Lucas Russignoli Peixoto

Sistema de Controle de Demanda Energética para Indústria

UBERLÂNDIA

2023

Lucas Russignoli Peixoto

Sistema de Controle de Demanda Energética para Indústria

Projeto de Trabalho de Conclusão de Curso da Engenharia de Controle e Automação da Universidade Federal de Uberlândia - UFU - Câmpus Santa Mônica, como requisito para a obtenção do título de Graduação em Engenharia de Controle e Automação.

Universidade Federal de Uberlândia – UFU

Faculdade de Engenharia Elétrica

Orientador: Prof. Dr. Renato Ferreira Fernandes Junior

UBERLÂNDIA

2023

Agradecimentos

À Deus, por me capacitar e ter me abençoado tanto até aqui.

À minha família, que nunca deixou de me apoiar durante todos os momentos, sempre me dando conselhos e me encorajando.

À minha namorada, Marina Bizzotto, que me acompanhou desde o início da faculdade e sempre esteve ao meu lado.

Ao meu orientador Renato Fernandes, que me incentivou, amparou tecnicamente e me ajudou durante todo o meu trabalho final.

Aos meus amigos, que me garantiram uma família dentro da universidade.

À Universidade Federal de Uberlândia e seus professores, por terem me dado a oportunidade de aprendizado.

Resumo

Desde sua invenção, a energia elétrica se tornou indispensável na vida das pessoas, do uso doméstico – com o bem-estar gerado pela eletricidade – até o papel crucial nos processos produtivos industriais. Devido ao impacto ambiental desencadeado pela geração de energia, é importante que seja utilizada de maneira consciente.

Além disso, uma vez que a infraestrutura instalada para distribuição de energia é baseada em uma demanda específica, o consumo estabelecido entre contratante e a concessionária de energia é um importante tópico a ser respeitado, principalmente no ramo industrial, onde as fábricas estão em constantes expansões, devido a necessidade de aumento de produção.

O projeto de controle de demanda proposto neste trabalho visa automatizar os motores de ventiladores dos secadores de grãos, onde podem ser desligados de acordo com a demanda da empresa. Para este sistema foi proposto uma lógica de controle dos motores de forma a proteger o transformador, otimizar a lógica de secagem e, principalmente, controlar a demanda energética obtida da distribuição. Para isso foram utilizados controladores, sistema supervisório de mercado e protocolo de comunicação Modbus. Os resultados mostraram que a aplicação do sistema de automatização pode gerar uma redução de custo considerável para a empresa.

Palavras-chave: Consumo de Energia, Controle de Demanda, Seletividade de Carga, Automação, Modbus.

Abstract

Since its invention, electrical energy has become indispensable in people’s life, domestic use – with the comfort generated by electricity – to the crucial role in production industrial process. Due to the environmental impact triggered by the generation of energy, it is important that it be used consciously.

Furthermore, since the installed infrastructure is based on a specific demand, the energy consumption established between hiring and power distribution energy is an important topic to be appreciated, mainly in the industrial branch, where the factories are in constant expansion, due to the increase capacity of production.

The demand control project proposed in this work aims to automate grain dryer fan motors, which they can be turned off according to the company's demand. For this system, a motor control logic was proposed to protect the transformer, optimize the drying logic and, mainly, control the energy demand obtained from distribution. For this, controllers, a market supervisory system and Modbus communication protocol were used. The results showed that the application of the automation system can generate considerable cost reduction for the company.

Keywords: Energy Consumption, Demand Control, Load Selectivity, Automation, Modbus.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

A	*Ampère – Unidade de medida de corrente elétrica*
CCM	*Centro de Controle de Motores*
CLP	*Controlador Lógico Programável*
CPU	*Unidade de Central de Processamento*
FBD	*Function Block Diagram*
FP	*Fator de potência*
Hz	*Hertz - Unidade de medida de frequência*
IHM	*Interface Homem-Máquina*
LD	*Ladder Diagram*
m	*Metros - Unidade de medida de distância*
mA	*miliAmpère*
OSI	*Open Systems Interconnection*
P	*Potência ativa*
Q	*Potência reativa*
RS-485	*Recommended Standard 485*
RTU	*Remote Terminal Unit*
S	*Potência aparente*
SCADA	*Supervisory Control and Data Acquisition*
TCP	*Transmission Control Protocol*
V	*Ampère – Unidade de medida de tensão elétrica*
VA	*Volt-Ampère – Unidade de medida de potência*
W	*Watts – Unidade de medida de potência*

Sumário

1. Introdução

Com o crescimento populacional do mundo, a necessidade de melhoria nos sistemas de produção de alimentos tem aumentado, para que se produza mais em espaços cada mais enxutos. A adoção de tecnologias inovadoras desempenha um papel fundamental no setor agrícola, buscando diminuir as perdas durante a produção, otimizando a produtividade e o armazenamento. (ELIAS, 2002). Por outro lado, existe a necessidade também da melhoria da eficiência energética, aumentando a produção de forma sustentável, por meio de instalações mais eficientes. (CASTELLANO, 2015).

Tais melhorias são obtidas através da modernização das técnicas de plantio, de colheita e pós-colheita como estocagem e secagem dos grãos. A eficiência energética pode estar em qualquer umas das etapas de produção, que se traduz na diminuição do consumo de energia, refletindo em menores custos e se tornando uma produção mais sustentável.

Os silos secadores são os dispositivos responsáveis por levar o grão até a umidade correta, a qual pode variar de acordo com a finalidade do produto. Geralmente, os secadores são sistemas antigos que muitas vezes podem ser mais eficientes no processo de secagem.

Em relação a energia que alimenta estes secadores, no Brasil as concessionárias de energia vendem um valor fixo em energia elétrica aos contratantes, cujos valores dependem da concessionária. Caso a demanda consumida seja maior que a contratada, a empresa pode receber valores de multa de até 3 vezes o de horário de pico. (CASTELLANO, 2015).

Desta forma, a automação destes sistemas de secagens é algo totalmente viável e traz um benefício enorme em relação a eficiência e a sustentabilidade dos sistemas.

1.1. Justificativa

Dentro de uma indústria, a eficiência energética é essencial pois traz redução de custos e melhoria em preços da produção. Quando se tem um consumo maior que a demanda contratada, o contratante sofrerá com cobranças de maiores valores que os acordados no ato da contratação da energia elétrica. O preço praticado tem um aumento devido à essa ultrapassagem, uma vez que a concessionária realiza toda a distribuição e infraestrutura – a qual corre riscos de ser danificada – baseados no valor contratado. As

taxas pagas ao extrapolar a demanda vão de acordo com as políticas de cada concessionária de energia e o valor a ser pago vem especificado na fatura seguinte.

Desta forma, o controle da demanda se torna mais uma das atenções para alcançar esses objetivos e, com o aumento de seus equipamentos – consequência do aumento de produção – acentua-se ainda mais a preocupação com o consumo de energia. Visto isso, torna-se importante o investimento em vias para manter esse crescimento de forma adequada, mantendo um consumo de energia dentro dos limites propostos, eliminando gastos com multas aplicadas à companhia e evitando danos à equipamentos e a infraestrutura instalada.

1.2. Objetivo

Este trabalho tem como objetivo elaborar um sistema de controle de demanda para aplicações em sistemas agrícolas de secagem de grãos. A proposta visa uma automação para que a empresa não extrapole – durante a sua operação – os valores limites da demanda pré-estabelecida pela concessionária de energia.

O sistema deve permitir que o usuário escolha qual o setor que os equipamentos serão desligados caso a demanda esteja acima da estipulada, gerando mais previsibilidade e evitando o desligamento de equipamentos prioritários no momento da operação.

Desta forma, este trabalho terá os seguintes objetivos específicos.

- Desenvolvimento de um controle de demanda usando CLP industrial visando desligamento de cargas por ordem de prioridade, setup de diferentes limites de demanda e setup do tempo de desligamento entre cargas;
- Comunicação do CLP central com os equipamentos distribuídos no campo através de protocolo Modbus;
- Configuração pelo operador através de um sistema supervisório em uma interface homem-máquina.

2. Referencial Teórico

Neste tópico será descrito todo o embasamento teórico necessário para melhor compreensão deste trabalho. Como o trabalho se trata de demanda de energia elétrica, serão apresentados os conceitos de um sistema de consumo de energia e as tecnologias da automação de sistema.

2.1. Processo de Secagem e Armazenagem de Grãos

No Brasil, na safra 2022/2023, a exportação de soja e milho somam mais de 280 milhões de toneladas, essa produção pode chegar a 346 milhões de toneladas na safra de 2032/2033, conforme Figura 1, o que evidencia a força que o Brasil carrega – com potencial de crescimento – na produção agrícola, o levando a ser um dos países com maior atividade no setor do mundo. (Ministério de Agricultura e Pecuária, 2023).

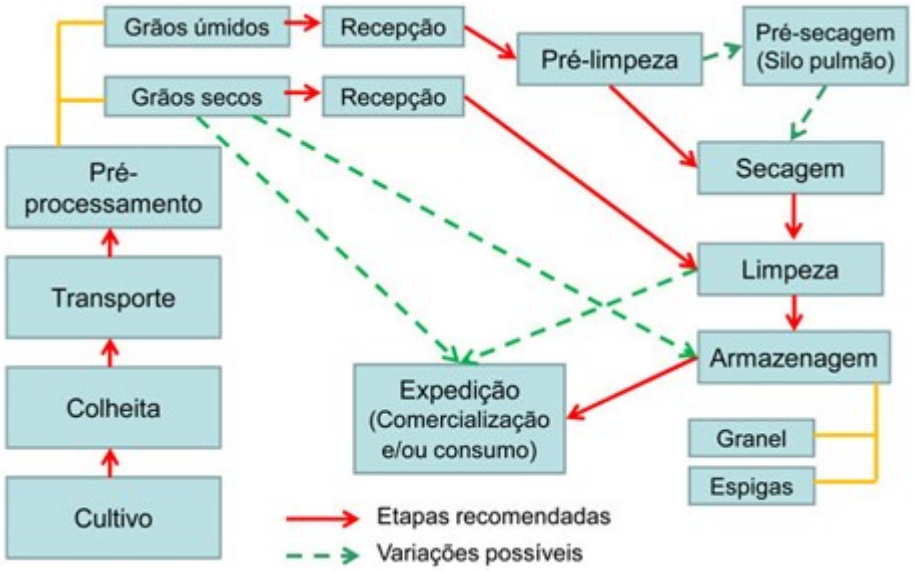
Figura 1 – Quantidade de grãos de soja e milho exportados pelo Brasil

Fonte: Adaptado de (Ministério de Agricultura e Pecuária, 2023).

Para que a qualidade dos grãos se mantenha no nível adequado durante a espera para venda ou replantio, o ponto crucial é, justamente, como manuseia-se o início do tratamento deles. Este manejo inicial de grãos requer um silo secador e um silo armazenador para sua preservação. Dessa maneira, os produtores rurais podem aguardar o melhor momento da venda, sem perder qualidade no produto, maximizando os lucros. (SOUZA, et.al., 2010). Além disso, pode-se usar a secagem para chegar no ponto ideal de umidade para seguir com o processo de beneficiamento de sementes, garantindo a germinação ideal

quando a mesma for usada para replantio. A Figura 2 mostra o processo de tratamento dos grãos.

Figura 2 – Fluxograma básico para secagem e armazenamento de grãos



Fonte: (OLIVEIRA, 2021)

A colheita dos grãos, na maioria dos casos, é feita com teor de umidade acima do adequado – principalmente por eventos climáticos – o que faz da secagem um processo quase obrigatório. (ELIAS, 2002). Para o beneficiamento da semente, a umidade ideal segue a carta psicométrica do grão secado e depende de fatores como umidade de entrada, temperatura ambiente, umidade ambiente e temperatura máxima de secagem. Já para o armazenamento, as umidades ideias em relação ao tempo de conservação seguem o padrão descrito na Figura 3.

Figura 3 – Grau de umidade [%] recomendado para armazenamento de grãos

Grão	Período de armazenamento (meses)			
	06	12	24	60
1. Feijão	14,5	13,5	12,5	11,5
2. Milho	14,0	13,0	12,0	11,0
3.Trigo, sorgo, arroz, centeio, aveia, triticale	13,5	12,5	11,5	10,5
4. Azevém	13,0	12,0	11,0	10,0
5. Soja	12,5	11,5	10,5	9,5
6. Amendoim	12,0	11,0	10,0	9,0
7. Colza/canola	9,0	8,0	7,0	7,0

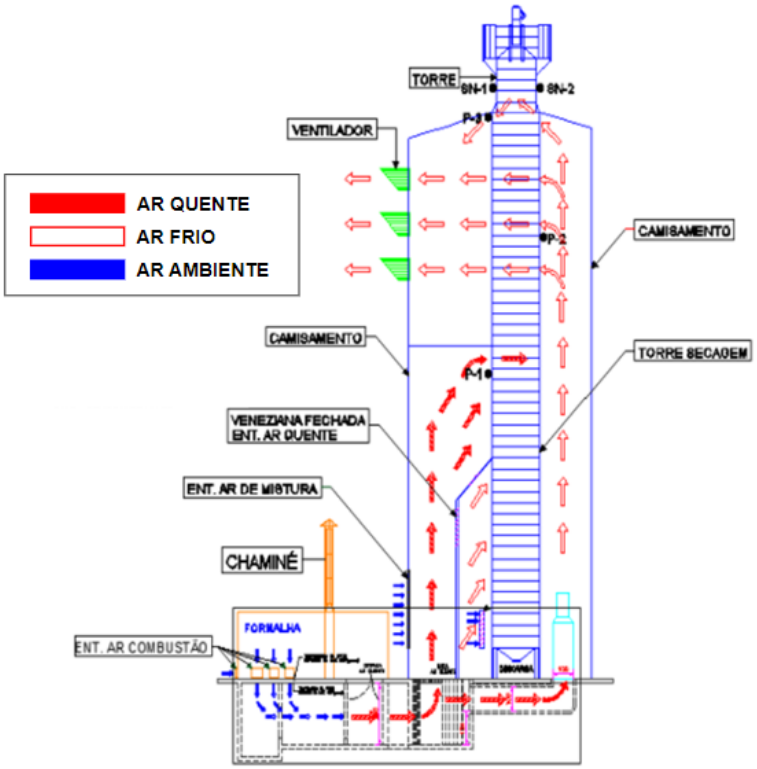
Fonte: (ELIAS, 2002).

Para a secagem, existem dois tipos básicos:

- Secagem que utiliza de uma fonte de calor externa, para secar com ar quente (secadores de alta temperatura ou alta velocidade);
- Secagem com ar em temperatura ambiente (secadores de baixa temperatura).

Na Figura 4, é ilustrado um secador que utiliza a secagem com ar quente, que, neste caso, através de uma fornalha, gera o calor necessário para esquentar o ar.

Figura 4 – Exemplo de sistema de secagem com ar quente

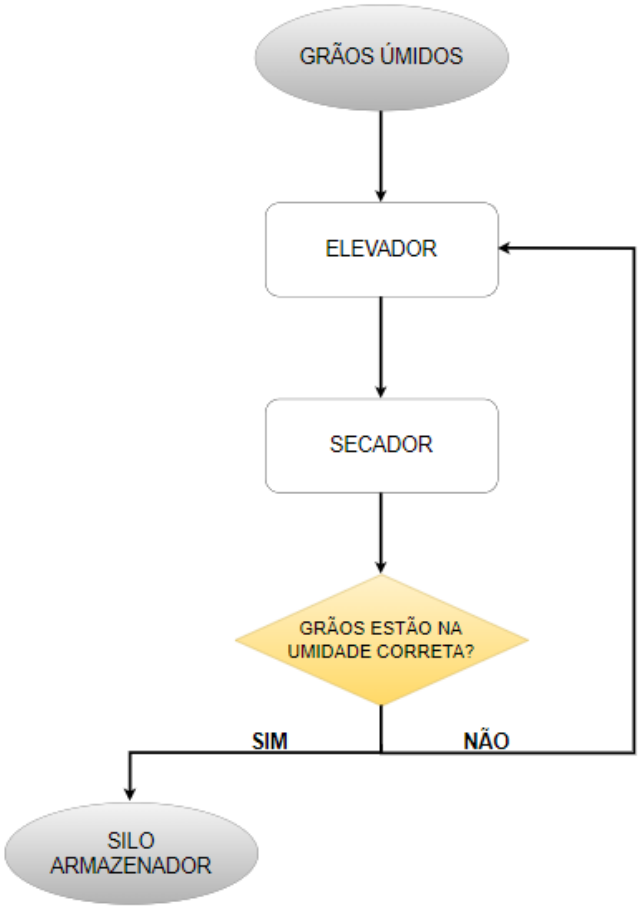


Fonte: Adaptado de (SULZBACHER, et. al., 2013).

O funcionamento do secador é baseado na passagem de ar pelos grãos, fazendo com que a umidade deles diminua. (Fimaco, 2018). Além disso, para alcançar a umidade ideal, existe o modo de secagem onde os grãos passam várias vezes pelo secador, secagem intermitente. (SANTOS, 2019). Neste tipo de secagem, o sistema conta com um elevador, o qual – através de uma válvula bifurcada – enviará os grãos novamente para o secador ou os depositará no silo armazenador. Sendo assim, se os grãos estiverem com umidade acima do valor desejado após o primeiro ciclo de secagem, é possível que eles se movimentem

para então iniciar um novo ciclo. Este processo é mostrado na Figura 5.

Figura 5 – Fluxograma simplificado da rotação de grãos



Fonte: Próprio autor (2023).

2.2. Controle de Demanda

Demanda e consumo são termos com significados distintos. O consumo elétrico é a energia que foi realmente usada para satisfazer um sistema de cargas durante um período decretado, já a demanda elétrica equivale ao valor carecido da rede para satisfazer o mesmo sistema dentro das mesmas condições, sendo assim, maior que o consumo.

Existem 3 vias que devem ser compreendidas ao definir uma demanda: a contratada, a requerida e a excedida. A demanda contratada é definida pela ANEEL (Agência Nacional de Energia Elétrica) como a demanda que é continua e forçosamente disponibilizada pela concessionária de energia no local em conformidade com os valores e períodos estabelecidos no contrato, sendo inteiramente paga independente da utilização

dela. Ou seja, aquela que é comprada da concessionária de energia, com as taxas e valores acordados previamente.

A demanda requerida é aquela que o sistema realmente precisará para ser satisfeito, sendo variável com a quantidade de cargas instaladas e, por fim, a demanda excedida é aquela usada além da contratada, podendo ser cobrada de maneira extraordinária com taxas acima das combinadas no ato da contratação.

No Brasil, divide-se os contratantes de demandas em grupos, o setor industrial, na maioria dos casos, se enquadra no Grupo A de demanda contratada, os quais são consumidores com maior consumo (OZUR, et.al., 2011). Sendo assim, as regras de compra de energia elétrica para consumo são específicas para esse tipo de fornecimento, criando uma estrutura que exige planejamento tanto dos contratantes quanto das concessionárias de energia.

Um sistema de automação para controlar e supervisionar a demanda de uma fábrica ainda não é comum de se encontrar, porém, com o avanço da tecnologia dentro da indústria, a automação está presente em cada vez mais dos diversos setores que lá existem. Sendo assim, é possível o desenvolvimento de um sistema completo de automação para manter o consumo de uma fábrica dentro dos limites estabelecidos, desde o uso de controladores industriais até o monitoramento através de um sistema supervisório.

2.3. Grandezas Elétricas

Para melhor compreensão do sistema implementado, é importante o conhecimento do que são e de como são calculadas as grandezas elétricas utilizadas neste trabalho, são elas: tensão elétrica, corrente elétrica, fator de potência, potência aparente, potência ativa e potência reativa.

2.3.1. Tensão elétrica

Sempre que ocorre a movimentação de um elétron em um condutor é indispensável que ocorra trabalho, o qual é realizado através de uma força eletromotriz, que é a tensão elétrica. Portanto, a energia imposta para mover uma carga de um ponto X para um ponto Y é a diferença de potencial entre esses pontos, e é medida na unidade Volts (V). (BOLESTAD, 2011). A tensão é, geralmente, representada pela letra V ou U.

2.3.2. Corrente elétrica

Uma vez aplicada a tensão, ocorre, por consequência, a formação de um fluxo de cargas elétricas no condutor, onde cargas positivas se movem para uma direção e cargas negativas se movem na direção oposta. Para esse fluxo de cargas se dá o nome de corrente elétrica, e, normalmente, convenciona-se o sentido da corrente no mesmo sentido do movimento das cargas positivas. (SADIKU, et.al., 2003). A unidade de medida da corrente elétrica é Ampère (A) e a letra que a representa é a letra I.

2.3.3. Fator de potência

O fator de potência é obtido através da razão entre a potência ativa e a potência aparente ou pelo cosseno do ângulo Φ – o qual representa o ângulo de defasagem entre as ondas de tensão e corrente do sistema – do triângulo de potências. É importante para o levantamento de quanto de potência consumida da rede está realmente sendo usada como trabalho. (MATTEDE, 2016). O fator de potência é representando pela sigla FP e calculado pela seguinte fórmula.

$$FP = \frac{Pot.Ativa}{Pot.Aparente} = cos\,\Phi \qquad (1)$$

2.3.4. Triângulo de Potências

O triângulo é composto por 3 potências – como visto na Figura 6 – aparente, ativa e reativa, onde o ângulo Φ representa, justamente, o ângulo do fator de potência. A potência ativa é responsável por realizar trabalho, também chamada de potência útil, e tem como unidade de medida Watts (W). (MATTEDE, 2016). A potência ativa é, comumente, representada pela letra P.

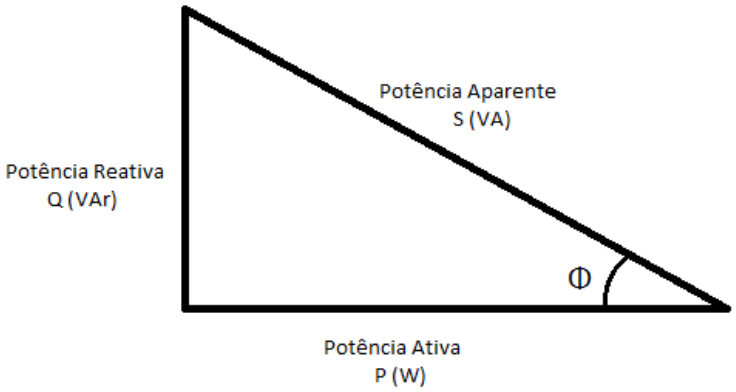
A potência reativa é aquela que alterna entre a fonte elétrica do sistema e as cargas dele. Importante para magnetizar as bobinas dos equipamentos elétrico, necessário para o funcionamento do motor elétrico, por exemplo. É medida através da unidade kilo volt-ampère reativo (kVAr). (AlugaGera, 2019). A potência reativa é, costumeiramente, representada pela letra Q.

Por fim, a potência aparente é composta pela parcela reativa e a parcela ativa. É a potência fornecida para que o consumidor seja satisfeito, ou seja, a potência que as cargas

precisam para funcionarem. É medida através da unidade kilo volt-ampère (kVA). (MACIEL, 2021). Para o sistema trifásico, segue a seguinte forma de calcular a potência aparente total, a qual é representada pela letra S. O triângulo de potências é mostrado na Figura 6.

$$S = 3 * V_{fase} * I_{fase} \quad (2)$$

Figura 6 – Triângulo de potências



Fonte: (MACIEL, 2021)

O cálculo das potências ativa e reativa se dão a partir do cálculo da potência aparente e do triângulo de potências, logo, tem-se as seguintes maneiras de calculá-las.

$$P = S * cos\,\Phi \quad (3)$$

$$Q = S * sen\,\Phi \quad (4)$$

2.4. Automação Industrial

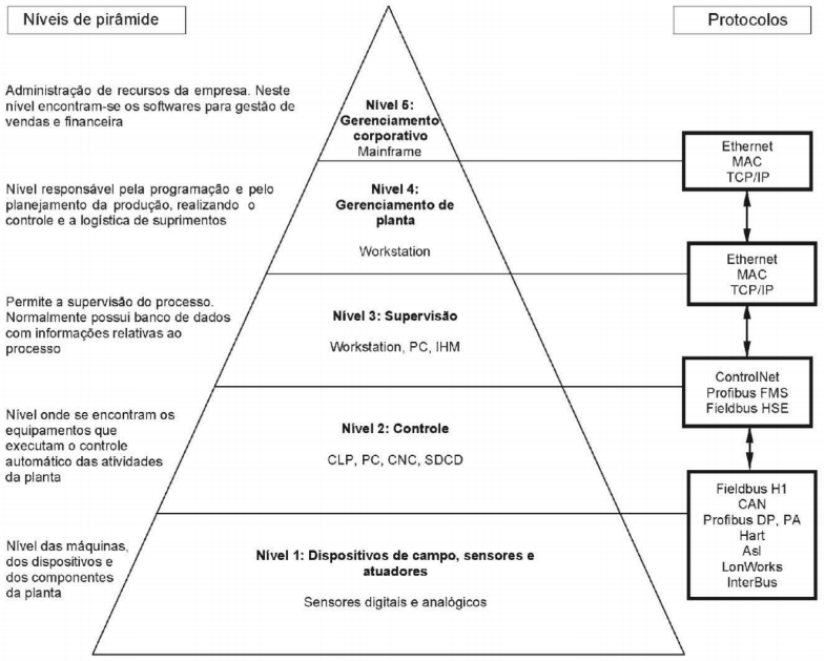
A concepção de automação industrial é vinculada a aplicação de ferramentas e tecnologias de hardware e software para otimizar processos industriais, reduzindo custos e aumentando a capacidade produtiva de uma empresa. A automação pode atuar substituindo trabalhos manuais, como é o caso de aplicação de alguns tipos de robôs, mas também pode cooperar com o usuário, uma vez que melhora a capacidade de entendimento do processo, colaborando para que haja um melhor controle e monitoramento do sistema.

