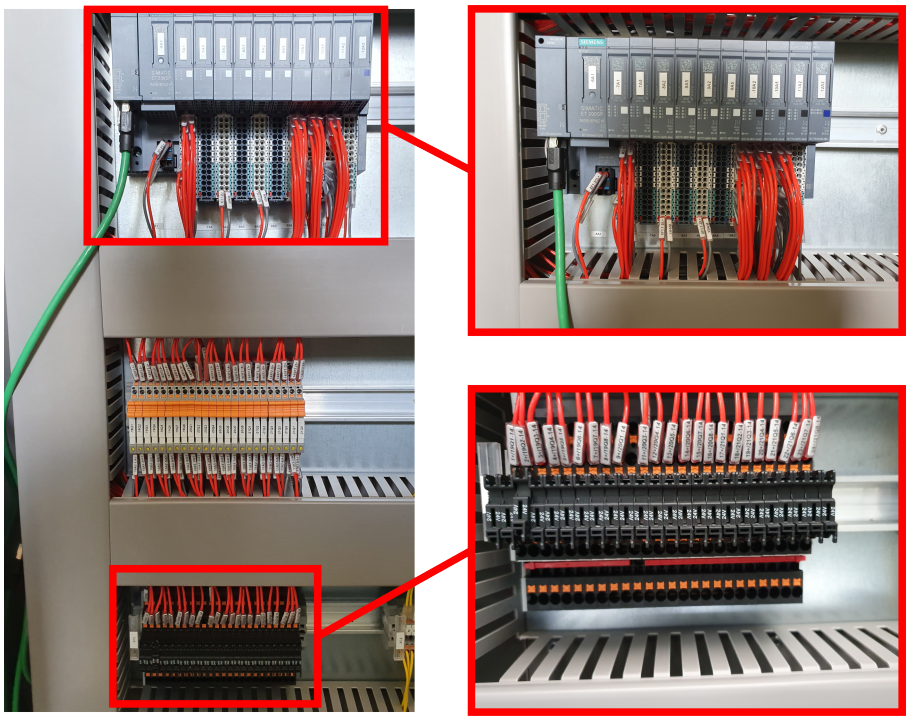

Fonte: Arquivo pessoal.

Após todos *hardwares* conectados fisicamente e no TIA Portal, deve-se atribuir o endereço correto para o cartão a ser testado, como visto na Figura 34.

Por fim, por meio da IHM navega-se até o opção desejada, por exemplo, cartão com 8 entradas digitais e executa-se o teste conectando as *tags* de acordo com o que a IHM determinar.

Por meio da Figura 62, observa-se as extremidades de uma conexão a ser testado, onde na parte superior encontram-se os cartões digitais e analógicas e na parte inferior encontram-se os bornes de saída do painel. Assim, para o teste de uma entrada digital, basta conectar o fio no saída digital do *hardware* de teste no borne que corresponde a entrada digital e pressionar o botão **TESTAR** na IHM. O teste é concluído quando testa-se todos os cartões e gera-se todos os relatórios, que devem ser todos assinados pelo executor do teste e o representante da empresa solicitante.

Figura 62 – Exemplo prático de um painel



Fonte: Arquivo pessoal.

# 4 RESULTADOS DO PROJETO

Após a finalização do *software*, *hardware* e os testes realizados, o resultado apresentado foi bastante satisfatório. O projeto atingiu os objetivos propostos inicialmente e dentro do prazo estipulado. Para execução do TAF, necessitou-se do deslocamento até o setor de montagem da empresa.

Já no local, no primeiro dia buscou-se a ambientação com os paineis e o *hardware*. O teste dos paineis ocorreu no segundo dia, onde foram testados 4 paineis de remota com 30 cartões de entradas e saídas, analógicas e digitais. Todo o processo foi cronometrado e os tempos totais serão demonstrados a seguir.