Dentro de um silo armazenador, podemos citar automação para controle de

temperatura e umidade, já em um silo secador – ou somente secador – a automação se dá com o controle dos ventiladores de secagem e controle da temperatura – através da atuação na origem do ar quente, podendo ser o controle de alimentação de uma fornalha.

A automação é um estudo de várias frentes e, devido a sua expansão, foi necessário o fracionamento em vários níveis. Com isso, nasceu a pirâmide da automação, facilitando o entendimento das funções de um sistema no todo, conforme Figura 7.

Figura 7 – Pirâmide de automação



Fonte: (MORAES, et.al., 2006).

Cada nível classificado possui um propósito, com responsabilidades exclusivas. No nível 1 estão os equipamentos, como transmissores, válvulas, motores e demais instrumentação de campo, que vão realizar a aquisição de dados e atuação no sistema. Já o nível 2, engloba dispositivos de controle, como os CLP's, equipamentos que controlam todo o processo da planta. (MORAES, et.al., 2006).

No terceiro nível, encontram-se os softwares de supervisão, como interface homem-máquina e sistema SCADA, onde é feito o monitoramento do processo. O nível 4 é responsável pelo gerenciamento da planta, busca realizar a metrificação do desempenho da produção, que são usadas para elaborar planos estratégicos para melhorar a eficiência do sistema. (MORAES, et.al., 2006).

Por fim, no último nível, o 5, é onde está o gerenciamento corporativo da empresa,

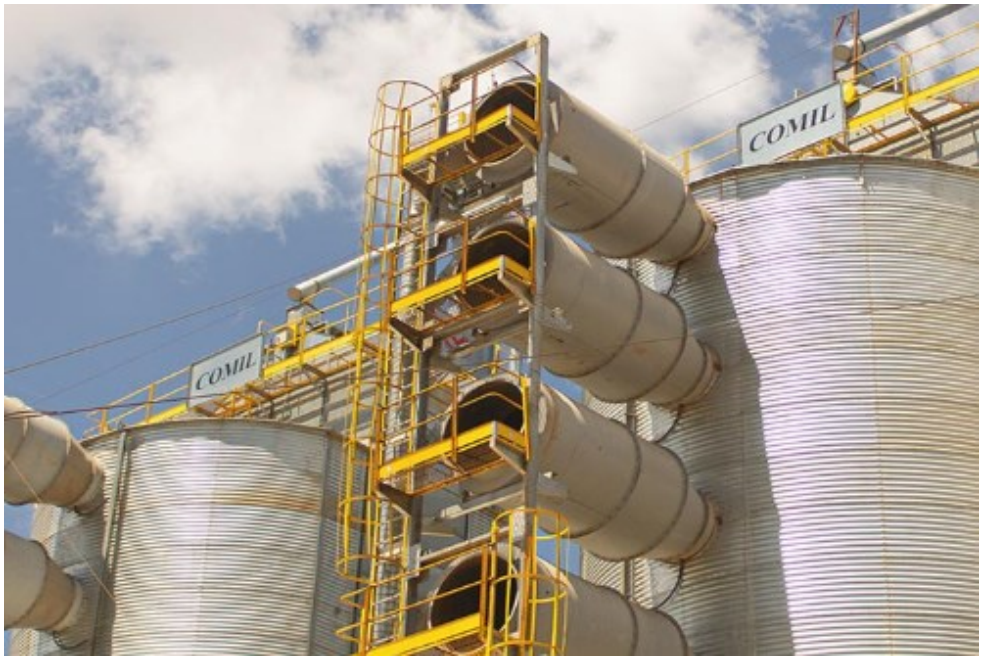
onde, a partir dos dados coletados, se tomam as decisões do futuro da empresa, da organização financeira e do planejamento geral através de softwares com essa finalidade. (MORAES, et.al., 2006).

2.4.1. Nível de sensores e atuadores

Em nível de sensores e atuadores, para armazenamento e secagem de grãos, os mais utilizados são transmissores de temperatura, de nível, de pressão e motores elétricos – para a ventilação. Uma vez que a automação realizada nesse projeto tem atuação sobre os ventiladores, torna-se necessário a explicação deles.

Como visto, os ventiladores são os equipamentos responsáveis por transportar o ar quente no interior do secador, possibilitando que a umidade dos grãos diminua. Eles são compostos por uma hélice conectada, direta ou indiretamente, ao eixo de um motor elétrico trifásico de indução. Nas Figuras 8 e 9 são mostrados os ventiladores acoplados no secador e a hélice conectada ao ventilador, respectivamente.

Figura 8 – Exemplo de ventiladores no silo secador



Fonte: (Comil, 2022).

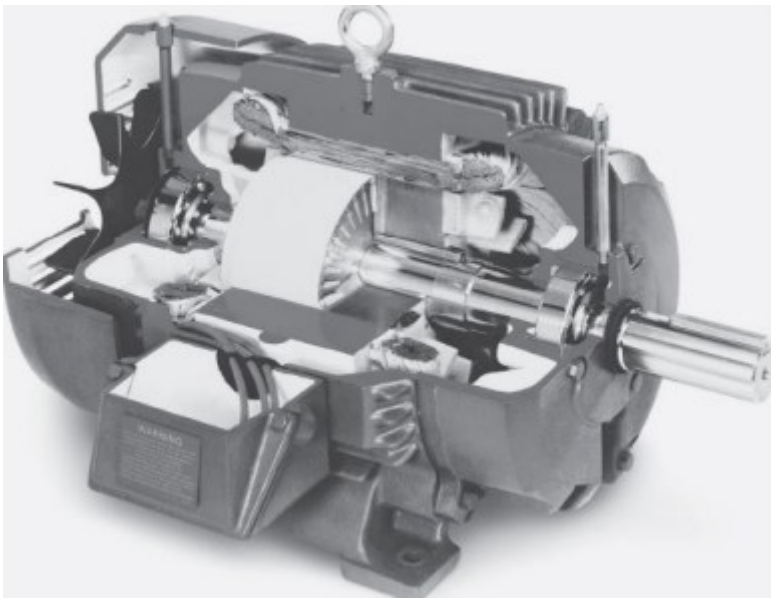
Figura 9 – Exemplo de hélice do ventilador

Fonte: (Comil, 2022).

O motor de indução trifásico é alimentado a três fios, com tensões defasadas de 120°. Ele pode transformar energia elétrica em energia mecânica e, dentre os motores usados na indústria, este é o mais utilizado. É composto por duas partes cruciais, o estator – onde se encontra os enrolamentos – e o rotor – onde está o eixo do motor. Além disso, há elementos como o ventilador acoplado ao eixo, tampa defletora – onde há aberturas na parte traseira do motor para melhor ventilação – terminais de ligação, rolamentos e a caixa de ligação. (FILHO, J. M., 2010).

O motor trifásico tem como princípio de funcionamento o campo magnético girante, o qual surge através da tensão alternada aplicada nos enrolamentos do estator. O campo magnético variável gerado provoca uma força eletromotriz no rotor, e gera um novo campo girante nele. O campo girante do rotor, ao buscar o alinhamento com o do estator, produz a rotação do eixo do motor. (MATTEDE, 2016). Na Figura 10 é evidenciado a visão longitudinal de um motor trifásico.

Figura 10 – Visão longitudinal de um motor trifásico



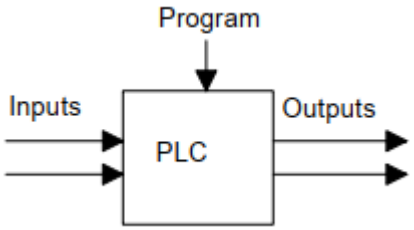
Fonte: (UMANS, 2014)

Devido ao amplo range de potências e tamanhos, os motores de indução trifásico estão em diversas aplicações, como esteiras, ventiladores e bombas, por exemplo. Além disso, estão presentes em vários setores, desde atuação na área rural, em hospitais, construção civil, mineração, sistemas de irrigação, compressores, dentre outros. Uma empresa desenvolvedora de motores de grande destaque é a WEG, com atuação nacional e internacional.

2.4.2. Controlador Lógico Industrial

Um controlador lógico programável (CLP) é um tipo de controlador fundamentado em um modelo de processador que usa uma memória onde pode-se programar para guardar instruções e para elaborar funções para controlar máquinas e processos. De maneira suscinta, o CLP é responsável por ler entradas, vindas de sensoreamento do campo, por exemplo, processá-las em uma lógica programável e definir as saídas, como é ilustrado na Figura 11.

Figura 11 – Controlador Lógico Programável

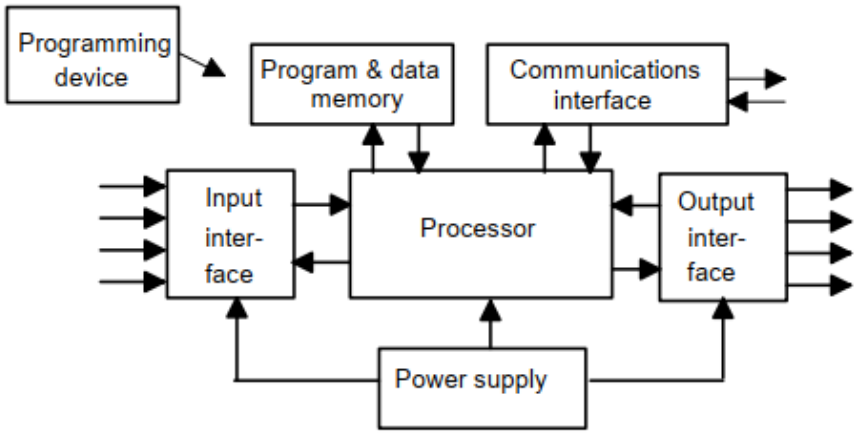


Fonte: (BOLTON, 2006).

Os CLPs são robustos e compactos, o que os tornam excelentes para aplicações industriais. O que também o faz popular é pela programação feita em blocos ou através de contatos e bobinas, o que o torna de fácil manipulação para programadores. (Barros, 2006). Além disso, é de fácil acesso para a configuração de protocolos de comunicação e conexão com sistemas supervisórios, IHM's e periféricos da indústria, como inversores e sensores. Por fim, pode-se instalar cartões adicionais nos controladores, seja para entrada ou saída, digital ou analógico, fazendo com que o CLP se torne ainda mais multifuncional.

Normalmente, um CLP conta com componentes básicos como processador, memória, entradas/saídas, interface de comunicação e o dispositivo de programação. (BOLTON, 2006). A Figura 12 mostra a disposição básica de um CLP.

Figura 12 – Sistema de um CLP



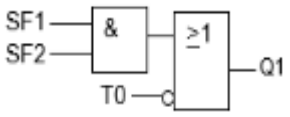
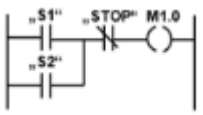
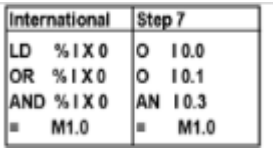
Fonte: (BOLTON, 2006).

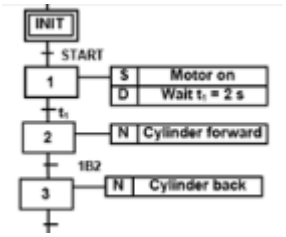
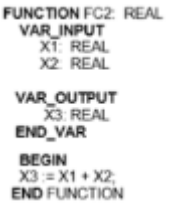
Como visto na Figura 11, os processos mais importantes de um CLP são: entradas, saídas e a lógica processada. Um exemplo de fácil entendimento desse processo seria: Sensores instalados em uma fornalha enviam a pressão através de sinais elétricos – os mais utilizados são 4 mA a 20 mA e 0 V a 10 V – o CLP os recebe e transforma isso para o valor propriamente dito da pressão, essa variável seria a entrada do CLP. Após receber esse valor, o controlador processa e toma, a cada ciclo, as decisões que foram previamente programadas nele. Por exemplo, a ação escolhida pelo CLP é abrir uma válvula guilhotina por acionamento pneumático, a qual alimenta a fornalha. O sinal de saída também é enviado através de sinais elétricos, nesse caso, como é uma ação discreta, é um sinal de 24 V.

2.4.3. Linguagens de programação de CLP

O padrão IEC61131 estabelece 5 linguagens de programação para CLPs, são elas: lista de instruções (IL), diagrama de blocos funcionais (FBD), texto estruturado (ST), mapa de função sequencial (SFC) e o diagrama Ladder (LD), conforme Tabela 1. Na Figura 13 mostra a implementação de uma lógica essas linguagens.

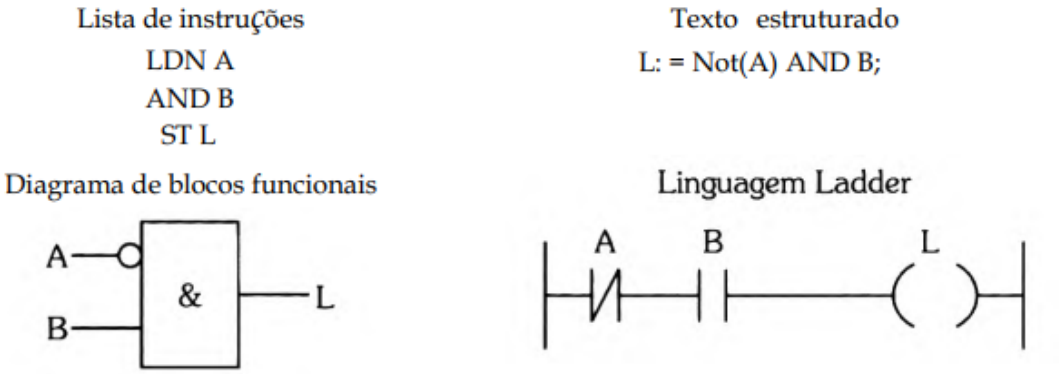
Tabela 1 – Linguagens de programação de CLP

Diagrama de Blocos de Função	
Ladder	
Lista de Instrução	

Mapa de Função Sequencial	
Texto Estruturado	

Fonte: Adaptado de (LearnChannel, 2017).

Figura 13 – Implementação da equação lógica L = Ā·B em quatro linguagens

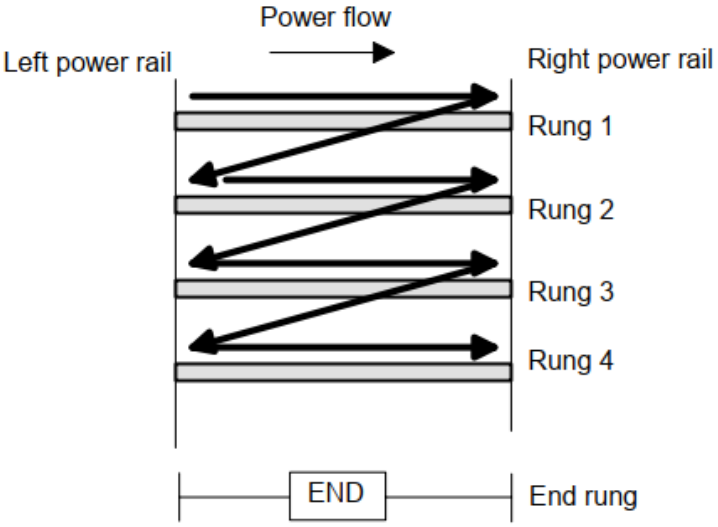


Fonte: (FRANCHI, et.al., 2008)

2.4.3.1. Diagrama Ladder (LD)

A lógica Ladder funciona em linhas horizontais, é executado da esquerda para a direita e de cima para baixo, a Figura 14 mostra como é feita a execução do código pelo CLP. Sempre que a varredura chega ao final do código, é dito que foi realizado um ciclo e, então, o CLP retorna a linha 1 para executar novamente a lógica. (BOLTON, 2006)

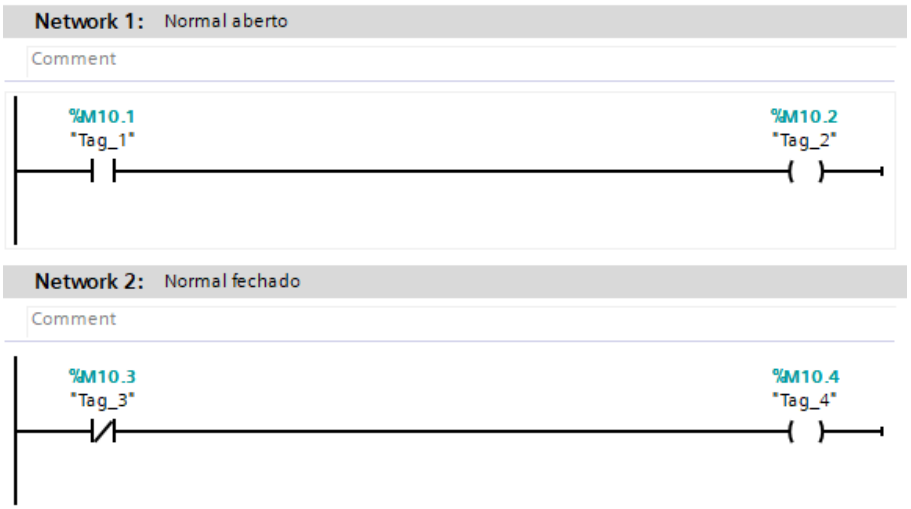
Figura 14 – Varredura da lógica Ladder



Fonte: (BOLTON, 2006)

Os elementos básicos a serem utilizados nessa linguagem são os contatos e as bobinas, entradas e saídas, respectivamente. Os contatos podem ser do tipo normal fechado ou normal aberto. O primeiro fica sempre acionado, a não ser que entre em nível lógico alto, abrindo o contato. Já o segundo, tem a lógica invertida, está normalmente aberto e fecha-se assim que estiver em nível lógico alto. Os contatos e bobinas são mostrados na Figura 15.

Figura 15 – Contatos e bobinas em Ladder

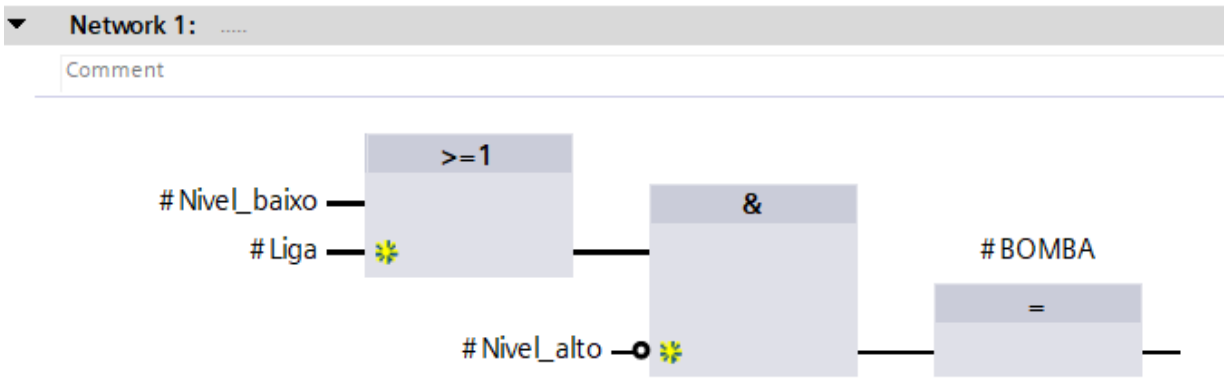


Fonte: Próprio Autor (2023).

2.4.3.2. Diagrama de Blocos de Função (FBD)

O diagrama de blocos de função é uma linguagem de CLP, assim como Ladder. A programação em FBD se trata de uma lógica em blocos que realiza funções pré-determinadas como, por exemplo, operações matemáticas, comparações, alocação e conversão de variáveis. Cada bloco é composto da seguinte maneira: entradas, execução da lógica interna e saídas. Na Figura 16 é ilustrado um exemplo de lógica em FBD.

Figura 16 – Lógica em FBD



Fonte: Próprio Autor (2023).

Outro ponto importante é que se pode criar blocos de função, de acordo com a necessidade, assim sendo, não é preciso refazer a lógica caso queira reutilizá-la, basta criar um bloco que executa aquela lógica desejada. Por exemplo, a lógica para acionamento de um motor. Caso exista 20 motores na planta, através da criação de um bloco padrão em FBD, a lógica será desenvolvida apenas uma vez e replicada para os demais motores.

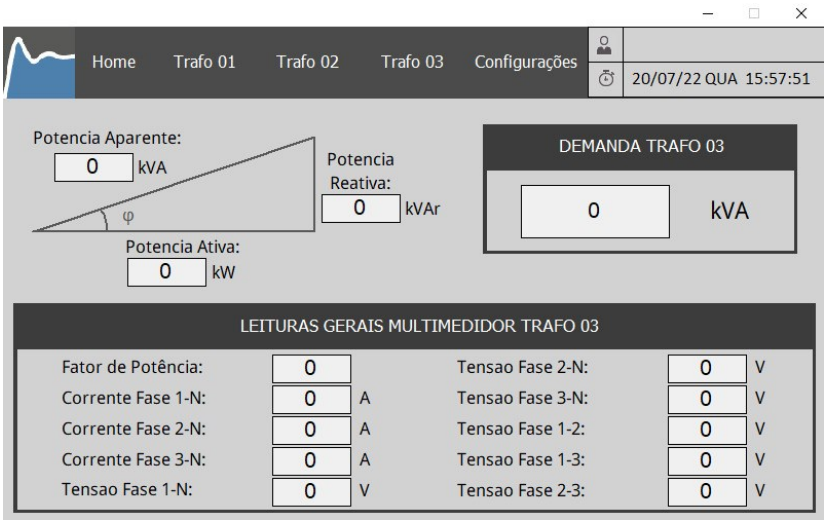
2.4.4. Sistema de Supervisão

A interface homem-máquina, com o avanço da modernização no meio industrial, se tornou crucial para um bom sistema de automação. É um equipamento que progrediu bastante nas suas funcionalidades. (ÛRZEDA, 2006). A utilização dos sistemas supervisórios varia de acordo com custos e necessidades, podem ser utilizados para recursos mais básicos, como textos ilustrativos, até recursos mais avançados, como

relatórios com históricos de produção.

IHM's são indispensáveis para melhorar a interatividade entre a lógica desenvolvida, o sistema monitorado e o operador que o utilizará, fazendo possível monitorar e escrever variáveis. Dito isso, a interface homem-máquina se torna um equipamento chave para ambientes que visam uma maior produtividade e segurança. Normalmente, a IHM é um equipamento que fica próximo ao processo monitorado pelo operador, está em campo, juntamente com as máquinas, muitas vezes no próprio painel elétrico do equipamento. Uma tela exemplo está mostrada na Figura 17.

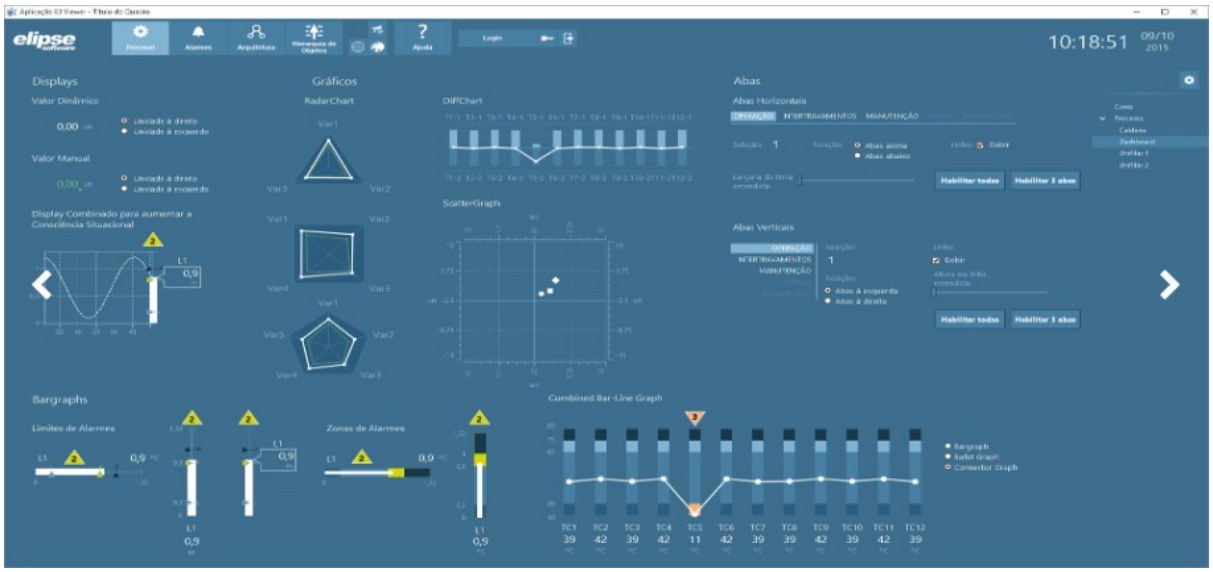
Figura 17 – Tela desenvolvida em uma IHM



Fonte: Próprio Autor (2023).

O software SCADA tem, no geral, o mesmo objetivo de uma IHM: monitorar e controlar a planta. Porém, ele geralmente está em um computador separado do processo, em uma sala de controle. Além disso, um sistema SCADA tem mais funcionalidades, como um histórico conectado a um banco de dados, por exemplo. Uma tela desenvolvida em um sistema SCADA está mostrada na Figura 18.

Figura 18 – Tela desenvolvida em um software SCADA (Elipse E3)



Fonte: (Elipse, 2015).

2.4.5. Protocolos de Redes de Comunicação

Protocolos de comunicação são preceitos estabelecidos para comunicação entre sistemas computacionais, de maneira análoga, é o idioma conversado entre dois ou mais equipamentos, como a troca de dados entre um sensor e um controlador, por exemplo. Existem vários protocolos e, neste trabalho, será utilizado o protocolo MODBUS, tanto RTU quanto o TCP. Por fim, será utilizado um gateway para estabelecer a comunicação entre o meio serial e o meio ethernet.

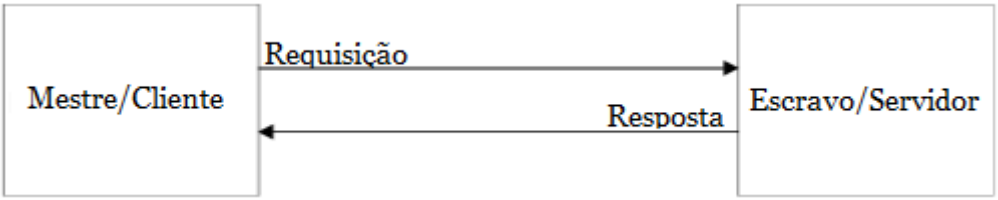
2.4.5.1. Modbus

Modbus é um protocolo de comunicação criado em 1979 pela Modicon, hoje Schneider Electric, um protocolo que utiliza do princípio mestre/escravo – ou cliente/servidor, no caso do Modbus TCP. Um controlador solicita uma mensagem, dispositivo mestre, e espera uma resposta do dispositivo escravo. (MODBUS, 1996).

Apenas o mestre pode iniciar uma mensagem, escravos não. Caso ocorra qualquer problema (como perda de comunicação) com algum escravo da rede Modbus, o mestre só saberá do feito quando fizer uma requisição para saber como está aquele escravo. O mestre não tem endereço, mas os escravos são numerados de 1 a 247. O endereço “0” é

reservado para o broadcast, mensagem enviada para todos os escravos, os quais receberão a mensagem, mas não a responderão. (THOMAS, 2008). A arquitetura de mestre/escravo é evidenciada na Figura 19.

Figura 19 – Arquitetura Requisição/Resposta



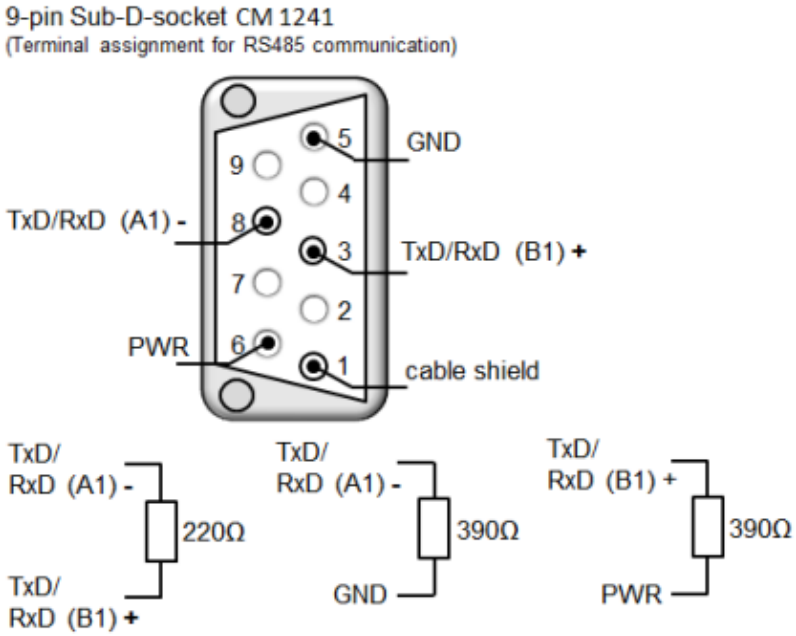
Fonte: Adaptado de (MODBUS, 2006).

O Modbus é uma rede de protocolo aberto, fácil implementação e disseminado na área industrial, ele permite que os dispositivos troquem dados discretos e analógicos, está presente via comunicação serial (RS232/RS485), como Modbus RTU e via ethernet como Modbus TCP. (MACKAY, et.al., 2004).

O meio RS485, usado no protocolo Modbus RTU, utiliza a estratégica do sinal diferencial, o que cresce a eficiência ao filtrar ruídos sobre o cabo. (FREITAS, 2017). Tem como padrão cabo de par trançado e pode ser utilizado em modo de operação half-duplex e full-duplex. (FRENZEL, 2013).

O RS485 usa como topologia de rede o barramento – ou daisy chain, uma variação dessa topologia – e suporta até 32 dispositivos por barramento, é um meio que tem velocidade de comunicação (BaudRate) máxima de 10 Mb/s e uma distância máxima de 1219 m. (FREITAS, 2017). O conector comumente utilizado para transmissão é o conector DB9, conforme Figura 20.

Figura 20 – Pinagem para cabo DB9/RS485

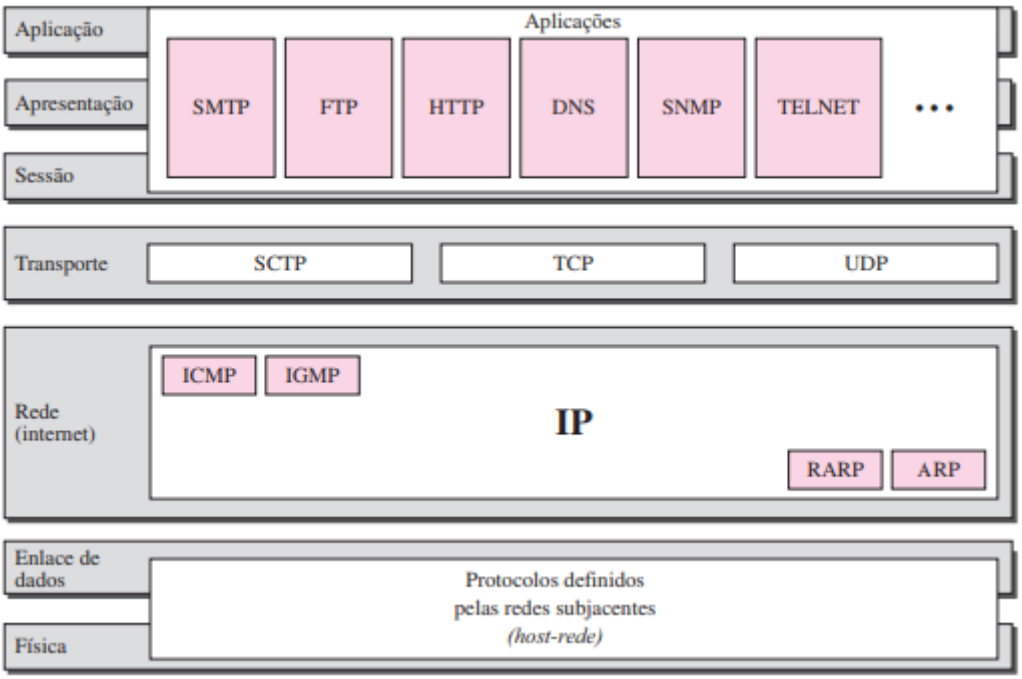


Fonte: **(**MONSETTATE, et.al., 2018).

Já no Modbus TCP, segundo (SOUZA, 2010), as conexões são estabelecidas pela porta 502 através de uma arquitetura tipo cliente/servidor – a qual segue a ideia de pergunta e resposta – e pelo encapsulamento TCP/IP.

Segundo (FOROUZAN, 2008), o complexo de protocolos TCP/IP conta com cinco camadas: física, enlace de dados, rede, transporte e aplicação. Onde as primeiras quatro camadas são responsáveis por disponibilizarem modelos físicos, interface de rede, conexão em rede e postos de transporte que equivalem às quatro camadas iniciais do modelo OSI. Entretanto, as primeiras três camadas do modelo OSI tem sua representação no TCP/IP por apenas uma camada, que é a camada de aplicação, conforme Figura 21.

Figura 21 – TCP/IP e modelo OSI



Fonte: (FOROUZAN, 2008).

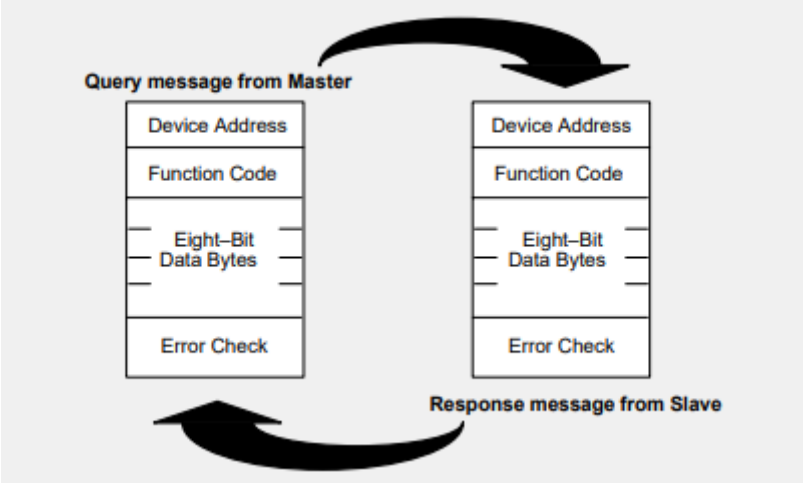
A informação enviada via Modbus RTU tem o formato típico do quadro de mensagem (*Data Frame*) ilustrado na Tabela 2. Na Figura 22 é ilustrado um exemplo de requisição e resposta com o data frame Modbus.

Tabela 2 – Formato do quadro de mensagem modbus

Adress field	**Function field**	**Data field**	**Error check field**
1 byte	1 byte	Variável	2 bytes

Fonte: (MACKAY, et.al., 2004).

Figura 22 – Exemplo de requisição e resposta com o data frame Modbus



Fonte: (MODBUS, 1996).

O primeiro campo da mensagem é responsável pelo endereçamento do dispositivo. Segundo (MODBUS, 1996), o dispositivo declarado como mestre endereça os dispositivos então escravos, esse endereçamento é colocado no campo de endereço da mensagem. Sempre que o escravo enviar a sua resposta, ele coloca o mesmo valor no campo de endereço da mensagem de resposta para que o mestre saiba qual escravo está enviando a mensagem.

Já o byte do campo de função, para (MACKAY, et.al., 2004), diz sobre a função que o dispositivo receptor da mensagem irá realizar. Se o dispositivo de destino for capaz de executar a função solicitada, o campo de função de sua resposta repetirá o mesmo valor da requisição vinda do mestre. De outro modo, o campo de função da solicitação será repetido com seu bit mais significativo igual a 1, declarando assim uma resposta anormal.

Alguns códigos de função Modbus utilizados estão ilustrados na Tabela 3.

Tabela 3 – Códigos de função Modbus

Code	Description	Range
01	Read coils	00001 – 10000
02	Read contacts	10001 – 20000
05	Write a single coil	00001 – 10000
15	Write a multiple coils	00001 – 10000
03	Read holding registers	40001 – 50000
04	Read input registers	30001 – 40000
06	Write single register	40001 – 50000
16	Write multiple registers	40001 – 50000
22	Mask write registers	40001 – 50000
23	Read/write multiple registers	40001 – 50000

Fonte: Adaptado de (THOMAS, 2008).

O campo de dados da mensagem, para Modbus RTU, sempre terá 11 bits e poderá ter as configurações ilustradas na Tabela 4.

Tabela 4 – Configurações do campo de dados do quadro de mensagem Modbus

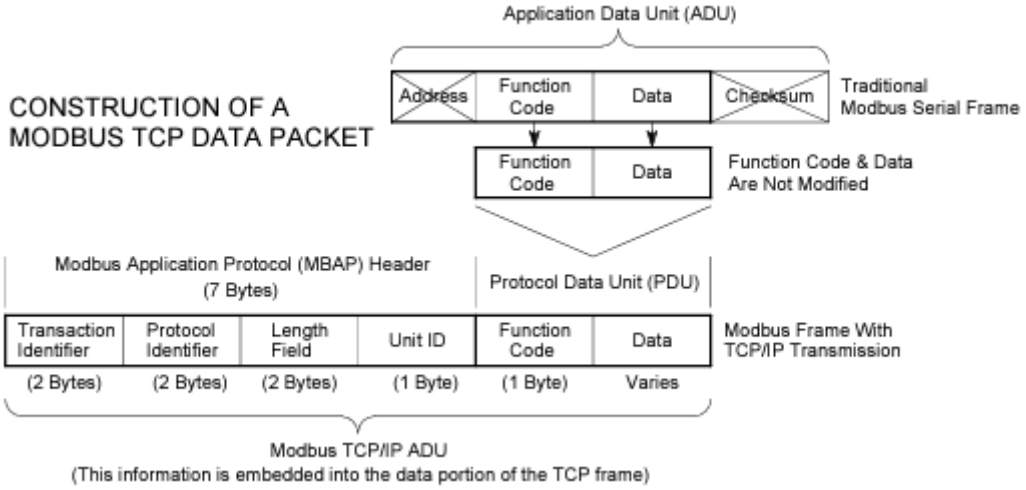
Start	1	2	3	4	5	6	7	8	Parity	Stop
Start	1	2	3	4	5	6	7	8	Stop	Stop

Fonte: Adaptado de (THOMAS, 2008).

O campo determinado para checagem de erro é composto por 2 bytes e segundo (MACKAY, et.al., 2004), para saber qual o valor deste campo é executado uma verificação de redundância cíclica (CRC-16) na mensagem. Isso assegura que os dispositivos não respondam a mensagens que sofreram qualquer tipo de problema durante a transmissão.

Já para modbus TCP, o data frame sofre uma alteração. Além da estrutura do Modbus RTU, há adição do cabeçalho chamado de Modbus *Application Protocol* (MBAP) *Header*, conforme Figura 23.

Figura 23 – Construção do quadro de mensagem Modbus TCP



Fonte: (Acromag, 2015).

O campo inicial do cabeçalho é encarregado de identificar cada solicitação feita pelo cliente, pois nem sempre as respostas recebidas dos servidores chegarão em ordem para o cliente. (IPC2U, 2017).

O segundo campo é para identificação do protocolo e sempre será “0” para o Modbus. (Acromag, 2015). O terceiro campo representa o tamanho dos bytes seguintes, incluindo o campo Unit ID e Data. (MODBUS, 2006).

Por fim, devido à necessidade de comunicação entre as redes Modbus RTU e TCP, torna-se necessário um dispositivo para conversão de dados entre elas, o gateway-conversor tem esse papel, convertendo dados da rede RS485 para a rede de padrão ethernet.

3. Desenvolvimento

Neste tópico, será descrito todo o processo de desenvolvimento para o sistema de controle da demanda de uma indústria, iniciando com o cenário que a empresa se encontra antes da aplicação do sistema de automação, metodologia de desenvolvimento e metodologia de validação.

3.1. Cenário atual

A indústria do agronegócio em questão conta com todo o processo para beneficiamento de sementes, desde secadores até o ensacamento. Sendo assim, existem inúmeros motores na planta, causando um alto consumo de energia.

A empresa consome em seu pátio, em média, 2400 kVA por mês, porém, tem-se a limitação da demanda contratada, de 1700 kVA. Sendo assim, já danificaram e continuam a correr riscos de danificar a infraestrutura instalada e ainda recebem multas constantemente. Vale ressaltar que já solicitaram diversas vezes o aumento da demanda contratada, mas, sem sucesso, uma vez que a concessionária não dispõe de recursos até então.

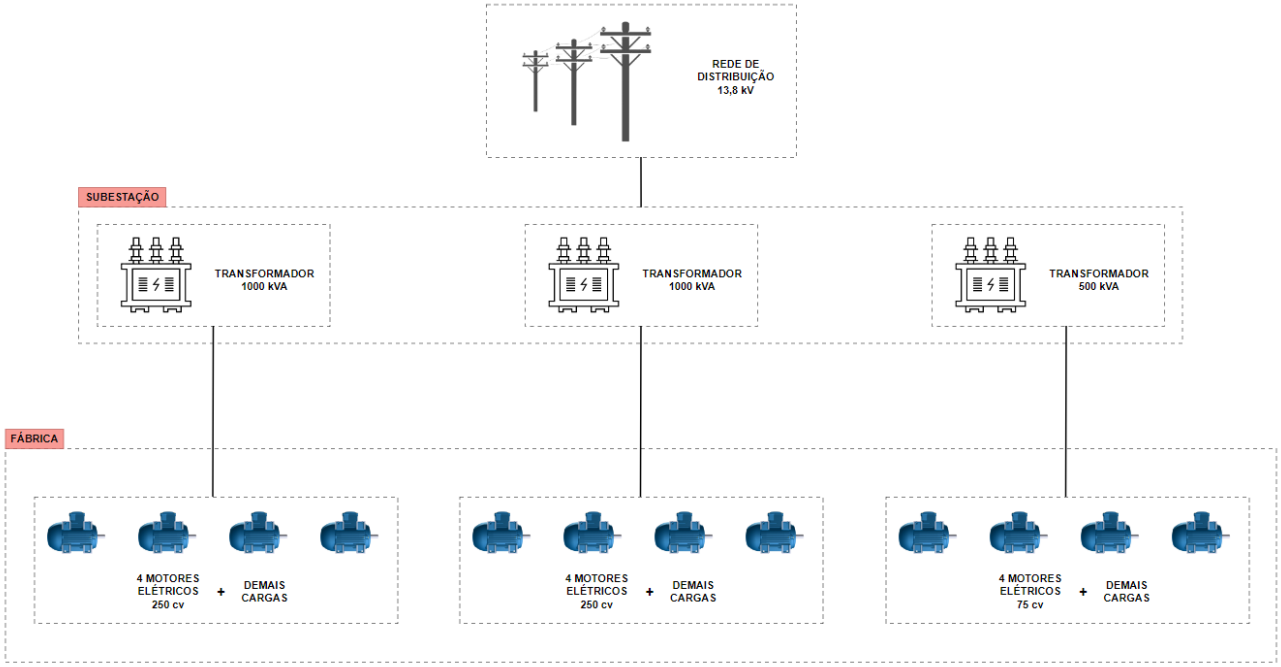
Em momentos de pico de funcionamento, nem mesmo os transformadores instalados no local, dois de 1000kVA e um de 500kVA, suportam a demanda energética e ocorre o desligamento de todas as cargas atreladas ao respectivo transformador que apresentar defeito, o que acarreta negativamente no faturamento e diminui abruptamente a vida útil dos transformadores devido o funcionamento acima de suas condições nominais, chegando a ser necessário a troca, ou reparo, em casos mais graves.

Existem 3 secadores, onde 2 deles contam com 4 motores elétricos de 250 cavalos de potência cada e o último conta com 4 motores elétricos de 75 cavalos de potência, totalizando 12 motores. Cada um dos secadores se alimenta energicamente de um transformador diferente, sendo o secador com motores de menor potência instalado no menor transformador, 500 kVA, e os outros dois secadores consomem energia vinda dos transformadores de 1000 kVA, além das demais cargas da fábrica distribuídas entre os transformadores.

Todos os 12 motores são ligados e desligados via botoeiras pulsantes no painel elétrico instalado nos secadores e cada motor contém um selo feito eletricamente para os manterem ligados. O sistema instalado não conta com nenhum artifício da automação para

otimizar o consumo, sendo somente os motores de cada secador e as demais cargas da fábrica ligadas aos transformadores, como é evidenciado na Figura 24.

Figura 24 – Sistema sem a implementação do controle de demanda



Fonte: Próprio Autor (2023).

3.2. Metodologia de desenvolvimento

Devido ao problema levantado, foi estabelecido uma solução rápida, eficaz e de baixo custo para fazer com que a demanda não ultrapasse os valores máximos estipulados. Com um sistema de automação para controle de demanda, além de evitar multas mensalmente, pode-se prever, e escolher – sempre que a demanda consumida venha a ser extrapolada – o que será desligado e organizar para minimizar os efeitos negativos na produção, além de preservar toda a infraestrutura da fábrica e da concessionária de energia.

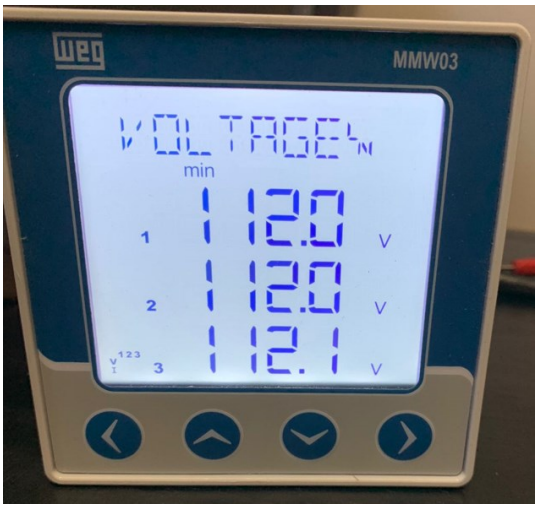
O projeto conta com uma remota Siemens ET 200SP, controlador lógico programável da Siemens, modelo S7-1200 – já existente na unidade – interface homem-máquina Weintek MT8071iP, multimedidores de energia WEG modelo MMW03-M22CH, gateway-conversor Modbus WEG modelo RS485-ETH-N, redes de comunicação Modbus RTU e Modbus TCP e relés com interface 24 V. O sistema atuará, exclusivamente, nas cargas instaladas nos secadores de grãos, uma vez que é onde existem os maiores

motores e, por consequência, maior demanda de energia por carga.

Sendo assim, a maneira para desligamento dos motores em caso de demanda ultrapassada será a instalação de um relé em série na ligação elétrica do selo do motor que se abrirá por um comando vindo do CLP, o que desligará o motor ao quebrar o selo e, por consequência, diminui-se a demanda vinda daquele secador. O motor desligado via automação precisará, para voltar ao funcionamento, que o operador faça o rearme via IHM e, posteriormente, o religue via botoeira no painel elétrico.

Cada um dos transformadores terá instalado um multimedidor de energia MMW03-M22CH da WEG, ilustrado na Figura 25, o qual coleta e fornece todos os dados necessários para o monitoramento e controle de consumo de energia elétrica, sendo realizável a análise a partir deles. Os medidores instalados comunicam via protocolo de comunicação Modbus RTU, como escravos, o que faz possível então a coleta desses dados por um mestre Modbus RTU.

Figura 25 – Multimedidor de energia WEG MMW03-M22CH



Fonte: Próprio Autor (2022).

O mestre Modbus RTU será o gateway-conversor Modbus RS485-ETH-N da WEG, mostrado na Figura 26, sendo também o servidor na rede modbus TCP, convertendo os dados da comunicação serial para a comunicação ethernet.

Figura 26 – Gateway-conversor Modbus WEG RS485-ETH-N

Fonte: Próprio Autor (2022).

O cliente Modbus TCP será o CLP S7-1200, já presente na unidade, e é onde se encontra a lógica desenvolvida para coleta e tratamento dos dados vindo dos multimedidores. O controlador se comunica via ethernet com o gateway através da remota ET 200SP, Figura 28. Por fim, o sistema é operado e monitorado pela interface homem-máquina da Weintek, Figura 27.

Figura 27 – IHM Weintek MT8071iP

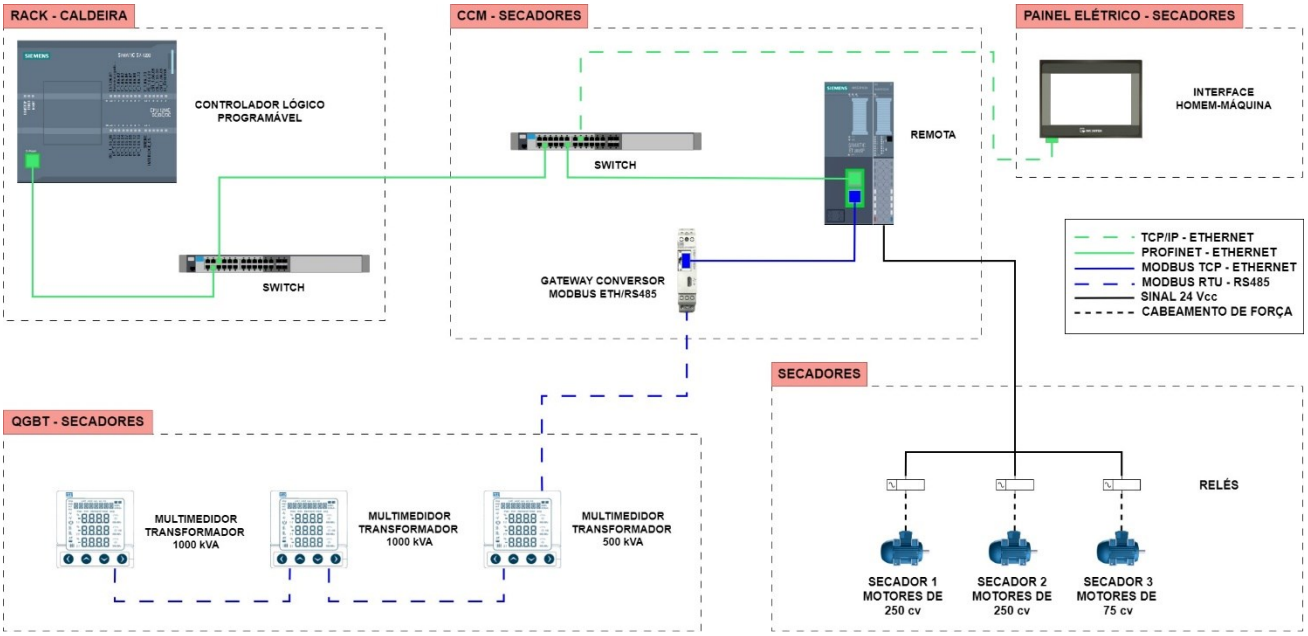
Fonte: Próprio Autor (2022).