- Para o TAF inicial da parte mecânica e elétrica obteve-se um tempo total de 220 minutos, ou seja, um tempo médio de 55 minutos por painel;
- Para a conexão do *hardware* e configuração do *software* obteve-se um tempo total de 20 minutos, ou seja, um tempo médio de 5 minutos por painel;
- Para execução do TAF dos cartões e geração de relatórios, obteve-se um tempo total de 30 minutos, ou seja, um tempo médio de 1 minuto por cartão.

Para validação dos objetivos, os mesmos paineis foram testados por funcionários da GreyLogix Brasil e os tempos totais serão demonstrados a seguir.

- Para o TAF inicial da parte mecânica e elétrica utilizou-se o mesmo tempo do teste anterior, pois esse teste não faz parte do escopo do projeto;
- Para execução do TAF dos cartões obteve-se um tempo total de 600 minutos, pois a função foi exercida em 300 minutos por 2 funcionários, ou seja, um tempo médio de 10 minutos por cartão.
- Para geração de relatórios necessitou-se de um tempo total de 60 minutos, ou seja, um tempo médio de 15 minutos por painel.

Assim, compara-se os valores e obtém os seguintes números:

- do modo tradicional, necessitou-se de 880 minutos para completar o teste dos paineis;
- com uso do *software*, necessitou-se de 270 minutos para completar o teste.

Assim, obteve uma diminuição de 69,3% de tempo em um TAF completo, que foi um resultado muito satisfatório. Após, realizou-se uma reunião com alguns coordenadores da empresas e desenvolveu-se diversos cálculos para o retorno do investimento. Estipulou-se que foi investido cerca de R$10.000,00 reais no projeto, incluindo todos os custos do execução, treinamentos e ferramentas e economizou-se cerca de R$250,00 no teste de cada painel. Assim, desconsiderando os custos com todo o *hardware* que estava disposto no laboratório da empresa, conclui-se que o retorno de investimento daria-se no teste de 40 paineis.

Além da diminuição do tempo de um TAF, a geração automática de relatórios trouxe um grande benefício para a empresa e para o cliente, visto que não necessita-se mais de um tempo após o teste para construir os relatórios e na realização dos testes junto ao cliente, o mesmo já visualiza a validação do teste através do relatório, podendo assinar e autenticar a execução. Outro ponto a ser considerado foi a eliminação de grande parte da responsabilidade do testador em interpretar o projeto, visto que a interpretação é feita pelo *software*, possibilitando que o testador foque exclusivamente na execução do teste.

# 5 CONSIDERAÇÕES FINAIS

Após o desenvolvimento do projeto e inúmeros testes no sistema, constatou-se os benefícios e eficiência que o mesmo trará para a organização. Foi possível acompanhar a sinergia das equipes da automação, elétrica e comercial, pois graças ao empenho dos envolvidos os objetivos foram alcançados com grande satisfação.

Ao longo deste documento foram apresentadas as etapas do projeto relativas ao desenvolvimento do protocolo e *software* para TAF em paineis de automação. Do ponto de vista acadêmico, o trabalho tem como característica a multidisciplinaridade, envolvendo diversos conceitos vistos ao longo da graduação em Engenharia de Controle e Automação, tais como: arquitetura e programação de sistemas microcontrolados, circuitos elétricos para automação, modelagem e controle de sistemas e eventos discretos, instrumentação, acionamentos elétricos e redes industriais. O projeto também proporcionou a oportunidade de adquirir conhecimentos nas ferramentas de engenharia mais modernas da atualidade, utilizando-as para a solução de um problema real de automação no meio industrial.

O projeto foi bastante motivador e desafiador. Motivador por ser algo novo implementado na empresa, onde colocou-se em prática todo o conhecimento adquirido na graduação e desafiador por proporcionar-me o uso de ferramentas sem uma experiência anterior, sempre prezando pela qualidade do projeto e cumprimento dos prazos. Apesar do código parecer simples, necessitou-se de vários dias de testes para chegar na versão final, levando em consideração o menor uso de memória do CLP, pois os códigos possuem milhares de linhas e caso não seja otimizado pode aumentar o tempo de *setup* do *software*.