Figura 28 – Remota ET 200SP Siemens

Fonte: Próprio Autor (2022).

Após a aplicação do projeto de automação para o controle de demanda, o novo sistema se encontra na Figura 29. Os secadores são ilustrados cada um com um motor por fins didáticos, mas, cada secador é composto por 4 motores – como visto anteriormente – que seguem exatamente a lógica mostrada na imagem a seguir.

Figura 29 – Sistema após implementação do projeto de controle de demanda

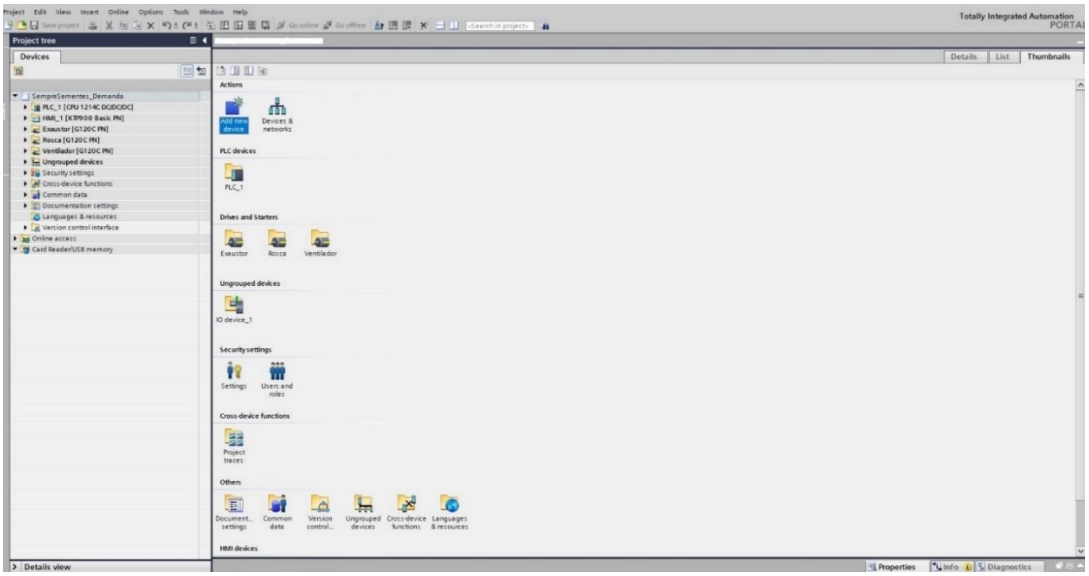


Fonte: Próprio Autor (2023).

3.2.1. Desenvolvimento lógica de controle

Para o desenvolvimento da lógica de programação para o controlador S7-1200, da Siemens, foi utilizado o software TIA Portal versão 16, mostrado na Figura 30.

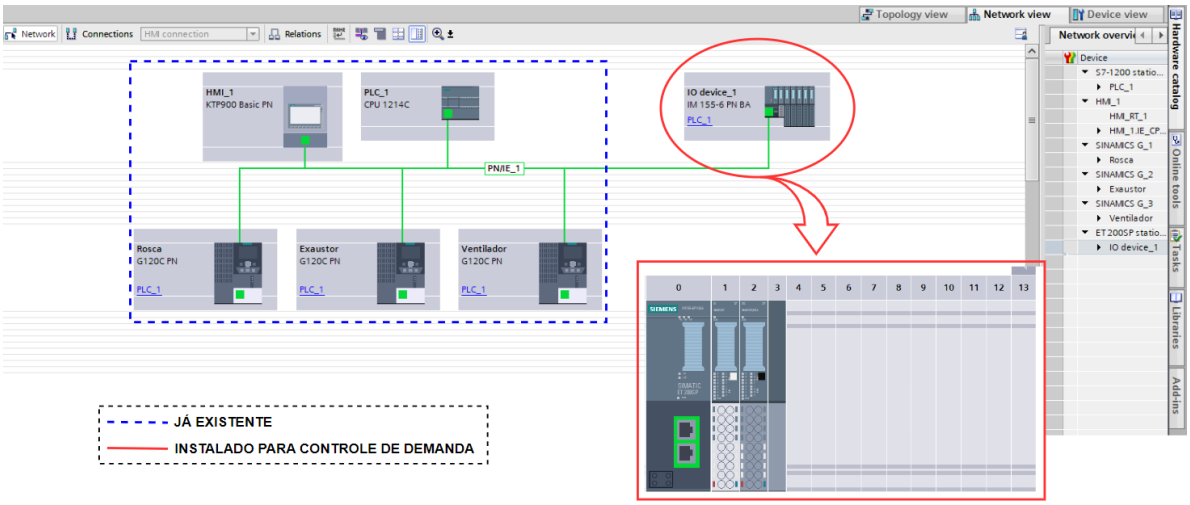
Figura 30 – Interface software TIA Portal v16



Fonte: Próprio Autor (2023).

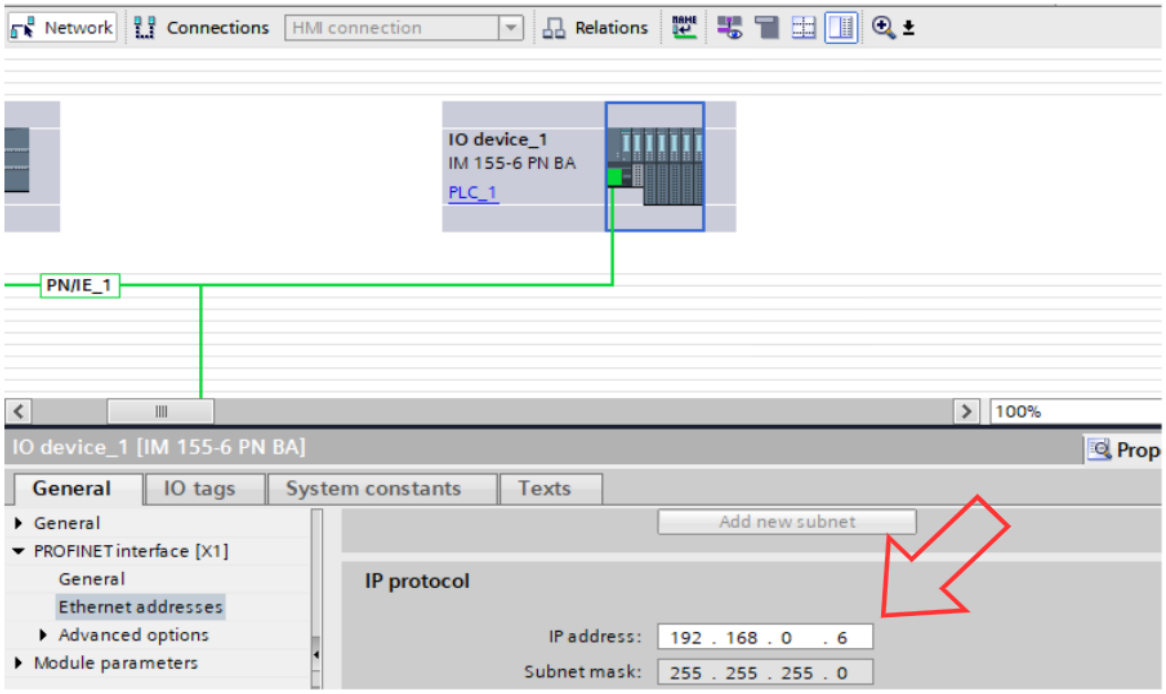
A primeira etapa para o desenvolvimento do software é a inserção e configuração do hardware utilizado direto no TIA. Foi adicionado a remota ET 200SP com um cartão de saídas digitais e um de entradas digitais, o último não foi utilizado para o projeto do controle de demanda, apenas o cartão de saídas. Além disso, a remota foi configurada com o IP 192.168.0.6, dentro da mesma faixa de IP do controlador presente na rede, o qual está configurado com o IP 192.168.0.2. Ambas as configurações são feitas pela aba *Devices & Networks* no TIA Portal e são ilustradas nas Figuras 31 e 32.

Figura 31 – Adição da remota ET 200SP no software TIA Portal



Fonte: Próprio Autor (2023).

Figura 32 – Configuração do IP da remota ET 200SP no software TIA Portal



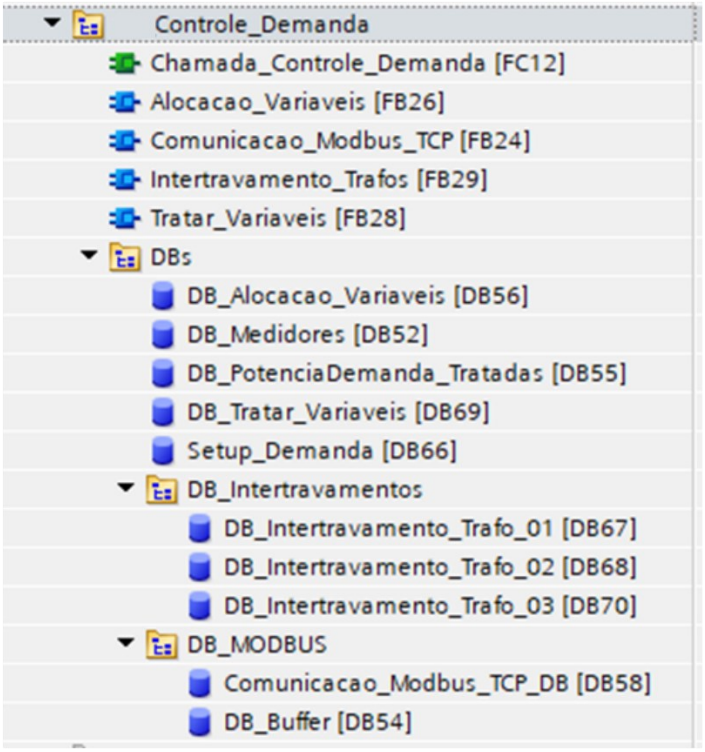
Fonte: Próprio Autor (2023).

O próximo passo para um bom desenvolvimento é planejar quais serão as rotinas executadas na lógica de programação. Neste projeto, serão 4 rotinas, todas desenvolvidas em Ladder e através de um *Function Block* (FB) para cada uma.

Os FB's criados serão executados em uma *Function* (FC), a qual será executada pelo *Organization Block* (OB), do tipo *Program Cycle*, que é executado ciclicamente pelo

controlador. Cada FB traz consigo um *Data Block* (DB) para armazenamento de dados. Os blocos desenvolvidos são estruturados dentro da sessão *Program blocks* no TIA Portal organizadamente, como visto na Figura 33 e descritos na Tabela 5.

Figura 33 – Blocos utilizados no sistema de controle de demanda



Fonte: Próprio Autor (2023).

Tabela 5 – Descrição dos blocos utilizados

Tipo de Bloco	**Sigla do Bloco**	**Nome no Projeto**	**Numeração no Projeto**
Function	FC	Chamada_Controle_Demanda	12
Function Block	FB	Alocacao_Variaveis	26
Function Block	FB	Comunicacao_Modbus_TCP	24
Function Block	FB	Intertravamento_Trafos	29
Function Block	FB	Tratar_Variaveis	28
Data Block	DB	DB_Alocacao_Variaveis	56
Data Block	DB	DB_Medidores	52
Data Block	DB	DB_PotenciaDemanda_Tratadas	55
Data Block	DB	DB_Tratar_Variaveis	69

Data Block	DB	Setup_Demanda	66
Data Block	DB	DB_Intertravamento_Trafo_01	67
Data Block	DB	DB_Intertravamento_Trafo_02	68
Data Block	DB	DB_Intertravamento_Trafo_03	70
Data Block	DB	Comunicacao_Modbus_TCP_DB	58
Data Block	DB	DB_Buffer	54
Organization Block	OB	Main	1

Fonte: Próprio Autor (2023).

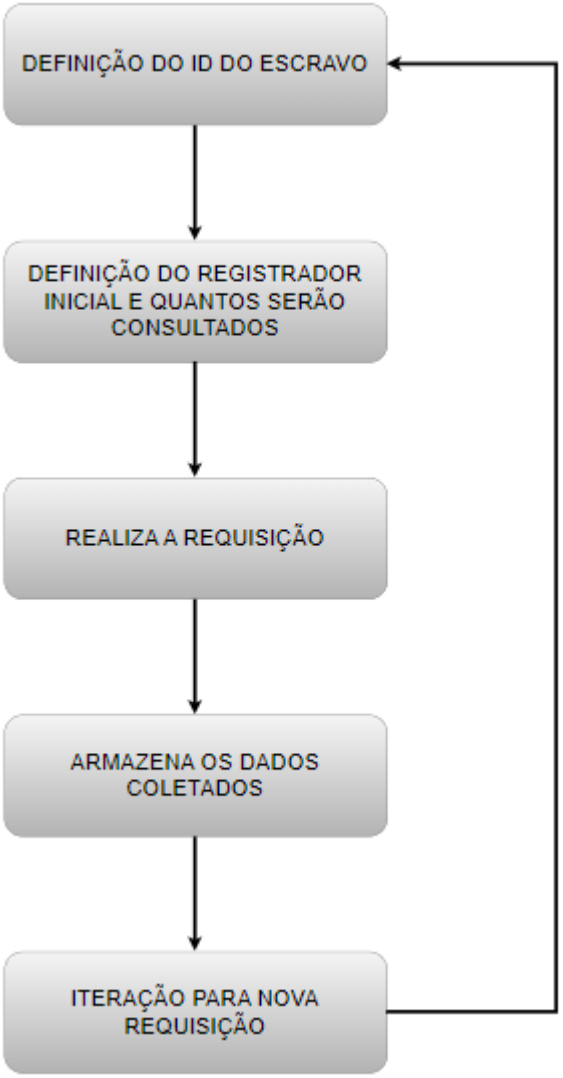
3.2.1.1. Desenvolvimento lógica máquina de estados para leitura Modbus

De acordo com a Figura 29, o cliente Modbus TCP consulta os dados de três escravos em Modbus RTU através do gateway-conversor. O cliente consegue requisitar informações através do IP de cada servidor existente na rede Modbus TCP, uma vez que existem 3 escravos, torna-se necessário o uso de 3 gateway-conversores. Porém, devido ao alto investimento por gateway-conversor, é elaborada uma lógica que o cliente solicite informações de apenas um servidor, mas, receba leituras de três escravos da rede Modbus RTU, economizando então, 2 gateway-conversores. Essa lógica é feita através de uma máquina de estados criada para alterar variáveis internas e externas do bloco de comunicação Modbus *client* da Siemens.

Como cada escravo na rede serial tem um ID exclusivo e a leitura via ethernet terá apenas um IP para consulta, torna-se necessário que o ID do escravo seja alterado de acordo com a requisição que está sendo feita. Por exemplo: se as requisições de 1 a 3 forem do escravo 1, o ID será 1, já se as requisições de 4 a 6 forem do escravo 2, muda-se o ID para 2.

Outrossim, cada requisição feita tem outras particularidades além do ID do escravo, são elas: o registrador Modbus inicial a consultar, quantos registradores serão lidos e onde serão armazenadas as informações coletadas. Sendo assim, essas variáveis estão presentes na lógica da máquina de estados, determinando os seus valores em cada requisição. O fluxograma da lógica está ilustrado na Figura 34.

Figura 34 – Fluxograma para requisições Modbus



Fonte: Próprio Autor (2023).

Para este projeto, o código de função Modbus utilizado será o 03, ou seja, *holding register*, o qual permite que sejam feitas leituras e escritas de 16 bits com endereçamento entre 40001 e 49999. Os registradores consultados foram selecionados através do manual do multimedidor e estão demonstrados na Tabela 6, vale ressaltar que cada grandeza elétrica fornecida pelo multimedidor tem 32 bits, ou seja, utiliza 2 registradores. O número do endereço de leitura feita pelo cliente segue a seguinte estrutura:

$$Endereço\ de\ Leitura\ Modbus = 40001 + Endereço\ do\ Multimedidor \qquad (5)$$

Tabela 6 – Registradores Modbus utilizados do multimedidor WEG

Endereço do Multimedidor	Endereço de Leitura Modbus	Parâmetro	Tipo de dado
0	40001	Tensão 1-N	*float*
2	40003	Tensão 1-2	*float*
4	40005	Corrente 1-N	*float*
20	40021	Tensão 2-N	*float*
22	40023	Tensão 2-3	*float*
24	40025	Corrente 2-N	*float*
40	40041	Tensão 3-N	*float*
42	40043	Tensão 3-1	*float*
44	40045	Corrente 3-N	*float*
66	40067	Fator de Potência	*float*
68	40069	Potência Ativa	*float*
70	40071	Potência Reativa	*float*
72	40073	Potência Aparente	*float*
458	40459	Demanda Potência Aparente Total	*float*

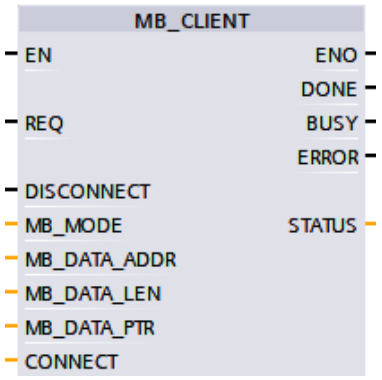
Fonte: Adaptado de (WEG, 2019).

Ainda antes da execução da lógica, é importante definir quantas requisições serão necessárias. Para que o cliente não faça longas leituras de registradores não utilizados neste projeto, serão feitas 5 requisições por escravo, as quais são divididas da seguinte maneira:

- Primeira requisição: registradores 40001 ao 40006;
- Segunda requisição: registradores 40021 ao 40026;
- Terceira requisição: registradores 40041 ao 40046;
- Quarta requisição: registradores 40067 ao 40074;
- Quinta requisição: registrador 40459 ao 40460.

Outro ponto importante, para melhor entendimento do desenvolvimento, é o funcionamento dos parâmetros do bloco MB_CLIENT, Figura 35, utilizado pela Siemens no TIA Portal.

Figura 35 – Bloco MB_CLIENT (TIA Portal)



Fonte: Próprio Autor (2023).

- EN: Habilita, quando em nível lógico alto, o funcionamento do bloco [Bool];
- REQ: Inicia uma requisição Modbus quando em nível lógico alto [Bool];
- DISCONNECT: Finaliza a conexão Modbus quando em nível lógico alto [Bool];
- MB_MODE: Seleciona se o modo da requisição Modbus será escrita ou leitura [USInt], será usado 0, o que indica apenas escrita;
- MB_DATA_ADDR: É o registrador inicial da requisição Modbus [UDInt];
- MB_DATA_LEN: É a quantidade de registradores que serão lidos/escritos [UInt];
- MB_DATA_PTR: É onde serão armazenados os dados coletados;
- CONNECT: É responsável pela configuração da conexão Modbus;
- DONE: Indica que a última requisição foi bem-sucedida quando em nível lógico alto [Bool];
- BUSY: Indica que uma requisição está sendo processada quando em nível lógico alto [Bool];
- ERROR: Fica em nível lógico alto quando ocorre algum erro na requisição [Bool];
- STATUS: Indica qual o erro, caso exista [Word].

Conforme Tabela 5, a FB criada para o desenvolvimento dessa etapa da lógica de programação é a 24. Então, o primeiro passo dentro do *Function block* é declarar todas as variáveis que serão utilizadas, conforme Figura 36.

Figura 36 – Variáveis declaradas na FB24

Static								
ID_Slave	Byte	1	Non-retain	☑	☑	☑	☐	ID do escravo
Requisicao	Int	0	Non-retain	☑	☑	☑	☐	Requisição Atual
MB_CLIENT	Struct		Non-retain	☑	☑	☑	☐	Auxiliar para bloco MB_CLIENT
Done	Bool	false	Non-retain	☑	☑	☑	☐	Done
Busy	Bool	false	Non-retain	☑	☑	☑	☐	Busy
Error	Bool	false	Non-retain	☑	☑	☑	☐	Error
Status	Word	16#0	Non-retain	☑	☑	☑	☐	Status do Erro
MB_DATA_ADDR	UDInt	40001	Non-retain	☑	☑	☑	☐	Registrador inicial
MB_DATA_LEN	UInt	0	Non-retain	☑	☑	☑	☐	Quantidade de registradores
REQ	Bool	false	Non-retain	☑	☑	☑	☐	Booleana para iniciar requisição
BUFFER	Array[0..15] of Real		Non-retain	☑	☑	☑	☐	Armazenamento dos dados coletados
CONNECT	TCON_IP_v4		Non-retain	☑	☑	☑	☐	Variável para configurar conexão Modbus
InterfaceId	HW_ANY	64	Non-retain	☑	☑	☑	☐	HW-identifier of IE-interface submodule
ID	CONN_OUC	1	Non-retain	☑	☑	☑	☐	connection reference / identifier
ConnectionType	Byte	11	Non-retain	☑	☑	☑	☐	type of connection: 11=TCP/IP, 19=UDP (17=TCP/IP)
ActiveEstablished	Bool	1	Non-retain	☑	☑	☑	☐	active/passive connection establishment
RemoteAddress	IP_V4		Non-retain	☑	☑	☑	☐	remote IP address (IPv4)
RemotePort	UInt	502	Non-retain	☑	☑	☑	☐	remote UDP/TCP port number
LocalPort	UInt	0	Non-retain	☑	☑	☑	☐	local UDP/TCP port number
Proxima_Requisicao	Bool	false	Non-retain	☑	☑	☑	☐	Variável auxiliar para borda de subida - indica próxima requisição
Timer_Busy	TON_TIME		Non-retain	☑	☑	☑	☑	Timer para limitar o tempo de processamento de requisição do bloco MB_CLIENT
Requisicao_Anterior	Int	0	Non-retain	☑	☑	☑	☑	Valor da última requisição
Ultimo_Status_Erro	Word	16#0	Non-retain	☑	☑	☑	☐	Armazena o status do último erro

Fonte: Próprio Autor (2023).

A variável “CONNECT” carrega consigo a configuração da comunicação Modbus – como o ID e o tipo da comunicação – e a informação do IP que o cliente buscará a comunicação para troca de dados, neste caso, será o IP do gateway-conversor. Feito isso, é possível iniciar o desenvolvimento da lógica da máquina de estados para a comunicação de um equipamento cliente Modbus TCP com 3 equipamentos escravos Modbus RTU através de um gateway-conversor RS485/ethernet.

O próximo passo é a definição do ID do escravo que será lido na requisição atual. Como são 3 dispositivos na rede RTU e são 5 requisições por cada um deles, o mapeamento será feito de acordo com a Tabela 7.

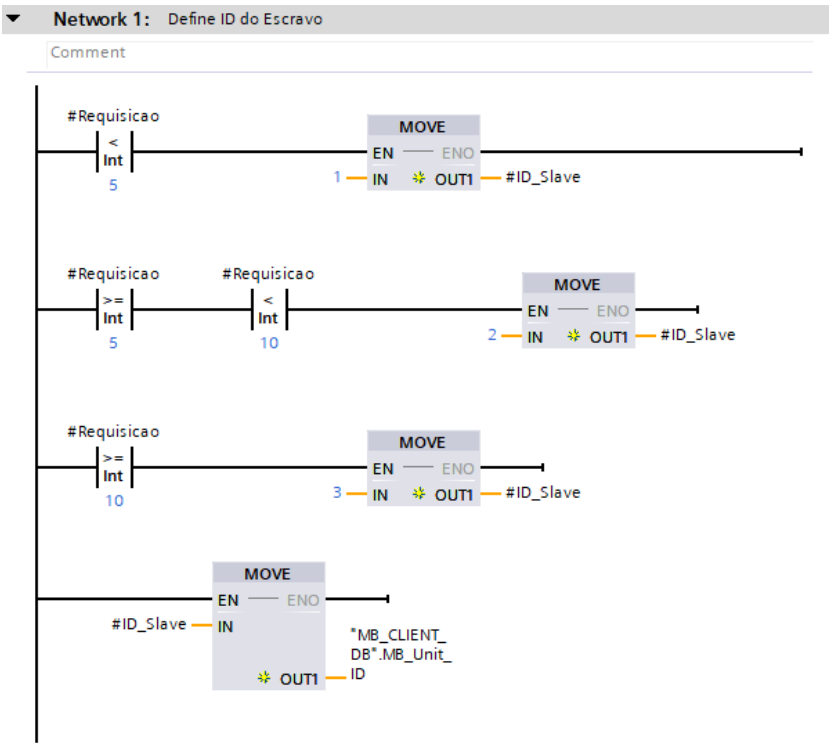
Tabela 7 – Mapeamento de requisições por ID do escravo

ID do escravo	Requisições correspondentes
1	0 a 4
2	5 a 9
3	10 a 14

Fonte: Próprio Autor (2023).

Sendo assim, a lógica para definição do ID é uma condição “SE” para o número da requisição atual e, como consequência, mover o valor correto para a variável “ID_Slave”. Um ponto importante é que a variável do bloco “MB_CLIENT” que define qual o ID de leitura é uma variável interna, assim sendo, é necessário mover o valor armazenado no “ID_Slave” para a variável interna “MB_Unit_ID”. Essa etapa é ilustrada na Figura 37.

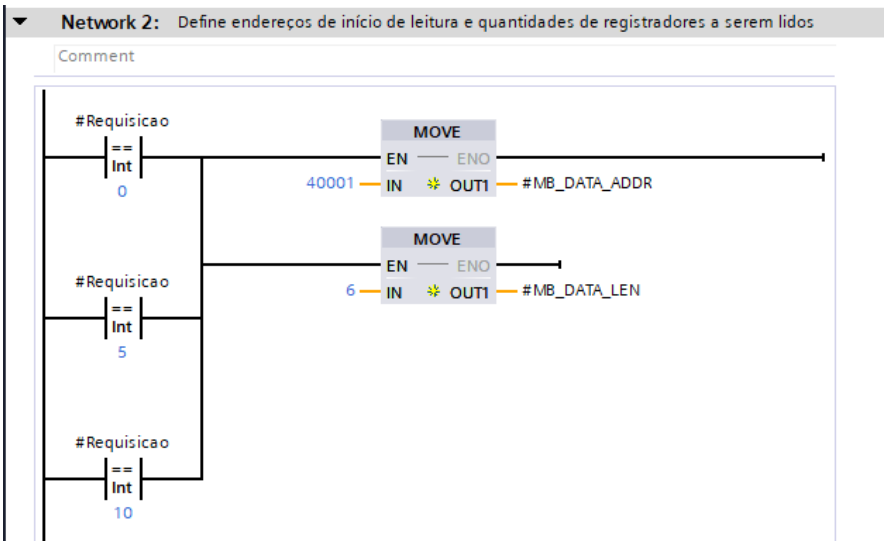
Figura 37 – Lógica para definição do ID do Escravo



Fonte: Próprio Autor (2023).

O segundo passo é a definição do registrador inicial e de quantos serão lidos na requisição atual. De acordo com as Tabelas 7 e 8, sabe-se que os registradores iniciais variam a cada requisição, porém, não variam entre escravos, por exemplo: nas requisições 0, 5 e 10 tem-se o mesmo registrador inicial – 40001 – e a mesma quantidade de registradores consultados – 6 – para qualquer escravo. Sendo assim, a lógica se baseia em: sempre que a requisição for 0 ou 5 ou 10, os valores serão 40001 e 6 para as variáveis “MB_DATA_ADDR” e “MB_DATA_LEN”, respectivamente, como mostrado na Figura 38. Então, essa lógica será replica de acordo com a Tabela 8.

Figura 38 – Lógica para definição do “MB_DATA_ADDR” e “MB_DATA_LEN”



Fonte: Próprio Autor (2023).

Tabela 8 – “MB_DATA_ADDR” e “MB_DATA_LEN” para cada requisição

Requisição	**“MB_DATA_ADDR”**	**“MB_DATA_LEN”**
0, 5 e 10	40001	6
1, 6 e 11	40021	6
2, 7 e 12	40041	6
3, 8 e 13	40067	8
4, 9 e 14	40459	2

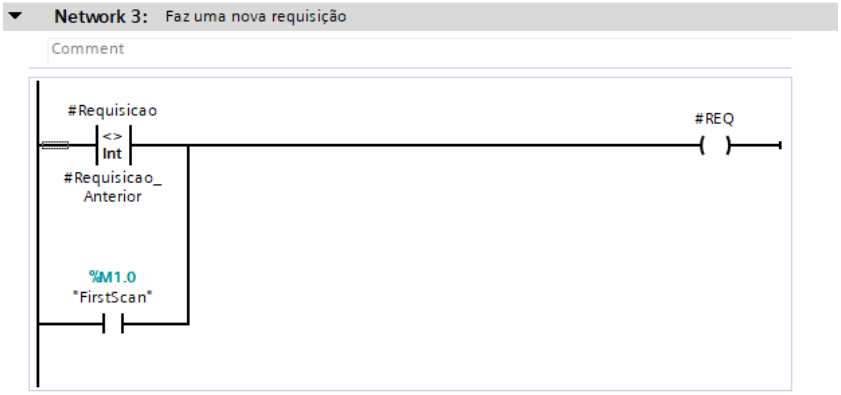
Fonte: Próprio Autor (2023).

Com essas variáveis já definidas, pode-se então executar uma requisição no bloco Modbus, essa requisição sempre é feita devido a duas condições, são elas:

- No primeiro ciclo do CLP;
- Quando a requisição atual for diferente da anterior.

Assim, a Figura 39 ilustra quando será feita uma nova requisição. A iteração de requisições será explicada na última etapa do fluxograma.

Figura 39 – Condições para uma nova requisição



Fonte: Próprio Autor (2023).

Feita a requisição, parte-se para o armazenamento dos dados coletados. As informações são, temporariamente, armazenadas na variável “BUFFER” em todas as requisições, que é um *array* de variáveis do tipo real, com 32 bits cada. Antes que seja feita uma nova requisição, é importante direcionar as leituras guardadas no “BUFFER” para variáveis definitivas, as quais depois serão utilizadas no sistema de supervisão e na lógica de controle e estão declaradas no *Data block* 54, “DB_Buffer”. Conforme Figura 40, as variáveis são separadas por medidor e são criados *array’s* da maneira que o buffer possa ser armazenado por completo a cada requisição.

Figura 40 – DB para armazenamento dos dados coletados via Modbus

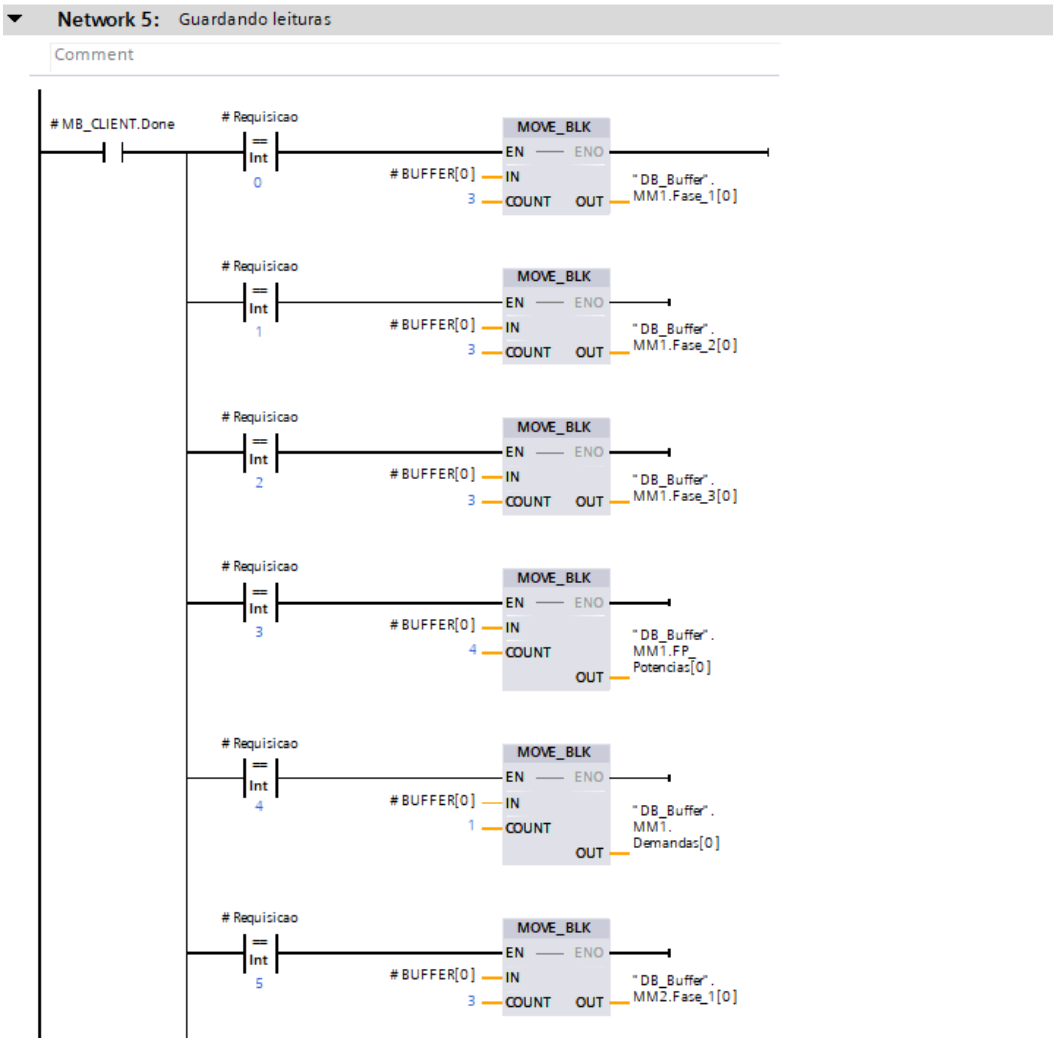
DB_Buffer

	Name	Data type	Start value	Retain	Accessible f...	Writa...	Visible in ...	Setpoint
1	Static							
2	MM1	Struct						
3	Fase_1	Array[0..2] of Real						
4	Fase_1[0]	Real	0.0					
5	Fase_1[1]	Real	0.0					
6	Fase_1[2]	Real	0.0					
7	Fase_2	Array[0..2] of Real						
8	Fase_2[0]	Real	0.0					
9	Fase_2[1]	Real	0.0					
10	Fase_2[2]	Real	0.0					
11	Fase_3	Array[0..2] of Real						
12	Fase_3[0]	Real	0.0					
13	Fase_3[1]	Real	0.0					
14	Fase_3[2]	Real	0.0					
15	FP_Potencias	Array[0..3] of Real						
16	FP_Potencias[0]	Real	0.0					
17	FP_Potencias[1]	Real	0.0					
18	FP_Potencias[2]	Real	0.0					
19	FP_Potencias[3]	Real	0.0					
20	Demandas	Array[0..0] of Real						
21	Demandas[0]	Real	0.0					
22	MM2	Struct						
23	MM3	Struct						

Fonte: Próprio Autor (2023).

Com isso, cada requisição armazena o “BUFFER” em seu respectivo lugar, através do bloco “MOVE_BLK”, que é responsável por mover uma quantidade determinada de variáveis de um array para outro, mantendo a posição das variáveis, por exemplo: caso seja escolhido a transferência de 4 variáveis de um *array*, as posições 0, 1, 2 e 3 no *array* de destino serão os mesmos valores dessas mesmas posições do *array* de origem. Além disso, o armazenamento só acontecerá quando uma requisição for bem-sucedida (“Done”=*True*), evitando que leituras equivocadas sejam guardadas. A lógica está demonstrada na Figura 41 e seguirá o modelo para todos os valores da Tabela 9.

Figura 41 – Lógica para armazenamento dos dados coletados via Modbus



Fonte: Próprio Autor (2023).

Tabela 9 – Armazenamento das variáveis para todas as requisições

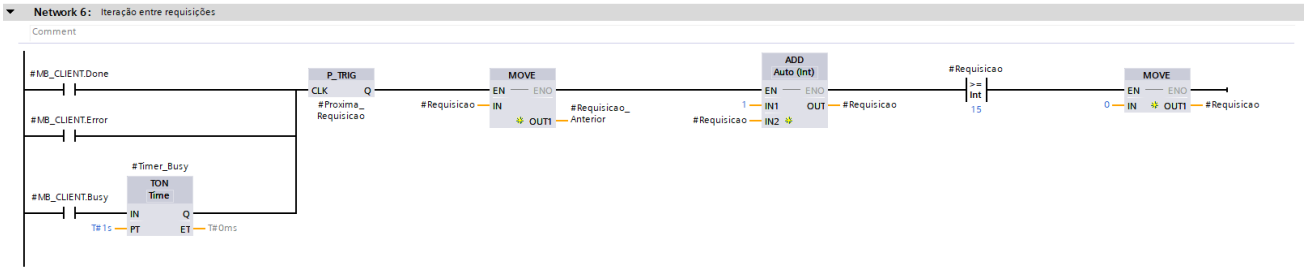
Requisição	Quantidade de variáveis armazenadas	Medidor	Array de destino
0	3	1	Fase_1
1	3	1	Fase_2
2	3	1	Fase_3
3	4	1	Potencias
4	1	1	Demandas
5	3	2	Fase_1
6	3	2	Fase_2
7	3	2	Fase_3
8	4	2	Potencias
9	1	2	Demandas
10	3	3	Fase_1
11	3	3	Fase_2
12	3	3	Fase_3
13	4	3	Potencias
14	1	3	Demandas

Fonte: Próprio Autor (2023).

Finalmente, é necessário que ocorra a iteração do valor da variável “Requisicao” e “Requisicao_Anterior”, para que, a cada ciclo do CLP, sejam alterados os parâmetros de leitura e seja feita uma nova requisição no bloco “MB_CLIENT”.

A lógica funciona da seguinte maneira: Sempre que o bloco finalizar uma leitura com sucesso (“Done” = *True*) ou ocorrer algum erro na leitura (“Error” = *True*) ou o bloco ficar ocupado (“Busy” = *True*) por mais de 1 segundo, é acrescentado 1 ao valor da variável “Requisicao” e armazena-se o seu último valor na variável “Requisicao_Anterior”. Com isso, sempre que há diferença entre essas duas variáveis, o sistema estará apto a fazer uma nova requisição, como visto na lógica ilustrada na Figura 42. Por fim, sempre que o valor de “Requisicao” for maior ou igual a 15, o sistema zera o seu valor e recomeça a contagem.

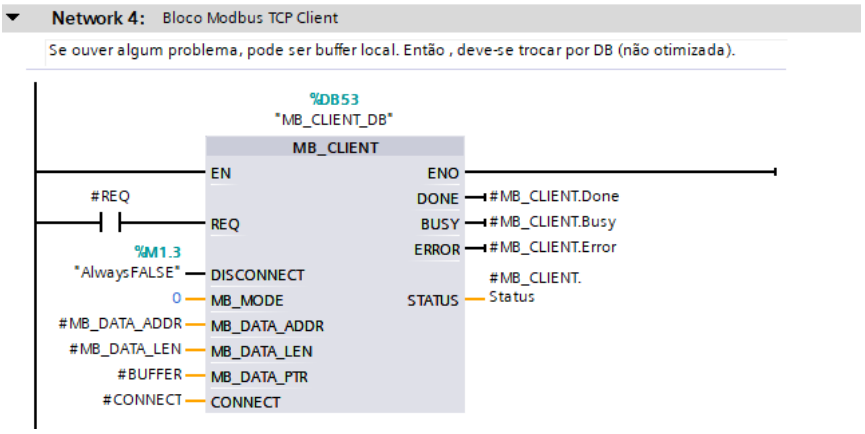
Figura 42 – Lógica de iteração para próxima requisição



Fonte: Próprio Autor (2023).

Portanto, a lógica para comunicação de um *Client* em Modbus TCP com três escravos em Modbus RTU através de uma gateway-conversor se torna possível e confiável com a lógica desenvolvida. O bloco “MB_CLIENT” após o desenvolvimento da lógica é configurado de acordo com a Figura 43.

Figura 43 – Configuração do bloco “MB_CLIENT”



Fonte: Próprio Autor (2023).

3.2.1.2. Lista de entradas e saídas digitais

Antes do desenvolvimento da lógica de controle, é importante definir a lista de entradas e saídas para que o projeto do software coincida com o projeto elétrico que é instalado no painel da automação, neste caso, o projeto conta com apenas saídas digitais. Portanto, a lista é definida conforme Tabela 10.

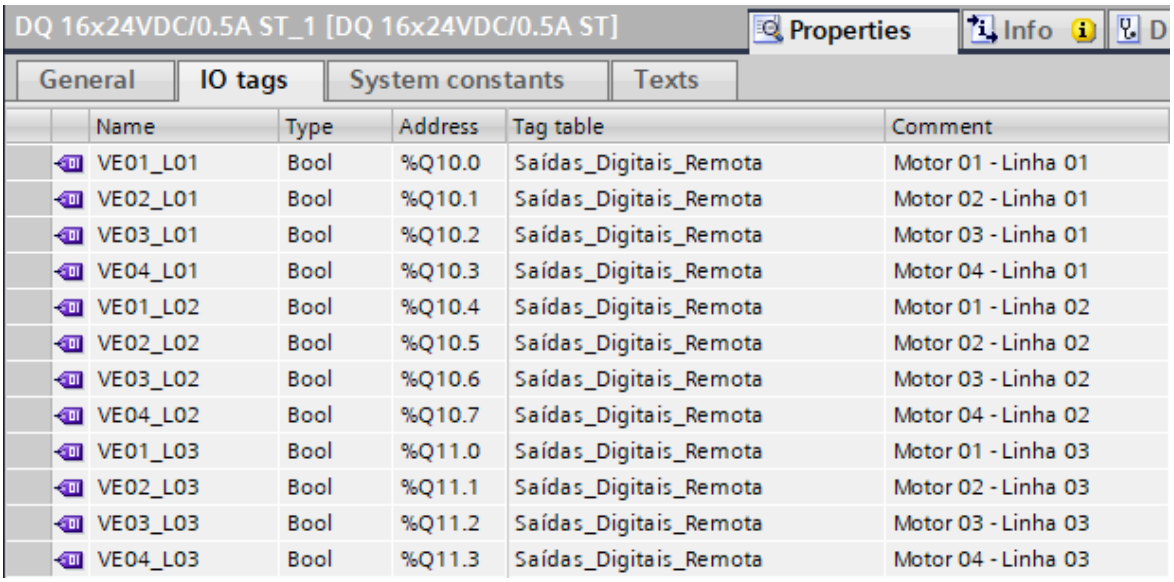
Tabela 10 – Lista de saídas do projeto

Tag	Transformador	Relé	Endereço de Saída CLP	Equipamento
VE01_L01	01	R1	%Q10.0	Motor de 75 cv
VE02_L01	01	R2	%Q10.1	Motor de 75 cv
VE03_L01	01	R3	%Q10.2	Motor de 75 cv
VE04_L01	01	R4	%Q10.3	Motor de 75 cv
VE01_L02	02	R5	%Q10.4	Motor de 250 cv
VE02_L02	02	R6	%Q10.5	Motor de 250 cv
VE03_L02	02	R7	%Q10.6	Motor de 250 cv
VE04_L02	02	R8	%Q10.7	Motor de 250 cv
VE01_L03	03	R9	%Q11.0	Motor de 250 cv
VE02_L03	03	R10	%Q11.1	Motor de 250 cv
VE03_L03	03	R11	%Q11.2	Motor de 250 cv
VE04_L03	03	R12	%Q11.3	Motor de 250 cv

Fonte: Próprio Autor (2023).

Posteriormente, é necessário a identificação no hardware do cartão de saída via TIA Portal, conforme Figura 44.

Figura 44 – Identificação das variáveis no cartão de saída

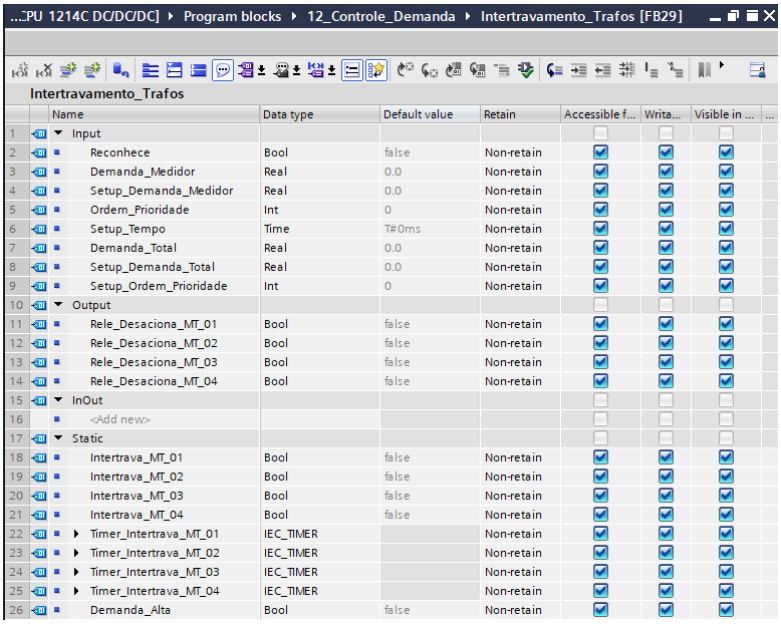


Fonte: Próprio Autor (2023).

3.2.1.3. Desenvolvimento lógica de desligamento dos motores

Feito isso, é iniciado o desenvolvimento da lógica para desarmar os motores através da interrupção de relés inseridos em série com o selo elétrico de cada motor. Assim como o último tópico, é necessário o início do desenvolvimento criando a FB – que neste caso é a FB29, “Intertravamento_Trafos” – e declarando quais as variáveis serão utilizadas, conforme Figura 45.

Figura 45 – Declaração de variáveis da FB29

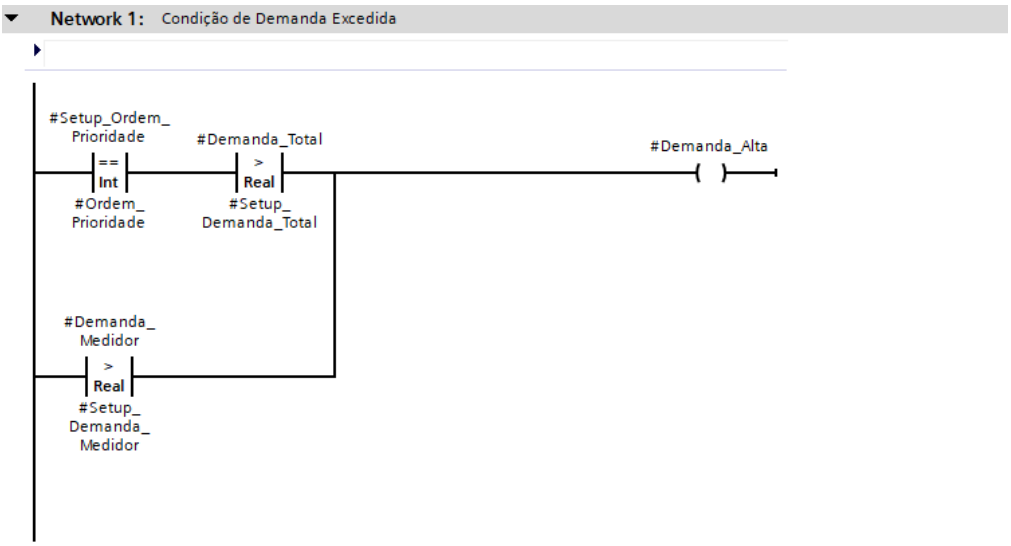


	Name	Data type	Default value	Retain	Accessible f...	Writa...	Visible in ...
1	Input				☐	☐	☐
2	Reconhece	Bool	false	Non-retain	☑	☑	☑
3	Demanda_Medidor	Real	0.0	Non-retain	☑	☑	☑
4	Setup_Demanda_Medidor	Real	0.0	Non-retain	☑	☑	☑
5	Ordem_Prioridade	Int	0	Non-retain	☑	☑	☑
6	Setup_Tempo	Time	T#0ms	Non-retain	☑	☑	☑
7	Demanda_Total	Real	0.0	Non-retain	☑	☑	☑
8	Setup_Demanda_Total	Real	0.0	Non-retain	☑	☑	☑
9	Setup_Ordem_Prioridade	Int	0	Non-retain	☑	☑	☑
10	Output				☐	☐	☐
11	Rele_Desaciona_MT_01	Bool	false	Non-retain	☑	☑	☑
12	Rele_Desaciona_MT_02	Bool	false	Non-retain	☑	☑	☑
13	Rele_Desaciona_MT_03	Bool	false	Non-retain	☑	☑	☑
14	Rele_Desaciona_MT_04	Bool	false	Non-retain	☑	☑	☑
15	InOut				☐	☐	☐
16	<Add new>				☐	☐	☐
17	Static				☐	☐	☐
18	Intertrava_MT_01	Bool	false	Non-retain	☑	☑	☑
19	Intertrava_MT_02	Bool	false	Non-retain	☑	☑	☑
20	Intertrava_MT_03	Bool	false	Non-retain	☑	☑	☑
21	Intertrava_MT_04	Bool	false	Non-retain	☑	☑	☑
22	Timer_Intertrava_MT_01	IEC_TIMER		Non-retain	☑	☑	☑
23	Timer_Intertrava_MT_02	IEC_TIMER		Non-retain	☑	☑	☑
24	Timer_Intertrava_MT_03	IEC_TIMER		Non-retain	☑	☑	☑
25	Timer_Intertrava_MT_04	IEC_TIMER		Non-retain	☑	☑	☑
26	Demanda_Alta	Bool	false	Non-retain	☑	☑	☑

Fonte: Próprio Autor (2023).

A lógica consta com cada saída do cartão do CLP controlando um relé, mantendo-o sempre acionado e interrompendo-o quando ocorrer condição de demanda alta. Caso a demanda esteja alta em um determinado transformador, o sistema desligará as cargas dele, sequencialmente, para que a demanda seja reestabelecida dentro dos limites. Caso a demanda dos 3 transformadores somadas estejam acima do total estipulado – com a demanda por transformador dentro do limite – o sistema conta com uma ordem de prioridade para desligar as cargas do transformador que menos impactará a secagem de grãos, prioridade essa que é escolhida pelo operador. Sendo assim, o primeiro passo da lógica é justamente estipular quando ocorre a condição de alta demanda, conforme Figura 46.

Figura 46 – Lógica para condição de demanda alta

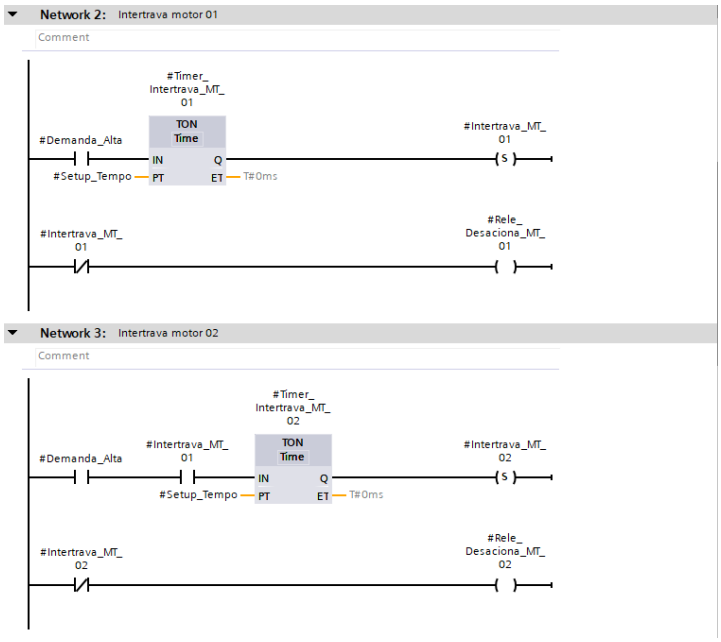


Fonte: Próprio Autor (2023).