Como perspectivas futuras, objetiva-se a construção mecânica de uma maleta de testes para o transporte do *hardware* de teste e a substituição da IHM por um tablet, que será possível através do uso de um pico pc ou IOT2040, melhorando o manuseio por conta do testador e possiblitando anotações das correções e *feedbacks* para a equipe elétrica diretamente no COMOS, assim, em tempo real e em qualquer lugar a equipe técnica estará em contato direto com o executor do teste, evitando de forma instantânea a propagação de erros de projeto.

Desta forma, conclui-se que através das tecnologias disponíveis no mercado e a contínua busca por novos conhecimentos, torna possível o desenvolvimento de sistemas aptos a otimizar processos, acarretando a uma maior produtividade com um menor custo e consequentemente tornando a organização mais eficiente.

# Referências

ABNT. [S.l.]: Associação Brasileira de Normas Técnicas, 2019. <https://www.abnt.org.br/>. [Acesso em 24 de Novembro de 2019]. Citado na página 29.

CAETANO, A. F. de S. Verificação automática de sistemas descritos usando a linguagem ladder. p. 33, 2018. Citado na página 22.

GLX-ALEMANHA. [S.l.]: Greylogix - xStory - Bilfinger Greylogix GmbH, 2019. <https://greylogix.com/xstory/>. [Acesso em 21 de Outubro de 2019]. Citado na página 12.

GLX-BRASIL. [S.l.]: GreyLogix Brasil, 2019. <http://www.greylogix.com.br/>. [Acesso em 13 de Maio de 2018]. Citado na página 12.

IEC-62381, IEC-62381. [S.l.]: Automation systems in the process industry – Factory acceptance test (FAT), site acceptance test (SAT), and site integration test (SIT), 2006. Citado na página 23.

NEMA. [S.l.]: National Electrical Manufacturer Association, 2019. <https://www.nema.org/>. [Acesso em 24 de Novembro de 2019]. Citado na página 29.

SANTANA, C. V. C. Z. . E. A. . E. V. F. J. C. F. J. R. Sistema de automação de pequenas ses com vários meios de comunicação. p. 4, 2008. Citado na página 22.

SIEMENS. [S.l.]: TIA Portal, 2019. <https://new.siemens.com/br/pt/produtos/automacao/software-industria/automation-software/tia-portal.html>. [Acesso em 23 de Outubro de 2019]. Citado na página 15.

SIEMENS. [S.l.]: TIA Selection Tool, 2019. <https://new.siemens.com/global/en/products/automation/topic-areas/tia/tia-selection-tool.html>. [Acesso em 23 de Outubro de 2019]. Citado na página 16.

SIEMENS. [S.l.]: COMOS, 2019. <https://new.siemens.com/br/pt/produtos/automacao/software-industria/comos.html>. [Acesso em 22 de Outubro de 2019]. Citado 2 vezes nas páginas 17 e 18.

SIEMENS. [S.l.]: SIMATIC WinCC, 2019. <https://w3.siemens.com.br/automation/br/pt/automacao-e-controle/Controladores-SIMATIC/SIMATIC-IHM/SIMATIC-WinCC/Pages/simatic-wincc.aspx>. [Acesso em 23 de Outubro de 2019]. Citado 2 vezes nas páginas 19 e 20.

SIEMENS. [S.l.]: SIMATIC IHM, 2019. <https://w3.siemens.com.br/automation/br/pt/automacao-e-controle/Controladores-SIMATIC/SIMATIC-IHM/Pages/simatic-ihm.aspx>. [Acesso em 23 de Outubro de 2019]. Citado na página 20.

SIEMENS. [S.l.]: SCL, 2019. <https://www.automation.siemens.com/sce-static/learning-training-documents/tia-portal/advanced-programming-s7-1200/sce-051-201-scl-s7-1200-r1709-pt.pdf>. [Acesso em 03 de Novembro de 2019]. Citado na página 21.

TORRES, G. “redes de computadores”. Novaterra Edidora e Distribuidora., 2010. Citado na página 20.