O segundo passo é desenvolver a lógica para desacionamento ordenado das cargas, buscando que seja desligado um motor por vez e, caso a demanda seja satisfeita desligando apenas um motor, o sistema reconheça e não desligue nenhum além dele e assim seja caso desligue 2 ou 3 motores.

O sistema conta com um timer do tipo retardo para ligar, onde o tempo para disparo dele é justamente o tempo de setup de desligamento entre cargas, escolhido pelo operador. A lógica segue o mesmo modelo para todos os motores, porém, o motor que será desligado sempre depende se o anterior dele já desligou, por exemplo: caso o motor 2 seja o próximo a desligar, ele só o faz se o motor 1 está desligado e é isso que mostra a Figura 47, onde a variável “Rele_Desaciona_MT_02” sempre está ligada até que a condição de “Demanda_Alta” e o desligamento do motor 1 ocorram. O motor 1 depende somente da alta demanda para desligar, uma vez que ele é o primeiro a ser desligado. Essa lógica é replicada para o motor 3 e motor 4, colocando os intertravamentos dos motores anteriores em série com o do motor 1.

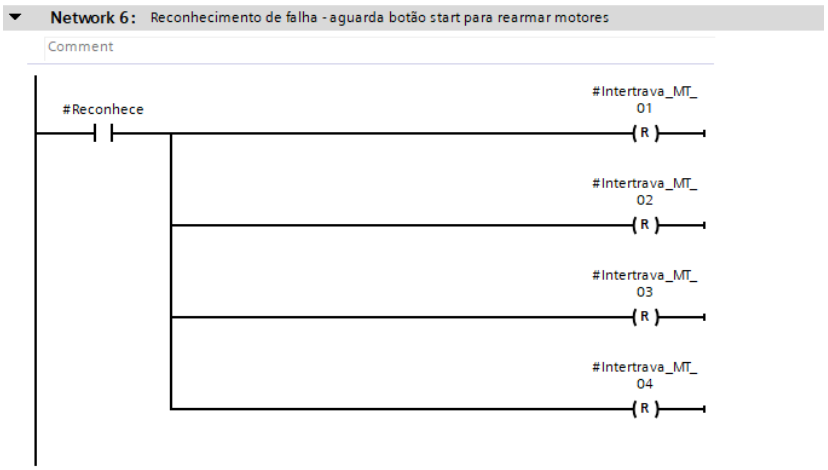
Figura 47 – Lógica para desacionamento de cargas dos dois primeiros motores



Fonte: Próprio Autor (2023).

A última etapa é o reconhecimento da falha, Figura 48. Como a variável de intertravamento recebe um *set*, ou seja, sempre fica ligada após essa ação, é necessário que ocorra o *reset* dela feito pelo operador, liberando, assim, a partida dos motores em falha. Vale ressaltar que o *reset* só é feito caso a condição de demanda alta tenha sido finalizada.

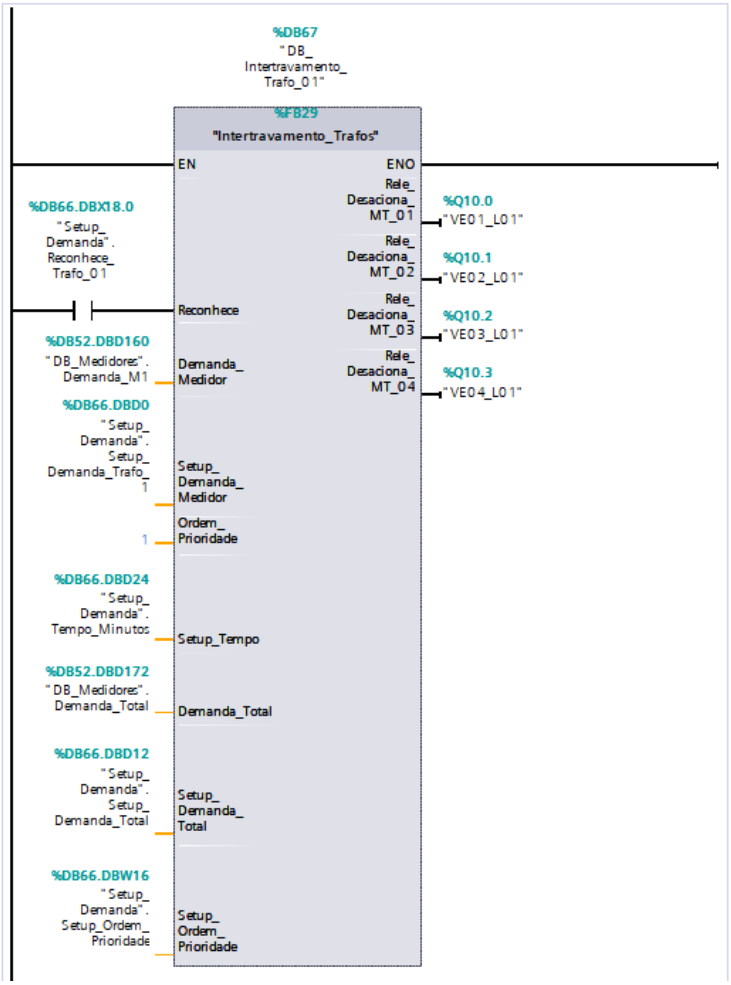
Figura 48 – Reconhecimento da falha dos motores



Fonte: Próprio Autor (2023).

Assim sendo, o bloco de desacionamento de cargas está pronto e já pode ser utilizado para os 3 transformadores, e são chamados conforme Figura 49.

Figura 49 – Bloco com a lógica de desacionamento de cargas para o Trafo 1



Fonte: Próprio Autor (2023).

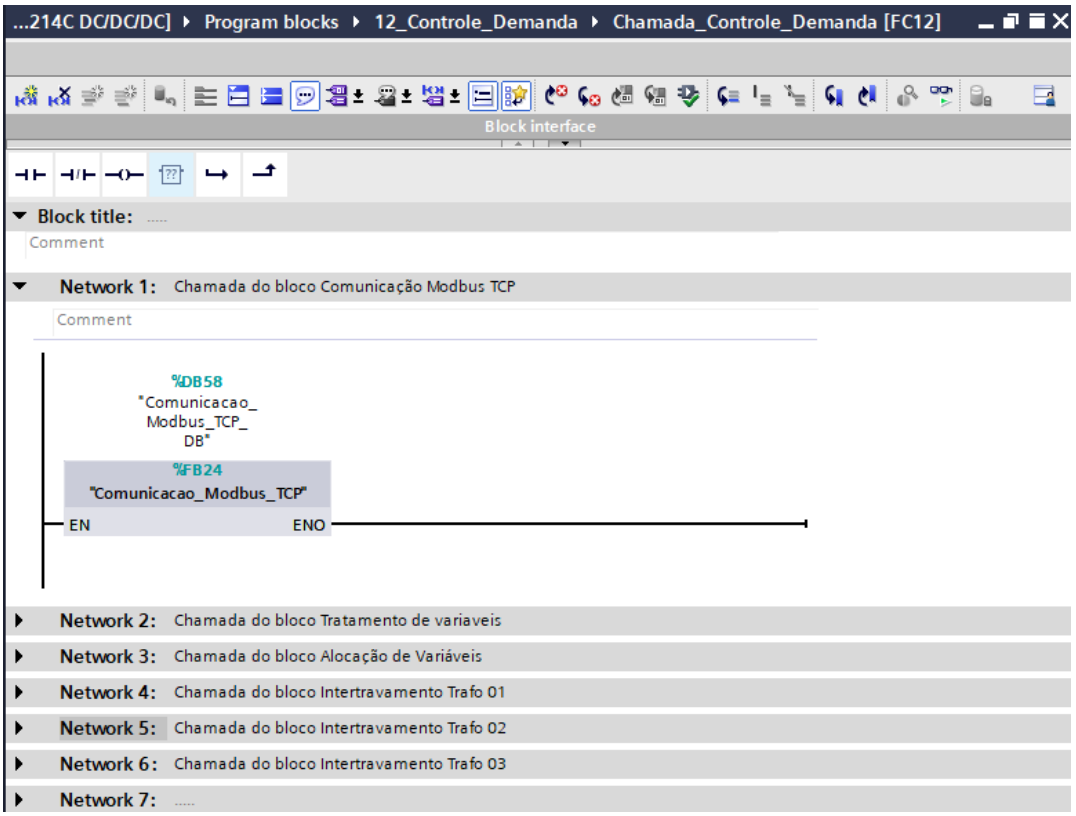
As variáveis de entrada são as medidas vindas via Modbus, as configurações e ações escolhidas pelo operador e a ordem de prioridade daquele transformador. As saídas do bloco são conectadas as saídas físicas do cartão do CLP, as quais estão conectadas aos relés.

3.2.1.4. Chamada dos blocos de função criados

Com os blocos da comunicação Modbus e de desacionamento de cargas concluídos, são feitos dois blocos de apoio antes da chamada de todos para a FC final, são eles: o FB28, para tratamento de variáveis, como a transformação do tempo digitado pelo operador para minutos e o FB26, para alocação das variáveis coletadas e tratadas paras as DB's finais das medições, DB52 e DB55.

Na Figura 50, cada FB criada é adicionada a FC12, "Chamada_Controle_Demanda", e então a lógica para leitura de dados e controle da demanda está pronto para utilização.

Figura 50 – Networks com chamada dos FB's criados



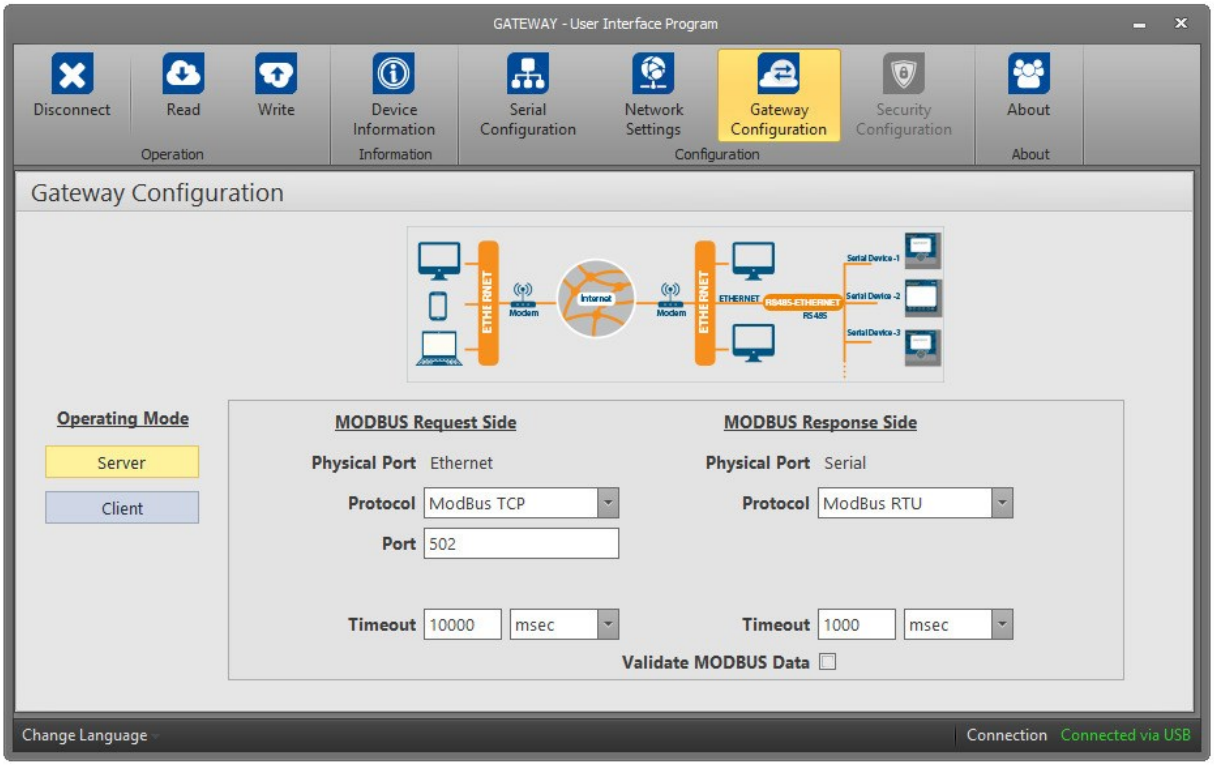
Fonte: Próprio Autor (2023).

Para que o bloco FC12 seja executado ciclicamente, torna-se necessária sua chamada no bloco "Main", pois ele é responsável pela execução no CLP. Feito isso, toda a lógica de controle desenvolvida será executada pelo controlador.

3.2.2. Configuração do gateway-conversor

O gateway é responsável por concentrar os dados da rede serial e da rede ethernet. Para a configuração do gateway da WEG Modbus RS485-ETH-N, é utilizado o software Gateway-master fornecido pela própria empresa desenvolvedora. Nele é parametrizado o IP do equipamento para a rede ethernet (Modbus TCP) e a parametrização Modbus RTU – BaudRate, paridade, stop bit – para a rede serial. Além disso, o equipamento é configurado como mestre na rede Modbus RTU e como servidor na rede Modbus TCP. É possível visualizar a interface do software na Figura 51. Neste projeto, a configuração conta com o BaudRate de 9600, sem paridade e com 1 stop bit.

Figura 51 – Interface do software Gateway-master



Fonte: Próprio Autor (2023).

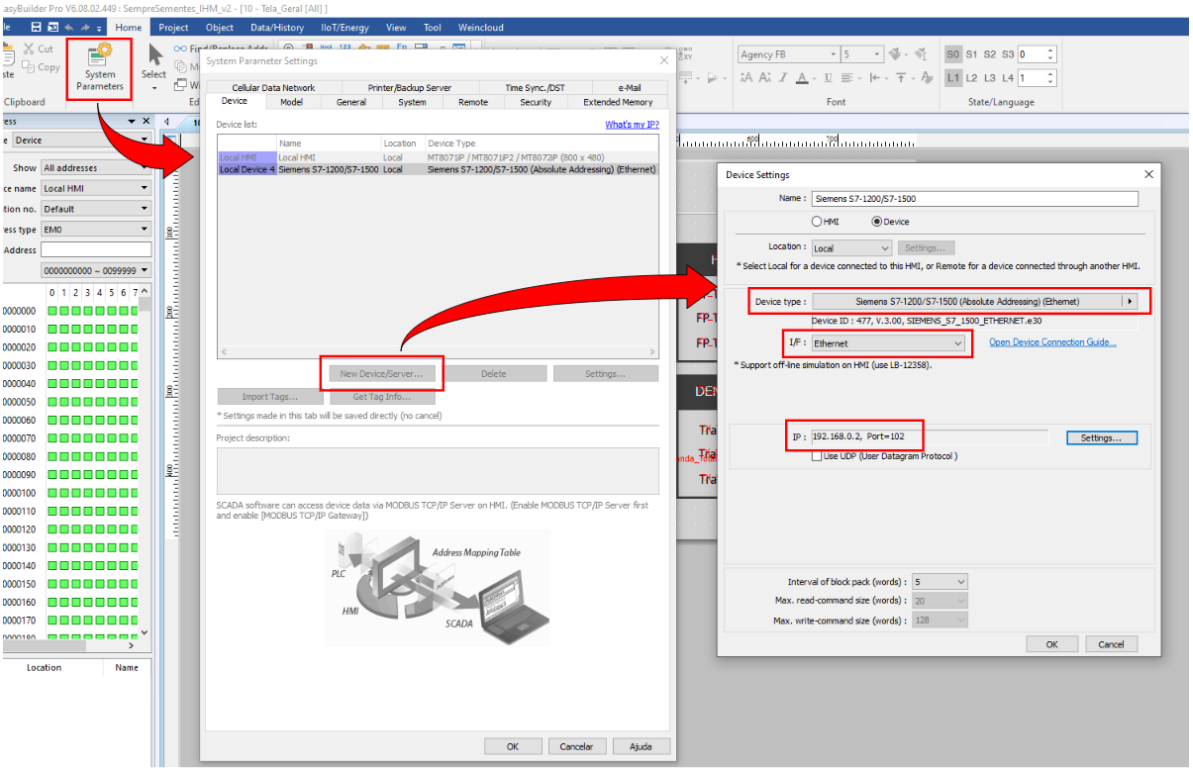
3.2.3. Desenvolvimento das telas de supervisão

Neste tópico, é mostrado o desenvolvimento do sistema de supervisão para monitoramento do projeto implementado. O supervisionamento será feito através de uma interface homem máquina (IHM) da marca Weintek. O software para desenvolvimento é o EasyBuilderPro.

3.2.3.1. Configuração da comunicação entre IHM e CLP

O primeiro passo é configurar a IHM para trocar dados com o controlador da planta, neste caso, um S7-1200 da Siemens. Então, é adicionado o controlador, estabelecido o padrão ethernet como meio de comunicação e selecionado o IP do CLP que a IHM comunicará, conforme Figura 52.

Figura 52 – Configuração da comunicação do sistema supervisório



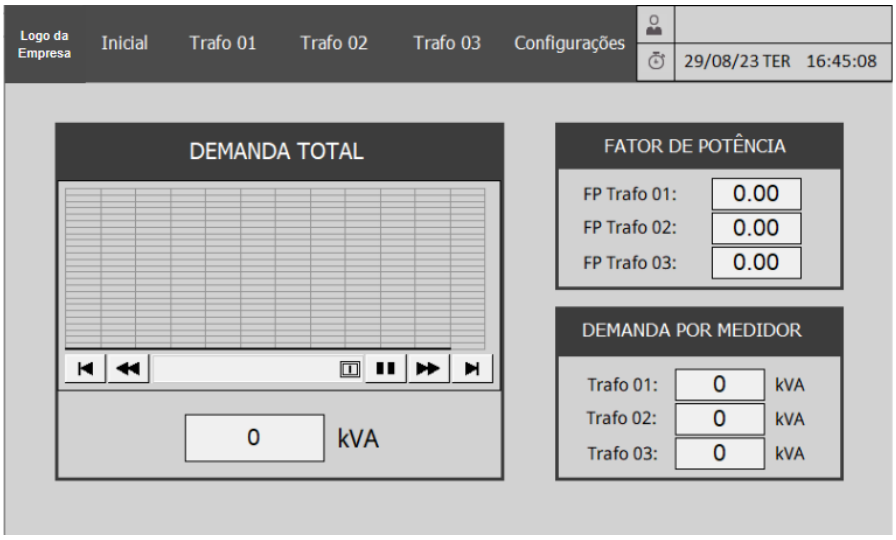
Fonte: Próprio Autor (2023).

O segundo passo é a importação das *tags* do CLP para a IHM, através da opção *Import Tags* disponível no software. É necessário que o CLP esteja ligado e conectado à rede para que a importação seja bem-sucedida.

3.2.3.2. Tela inicial

A tela inicial desenvolvida, ilustrada na Figura 53, conta com um overview da demanda total – através de um gráfico de linha – e da demanda e fator de potência de cada transformador, importante para visualização de todas essas variáveis em um só lugar.

Figura 53 – Tela inicial da IHM

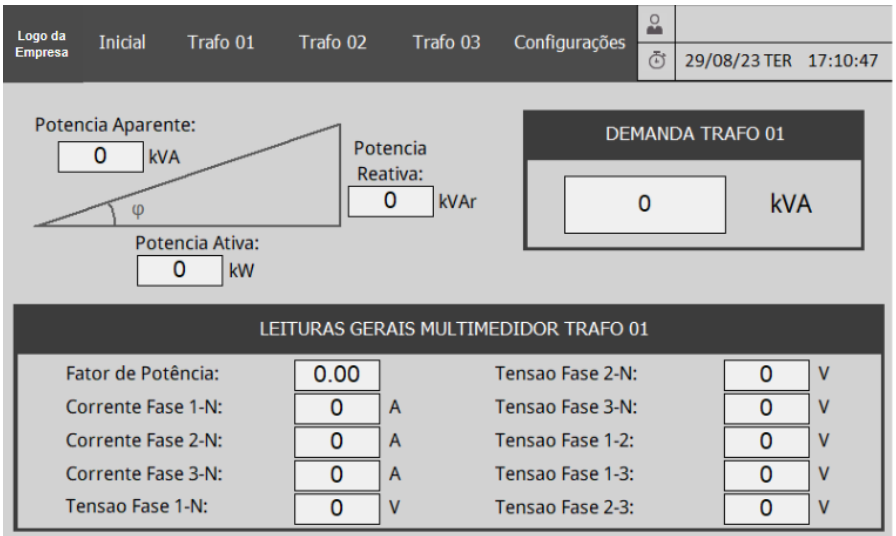


Fonte: Próprio Autor (2023).

3.2.3.3. Tela de cada transformador

Para mais informações e especificidade das leituras, foram desenvolvidas três telas – uma para cada transformador – as quais contam com 14 leituras cada, sendo: potências, demanda, fator de potência, correntes fase-neutro, tensões fase-neutro e tensões fase-fase. A tela correspondente ao transformador 1 pode ser vista na Figura 54 e é replicada para os demais transformadores.

Figura 54 – Tela para medições de um transformador



Fonte: Próprio Autor (2023).

3.2.3.4. Tela de configuração

Para a escolha dos parâmetros do sistema controlado, desenvolve-se a tela de configuração, onde é possível que o usuário digite a demanda limite dos transformadores e da demanda limite total do sistema e, ainda, defina a ordem de prioridade de desligamento das cargas. Por fim, tem-se o botão que abrirá a tela de manutenção do sistema, a qual é mostrada no item 3.2.3.5. A tela é ilustrada na Figura 55.

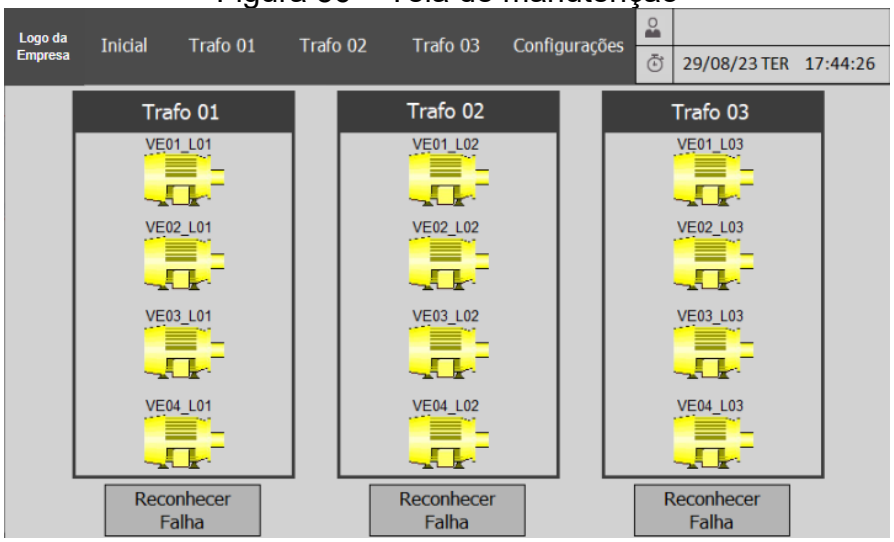
Figura 55 – Tela para configuração do sistema

Fonte: Próprio Autor (2023).

3.2.3.5. Tela de manutenção

Para o supervisionamento dos motores do sistema é desenvolvida a tela de manutenção, onde o usuário monitora quais motores estão prontos para partir – coloração verde – e quais estão em falha – coloração amarela. Além disso, o operador pode reconhecer as falhas do sistema através de um botão, o que deixará o motor pronto para partida caso o relé correspondente àquele motor esteja em condições de funcionamento, ou seja, sem demanda excedida. A tela de manutenção é ilustrada na Figura 56.

Figura 56 – Tela de manutenção



Fonte: Próprio Autor (2023).

3.3. Metodologia de Validação

Devido à privacidade e impossibilidade de coleta dos dados após a implementação do sistema, o projeto desenvolvido é refeito no laboratório para fins de validação dos resultados.

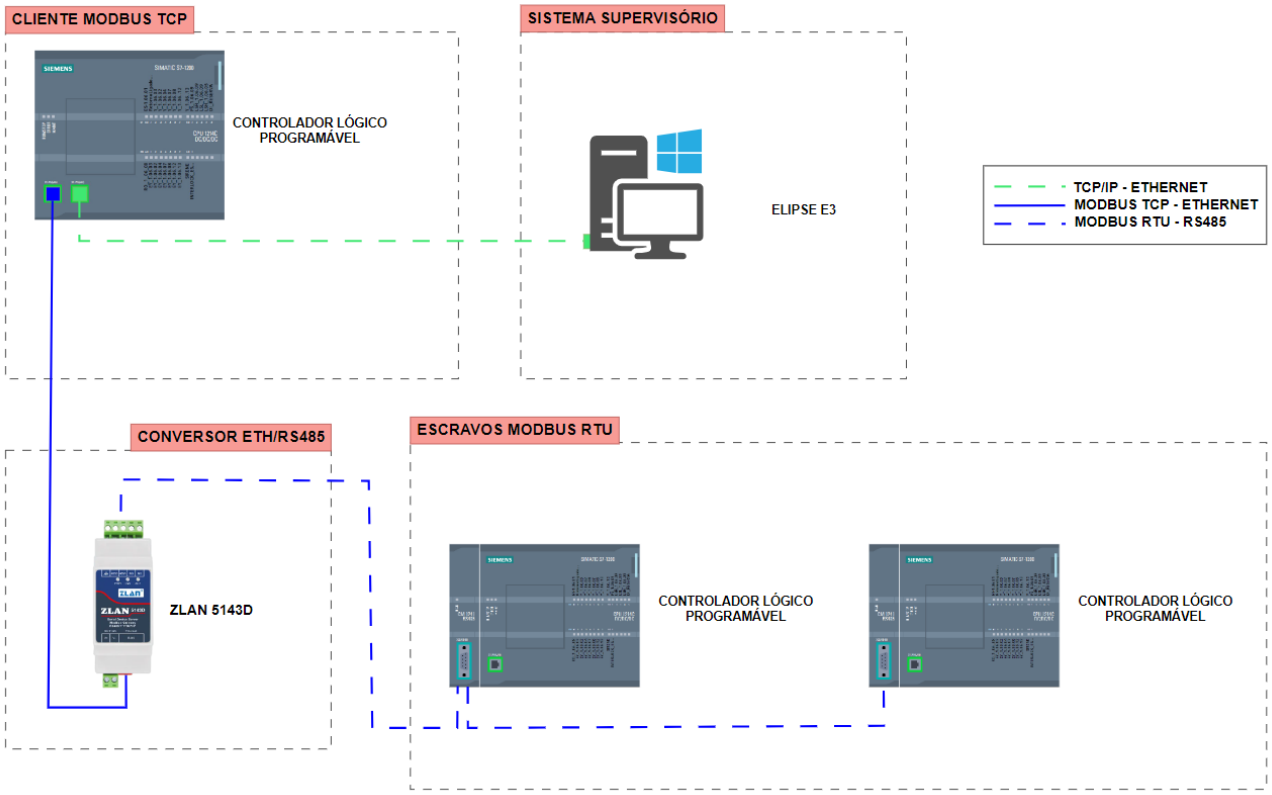
A simulação segue exatamente a lógica do projeto implementado, conta com um sistema supervisório SCADA Elipse E3 e usa o gateway-conversor ZLAN 5143D para comunicação entre o meio serial (Modbus RTU) e meio ethernet (Modbus TCP), mostrado na Figura 57.

Figura 57 – Gateway-conversor ZLAN 5143D

Fonte: Próprio Autor (2022).

Os medidores de energia serão simulados através de controladores lógicos programáveis da Siemens modelo S7-1200 como escravos na rede Modbus RTU. Já o cliente Modbus TCP será outro controlador da Siemens de mesmo modelo. Os motores serão simulados através dos LEDs nas saídas do CLP, conforme Figura 58.

Figura 58 – Sistema de simulação para validação



Fonte: Próprio Autor (2023).

Por fim, as leituras feitas no sistema para validação são simuladas, sendo tensão e corrente lidas através de potenciômetros conectados as entradas analógicas e as demais leituras calculadas internamente em cada CLP escravo, de acordo com as fórmulas descritas no item 2.3. As grandezas utilizadas na simulação são:

- Tensão por fase – leitura simulada via potenciômetro;
- Corrente por fase – leitura simulada via potenciômetro;
- Fator de potência – pré-determinado para cada escravo;
- Potência ativa trifásica – calculada internamente em cada escravo;
- Potência reativa trifásica – calculada internamente em cada escravo;

- Potência aparente trifásica (Demanda) – calculada internamente em cada escravo.

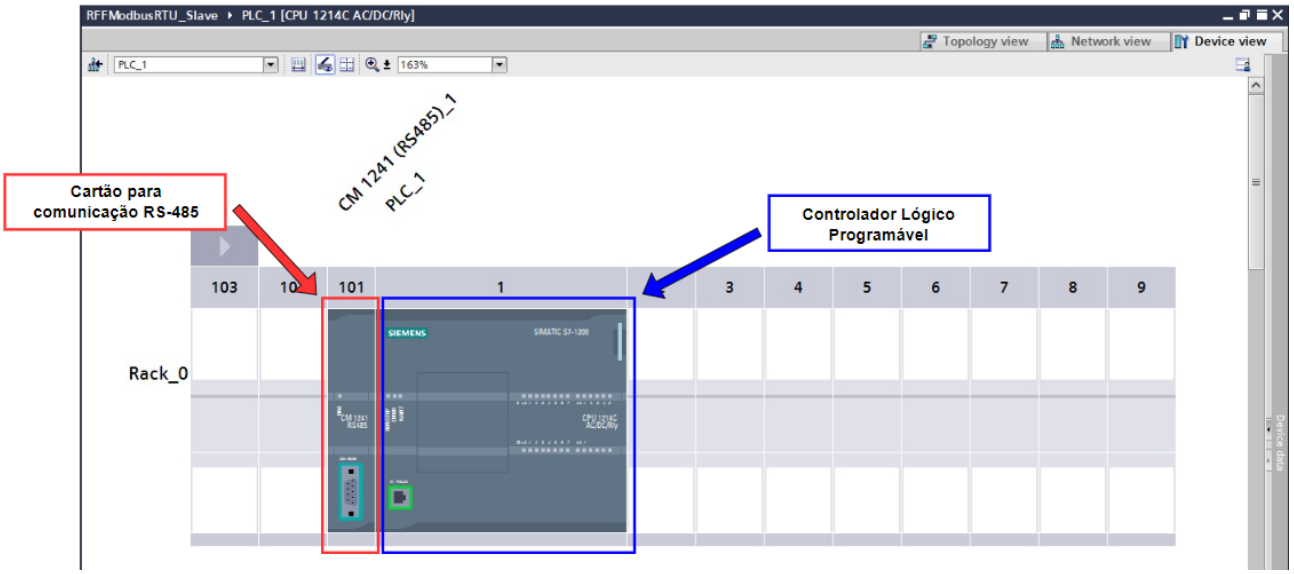
3.3.1. Desenvolvimento da lógica de controle

A lógica desenvolvida para simulação segue os mesmos passos do desenvolvimento do projeto original, com algumas pequenas alterações, como: o número de escravos Modbus, onde o projeto primordial conta com três e a simulação conta com dois escravos e a adição da lógica de acionamento dos motores que simula o painel elétrico do secador. Além disso, cria-se uma lógica para os escravos Modbus RTU.

3.3.1.1. Lógica do escravo Modbus RTU

O CLP escravo na rede Modbus RTU representa o multimedidor do projeto original, sendo assim, é importante que ele forneça as grandezas necessárias para a simulação do sistema. Antes de iniciar a programação, tem-se que selecionar o hardware que é usado, neste caso, apenas o CLP com o cartão CM1241, que permite a comunicação Modbus via RS-485, conforme Figura 59.

Figura 59 – Hardware do CLP escravo em Modbus RTU

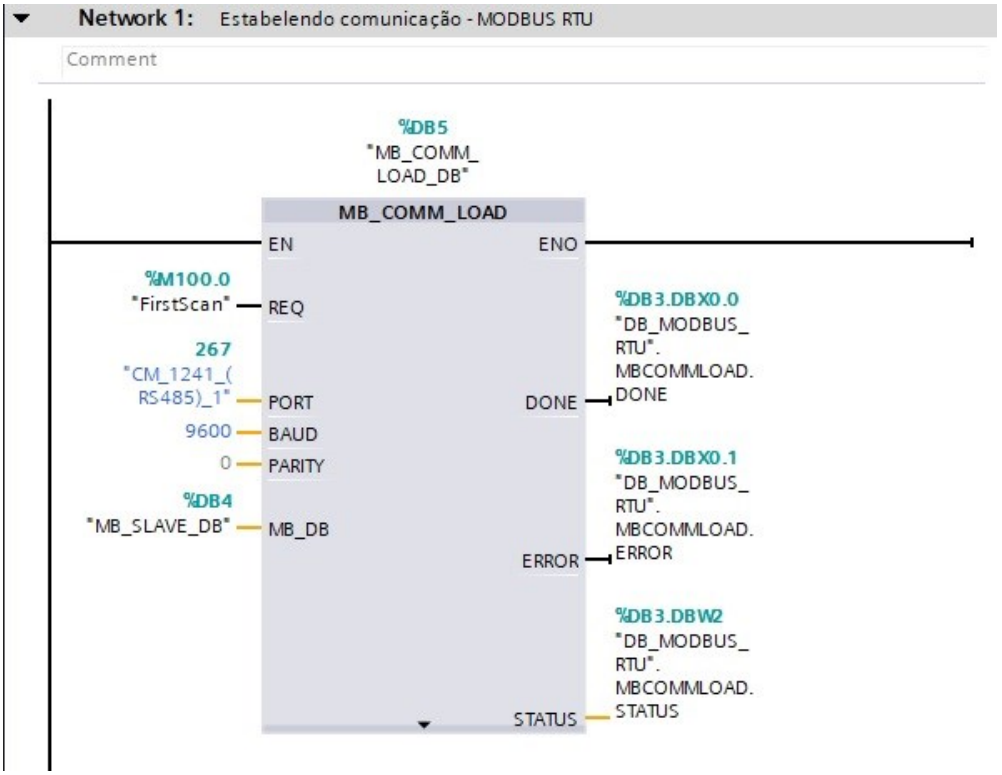


Fonte: Próprio Autor (2023).

Feito isso, é possível executar a lógica de programação do escravo. O primeiro passo

é estabelecer a comunicação Modbus, isso é feito através do bloco “MB_COMM_LOAD”, o qual é ilustrado na Figura 60.

Figura 60 – Bloco “MB_COMM_LOAD”

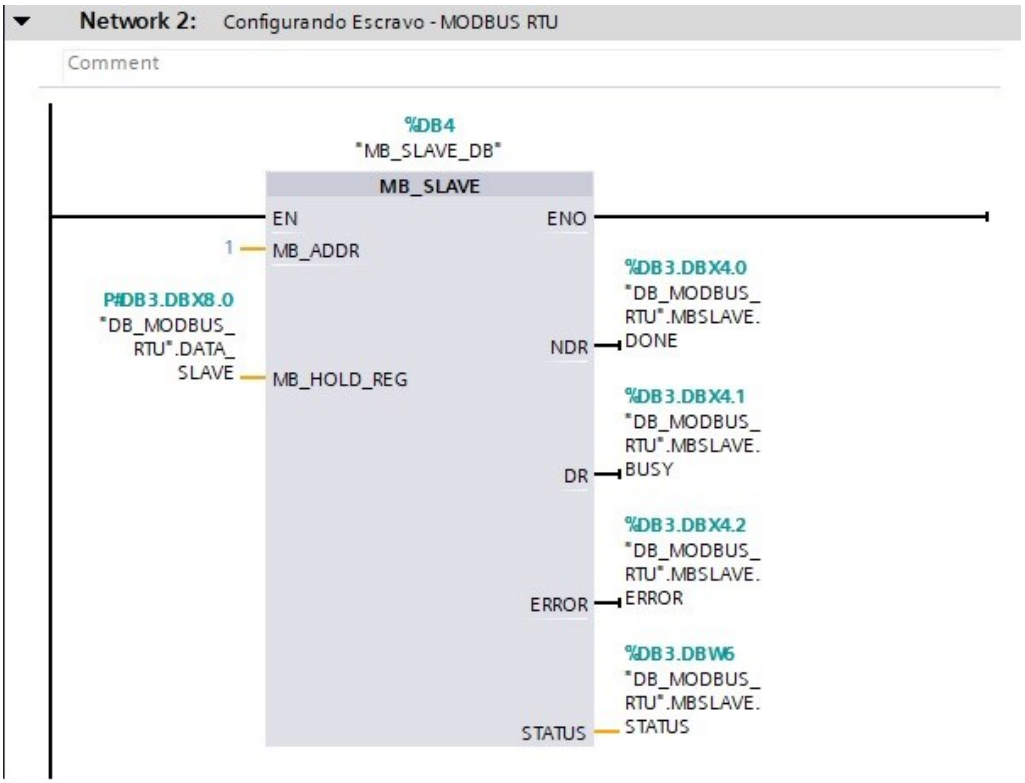


Fonte: Próprio Autor (2023).

Configura-se neste bloco a porta de comunicação RS-485, o BaudRate da comunicação serial, que neste caso vale 9600 e a paridade, que ao defini-la como 0, não há paridade. Além disso, ao editar a variável REQ como *“FirstScan”*, sempre que o CLP ligar o sistema inicializará a comunicação Modbus.

Com a comunicação estabelecida, o próximo passo é realizar a configuração do escravo, Figura 61, o que é feito através do bloco “MB_SLAVE”. Ele é responsável por definir o ID – através da variável “MB_ADDR” – e enviar os dados, neste caso do tipo *holding register*, solicitados pelo mestre – através da variável “MB_HOLD_REG”.

Figura 61 – Bloco “MB_SLAVE”



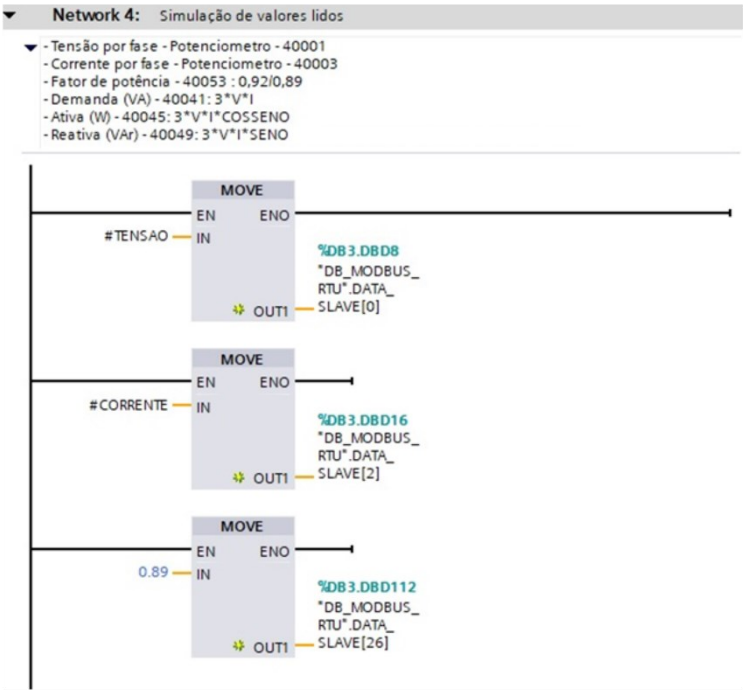
Fonte: Próprio Autor (2023).

O próximo passo do desenvolvimento é a normalização das variáveis lidas analogicamente. Cada potenciômetro fornece leitura de 0 a 10 V, as quais são lidas pelo CLP em uma escala entre 0 e 27648.

Sendo assim, o valor lido é normalizado dentro da escala da Siemens e retorna um valor entre 0 e 1, proporcionalmente. Este valor retornado é colocado dentro da escala desejável, que será entre 0 e 250 para a leitura de tensão e 0 a 2000 para a leitura de corrente.

Com as variáveis devidamente normalizadas, é possível armazená-las em um *data block* para envio ao mestre através da variável “MB_HOLD_REG”. Neste *data block* contém um array com todas as variáveis que serão enviadas, onde, conforme Figura 62, a variável “TENSAO” está armazenada no primeiro elemento do array, “CORRENTE” está no terceiro elemento e define-se um fator de potência para que a simulação possa ser a mais próxima da realidade, o qual está armazenado na posição 26 do *array*. Vale ressaltar que cada variável ocupa duas posições, uma vez que são 32 bits. Esta etapa é possível através do bloco *move*, o qual move valores de uma variável para outra conforme Figura 62.

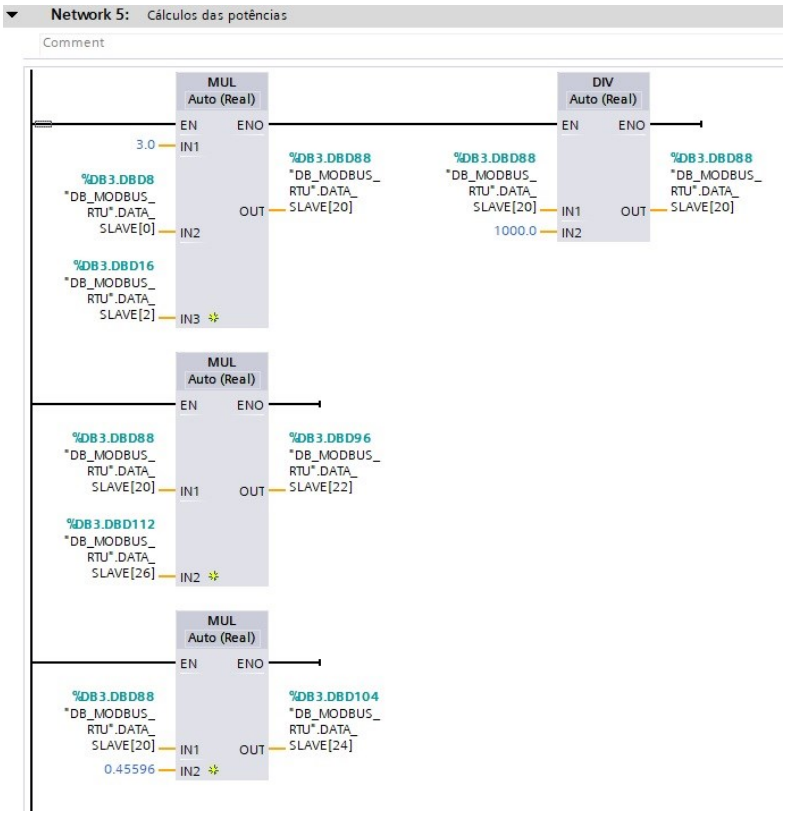
Figura 62 – Alocação de variáveis



Fonte: Próprio Autor (2023).

Por fim, calcula-se as potências (aparente, ativa e reativa) do sistema, de acordo com as fórmulas apresentadas no item 2.3, essa etapa é evidenciada na Figura 63.

Figura 63 – Cálculo das potências para um dos escravos



Fonte: Próprio Autor (2023).

Na primeira linha calcula-se a potência aparente trifásica do sistema, na segunda linha a potência ativa trifásica e, por fim, a potência reativa trifásica do sistema, a qual precisa do valor do seno do ângulo do fator de potência.

3.3.1.2. Lógica do cliente Modbus TCP

Neste tópico, são descritas as etapas do desenvolvimento do software do CLP que se comporta como cliente na rede Modbus TCP, o qual carrega consigo o IP 192.168.1.232.

3.3.1.2.1. Comunicação Modbus

A lógica para comunicação Modbus segue o fluxograma da Figura 34, assim como no projeto primordial. O ponto de alteração é no mapa dos registradores Modbus, os quais foram escolhidos na lógica do escravo e estão evidenciados na Tabela 11.

Tabela 11 – Registradores Modbus utilizados na simulação

Registrador	Variável	Tipo de dado
40001	Tensão	*float*
40005	Corrente	*float*
40041	Potência Aparente	*float*
40045	Potência Ativa	*float*
40049	Potência Reativa	*float*
40053	Fator de Potência	*float*

Fonte: Próprio Autor (2023).

Portanto, as requisições serão separadas em duas por escravo, sendo a primeira entre os registradores 40001 e 40006 e a segunda requisição entre os registradores 40041 e 40054, conforme Tabela 12.

Tabela 12 – Dados para cada requisição

Requisição	ID do escravo	“MB_DATA_ADDR”	“MB_DATA_LEN”
0	1	40001	6
1	1	40041	14
2	2	40001	6
3	2	40041	14

Fonte: Próprio Autor (2023).

Por fim, a lógica segue os mesmos passos executados no item 3.2.1.1.

3.3.1.2.2. Simulação do painel elétrico para acionamento dos motores

Uma vez que os motores no projeto original são ligados e desligados via painel elétrico e o relé que quebra o selo do motor também é instalado em campo, para simulação se torna necessário uma lógica que represente o funcionamento desse sistema. O desenvolvimento conta com o selo e um contato aberto em série representando o relé, conforme Figura 64. A lógica é replicada para todos os 8 motores existentes na simulação.

Cria-se a variável de 32 bits “ACIONA_MOTOR”, onde cada bit representa um elemento na lógica. Na Figura 64, por exemplo, tem-se o bit 0 como o botão liga, bit 1 como o botão desliga, bit 2 representa o relé e o bit 3 representa o motor ligado, por fim, o bit 3 é

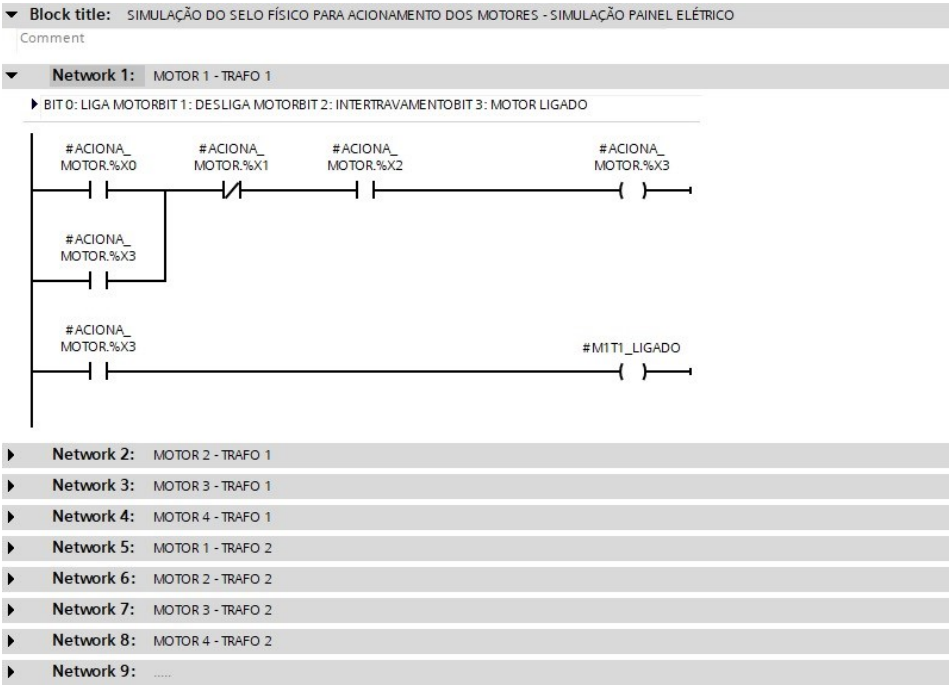
espelhado na variável de saída do bloco para que seja feito o link desta variável com a saída digital do CLP. Na Tabela 13, todos os bits da variável “ACIONA_MOTOR” e suas respectivas funções estão determinadas e seguem o padrão do desenvolvimento da Figura 64.

Tabela 13 – Bits da variável “ACIONA_MOTOR”

Bits	Função
0, 4, 8, 12, 16, 20, 24, 28	Botoeira liga
1, 5, 9, 13, 17, 21, 25, 29	Botoeira desliga
2, 6, 10, 14, 18, 22, 26, 30	Relé
3, 7, 11, 15, 19, 23, 27, 31	Motor ligado

Fonte: Próprio Autor (2023).

Figura 64 – Simulação para acionamento dos motores

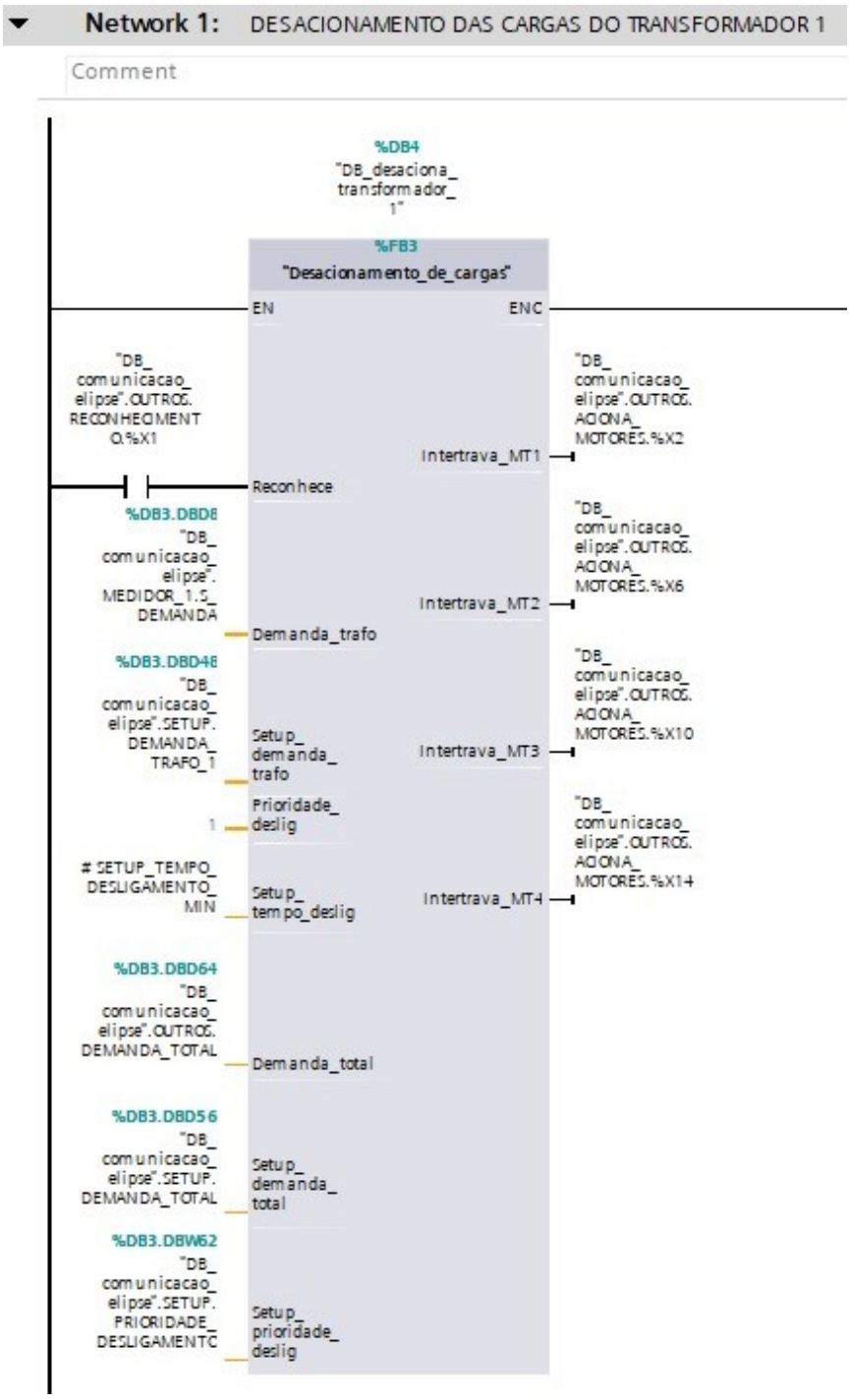


Fonte: Próprio Autor (2023).

3.3.1.2.3. Controle de demanda

O bloco de controle de demanda, Figura 65, foi desenvolvido novamente para a simulação seguindo a mesma lógica do bloco para o projeto original, tem como objetivo manter a demanda do sistema dentro dos limites estabelecidos.

Figura 65 – Bloco de desacionamento de cargas



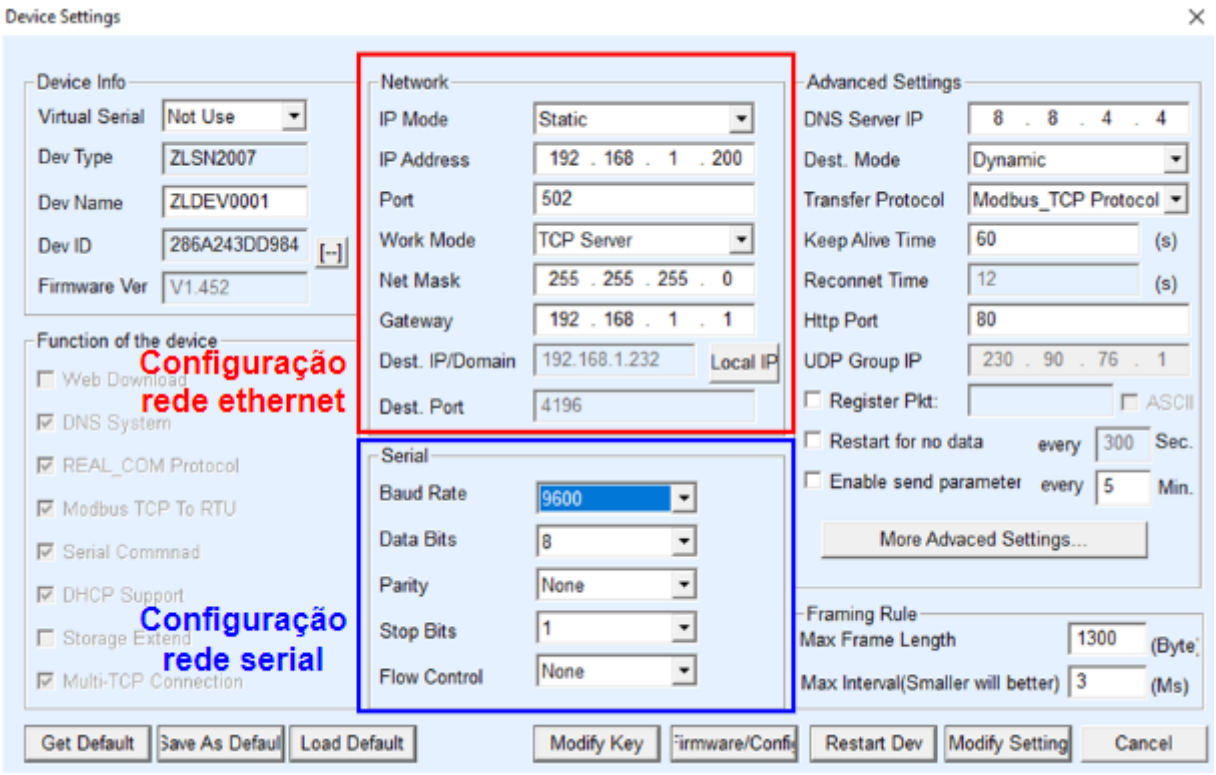
Fonte: Próprio Autor (2023).

3.3.2. Configuração do gateway-conversor da simulação

Para a simulação, é utilizado o gateway ZLAN 5143D e é configurado através do software ZLVirCom. O gateway é configurado como servidor na rede TCP e com o IP 192.168.1.200. Além disso, o IP do destino das informações coletadas por ele é o IP do CLP

cliente da rede TCP, sendo assim, 192.168.1.232. Já para a rede serial, configura-se o Baudrate de 9600, sem paridade e com 1 stop bit, conforme Figura 66.

Figura 66 – Configuração do gateway através do software ZLVirCom



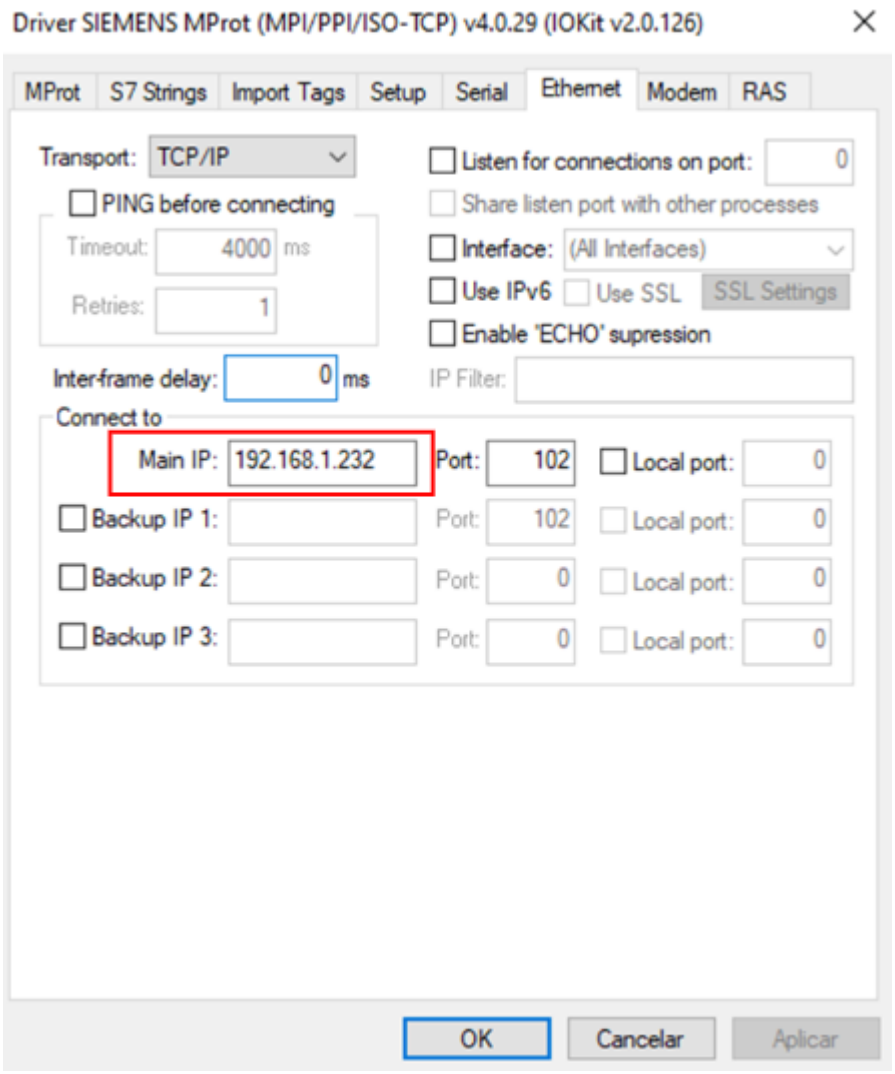
Fonte: Próprio Autor (2023).

3.3.3. Desenvolvimento sistema supervisório

Para a supervisão do sistema simulado, é desenvolvido o projeto no software Elipse E3. As telas desenvolvidas foram as mesmas do projeto ancestral, com o adendo da simulação do painel elétrico na tela de manutenção. Antes do desenvolvimento das telas, é importante a configuração do driver de comunicação, neste caso, para comunicação entre o Elipse E3 e o CLP da Siemens, S7-1200.

O driver responsável por essa tarefa é o MProt. Uma vez que a comunicação será via ethernet, a configuração do driver se dá exatamente na aba ethernet, onde pode-se configurar qual o IP do equipamento que o supervisório comunicará, conforme Figura 67.

Figura 67 – Configuração do driver MProt



Fonte: Próprio Autor (2023).

Posto isso, é necessário declarar todas as variáveis que serão utilizadas no projeto e fazer o link com a respectiva variável do CLP, neste caso, todas as variáveis do CLP que serão utilizadas no sistema supervisório são vindas de um *data block* e a estrutura para declarar este tipo de variável no Elipse segue o seguinte padrão:

$$< [Device:] >< Area >< [Type] >< Address > [.Bit] \quad (6)$$

Sendo assim, o primeiro parâmetro é o número que representa o device, neste trabalho existe apenas um device, usa-se então o valor zero para este parâmetro de todas as variáveis. O segundo campo representa qual a área do dado, ou seja, qual o tipo de arranjo de dados, se é entrada física, saída física ou um *data block*. Para este projeto, todas

as variáveis estão dentro de um *data block*, portanto, usa-se a sigla DB e, logo após o número da DB que está sendo consultada.

O terceiro parâmetro representa qual o tipo do dado, por exemplo, se é float, int ou booleano, para cada tipo de dado existe uma sigla que segue a Tabela 14. Por fim, coloca-se o endereço da variável, neste ponto, é válido ressaltar que, para um data block no software TIA Portal comunicar-se com o Elipse E3, é crucial que ele não esteja otimizado, o que gera um valor de *offset* para cada variável da DB, e esse *offset* é justamente o endereço que é usado para declarar a variável.

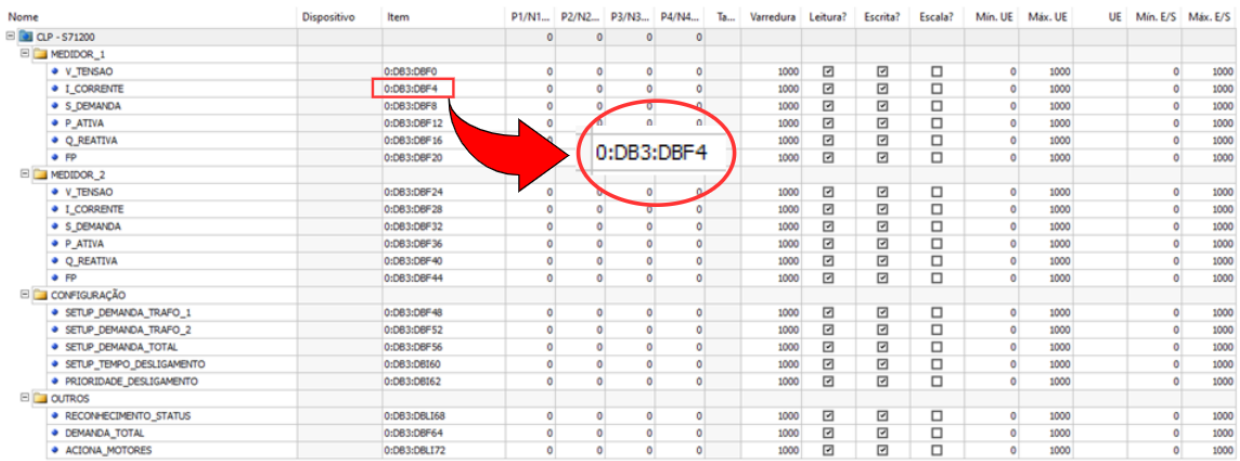
Na Figura 68, a segunda variável declarada – e marcada de vermelho na Figura – está localizada no *device* 0, *data block* 3 e é um dado do tipo float com endereço (*offset*) igual a 4.

Tabela 14 – Siglas para declaração de variáveis vindas de uma DB no Elipse E3.

Tipo de dado	**Sigla utilizada no elipse E3**
Bit	DBX
Byte	DBB
Float	DBF
Int	DBI
Dint	DBLI

Fonte: (Elipse, 2022).

Figura 68 – Declaração de variáveis no Elipse E3



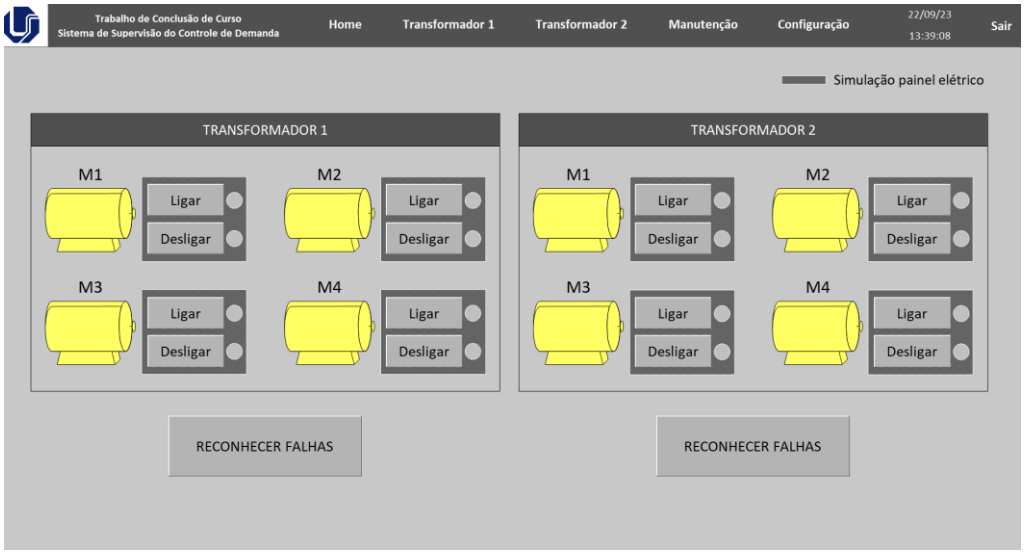
Fonte: Próprio Autor (2023).

3.3.3.1. Telas de supervisão

Assim como no projeto original, o sistema de supervisão foi dividido nas seguintes telas: inicial, monitoramento de cada transformador, configuração e manutenção. Com exceção da última, todas as demais foram desenvolvidas iguais às telas já descritas neste trabalho.

A tela de manutenção, além da indicação dos motores e do botão para reconhecimento de falha, conta com a simulação do painel elétrico, onde é possível ligar e desligar cada um dos motores existentes, conforme Figura 69.

Figura 69 – Tela de manutenção do sistema simulado



Fonte: Próprio Autor (2023).

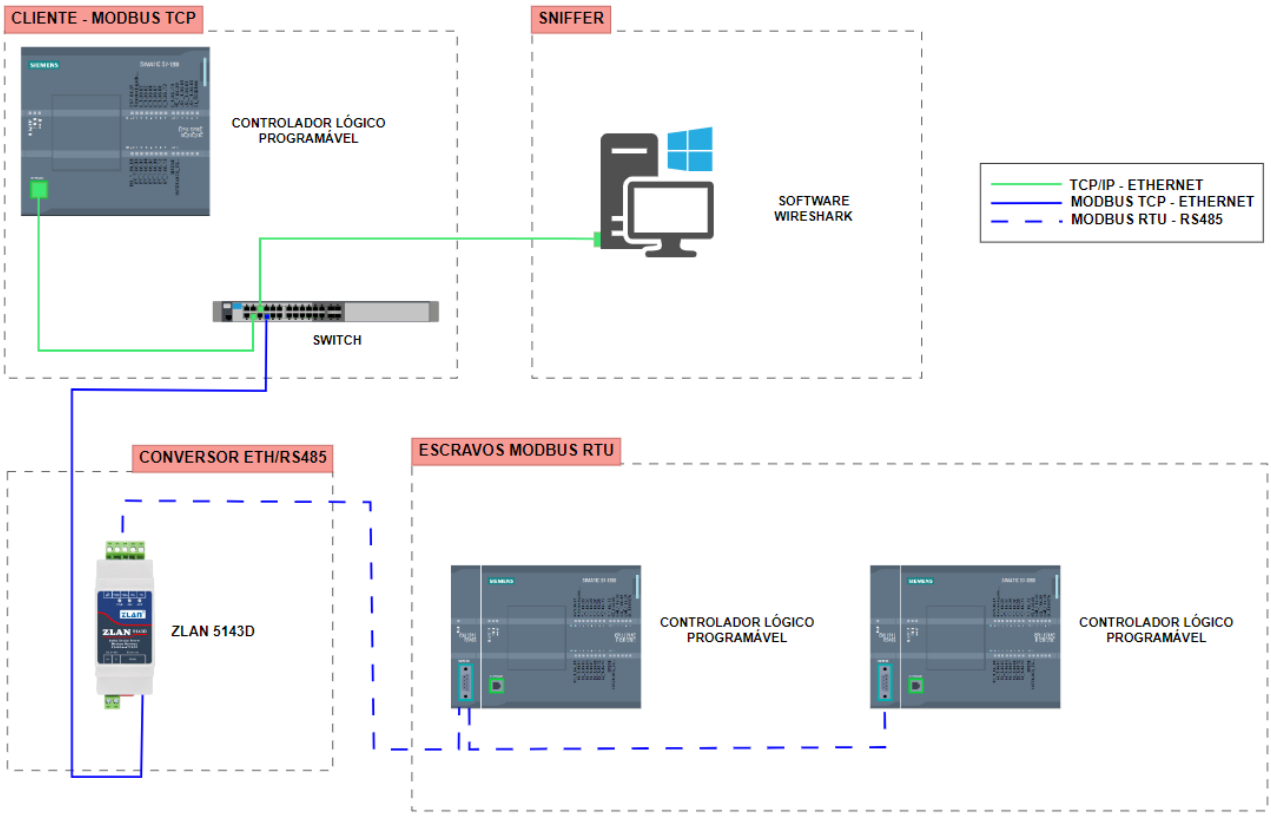
4. Resultados e Discussões

Após finalizar o projeto de automação para controle de demanda energética – e a sua simulação – foi possível coletar os resultados do sistema em funcionamento durante os testes. Neste tópico serão apresentados a análise de rede Modbus TCP e os testes de controle da demanda no sistema simulado.

4.1. Análise da rede Modbus TCP

Com o sistema desenvolvido e em funcionamento pleno, é possível fazer a captura de linha da rede, seja ethernet ou serial, neste caso, a captura é feita na rede Modbus TCP, portanto, na rede ethernet. Para a validação da comunicação Modbus foi montado o seguinte cenário de teste, ilustrado na Figura 70.

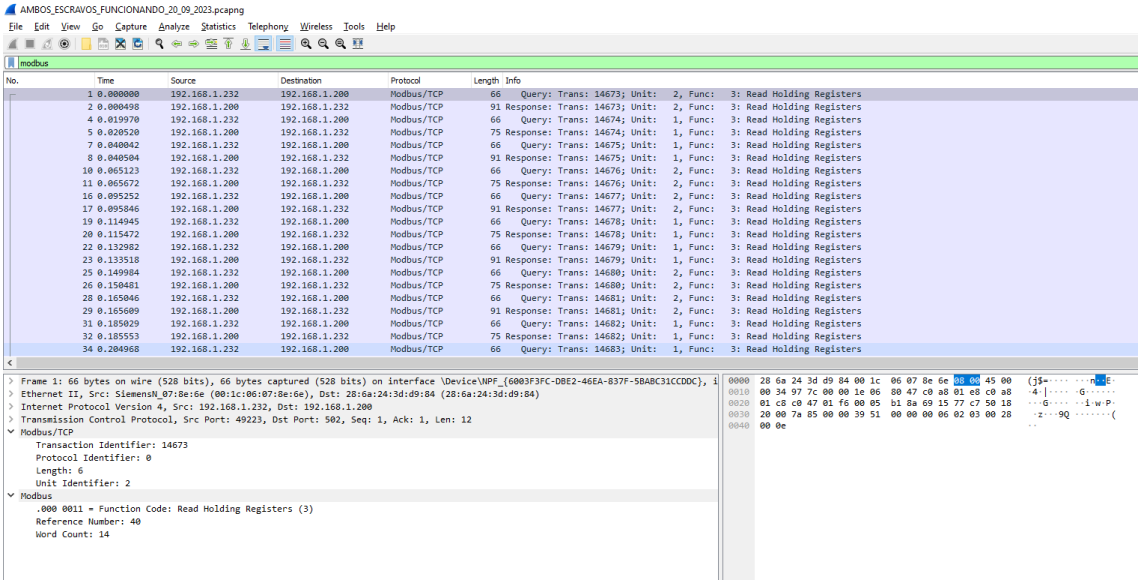
Figura 70 – Estrutura para captura de linha da rede Modbus TCP



Fonte: Próprio Autor (2023).

Para analisar a comunicação foi utilizado o software aberto *Wireshark*. Uma tela da captura é mostrada na Figura 71.

Figura 71 – Exemplo de captura no software *WireShark*



Fonte: Próprio Autor (2023).

Os dados foram salvos em csv e, desta forma, foi compilado uma Tabela com os dados obtidos, conforme mostrado na Tabela 15.

Tabela 15 – Parte dos dados capturados via *WireShark*

N	Time [s]	Source	Dest	Length	REQ/RES	Trans	Unit	Func	Frame	latência Medido [s]	latência Estimado [s]
1	0	IP_232	IP_200	66	REQ	43018	1	3	01 03 00 00 00 06		
2	0.000521	IP_200	IP_232	75	RES	43018	1	3	01 03 0c xx xx xx 12 bytes xx xx	0.521	0.01128
3	0.013927	IP_232	IP_200	66	REQ	43019	1	3	01 03 00 28 00 0E		
4	0.01447	IP_200	IP_232	91	RES	43019	1	3	01 03 1c xx xx xx 28 bytes xx xx	0.543	0.01256
5	0.0307	IP_232	IP_200	66	REQ	43020	2	3	02 03 00 00 00 06		
6	0.031215	IP_200	IP_232	75	RES	43020	2	3	02 03 0c xx xx xx 12 bytes xx xx	0.515	0.01128
7	0.048899	IP_232	IP_200	66	REQ	43021	2	3	02 03 00 28 00 0E		
8	0.049448	IP_200	IP_232	91	RES	43021	2	3	02 03 1c xx xx xx 28 bytes xx xx	0.549	0.01256
9	0.065328	IP_232	IP_200	66	REQ	43022	1	3	01 03 00 00 00 06		
10	0.065827	IP_200	IP_232	75	RES	43022	1	3	01 03 0c xx xx xx 12 bytes xx xx	0.499	0.01128

11	0.080313	IP_232	IP_200	66	REQ	43023	1	3	01 03 00 28 00 0E		
12	0.080824	IP_200	IP_232	91	RES	43023	1	3	01 03 1c xx xx xx 28 bytes xx xx	0.511	0.01256
13	0.094016	IP_232	IP_200	66	REQ	43024	2	3	02 03 00 00 00 06		
14	0.094416	IP_200	IP_232	75	RES	43024	2	3	02 03 0c xx xx xx 12 bytes xx xx	0.4	0.01128
15	0.110599	IP_232	IP_200	66	REQ	43025	2	3	02 03 00 28 00 0E		
16	0.111158	IP_200	IP_232	91	RES	43025	2	3	02 03 1c xx xx xx 28 bytes xx xx	0.559	0.01256
17	0.124862	IP_232	IP_200	66	REQ	43026	1	3	01 03 00 00 00 06		
18	0.125327	IP_200	IP_232	75	RES	43026	1	3	01 03 0c xx xx xx 12 bytes xx xx	0.465	0.01128
19	0.140389	IP_232	IP_200	66	REQ	43027	1	3	01 03 00 28 00 0E		
20	0.140907	IP_200	IP_232	91	RES	43027	1	3	01 03 1c xx xx xx 28 bytes xx xx	0.518	0.01256
21	0.15533	IP_232	IP_200	66	REQ	43028	2	3	02 03 00 00 00 06		
22	0.155813	IP_200	IP_232	75	RES	43028	2	3	02 03 0c xx xx xx 12 bytes xx xx	0.483	0.01128
23	0.169808	IP_232	IP_200	66	REQ	43029	2	3	02 03 00 28 00 0E		
24	0.170318	IP_200	IP_232	91	RES	43029	2	3	02 03 1c xx xx xx 28 bytes xx xx	0.51	0.01256
25	0.185345	IP_232	IP_200	66	REQ	43030	1	3	01 03 00 00 00 06		
26	0.18583	IP_200	IP_232	75	RES	43030	1	3	01 03 0c xx xx xx 12 bytes xx xx	0.485	0.01128
27	0.200341	IP_232	IP_200	66	REQ	43031	1	3	01 03 00 28 00 0E		
28	0.200882	IP_200	IP_232	91	RES	43031	1	3	01 03 1c xx xx xx 28 bytes xx xx	0.541	0.01256
29	0.214757	IP_232	IP_200	66	REQ	43032	2	3	02 03 00 00 00 06		
30	0.215272	IP_200	IP_232	75	RES	43032	2	3	02 03 0c xx xx xx 12 bytes xx xx	0.515	0.01128
31	0.230581	IP_232	IP_200	66	REQ	43033	2	3	02 03 00 28 00 0E		
32	0.231132	IP_200	IP_232	91	RES	43033	2	3	02 03 1c xx xx xx 28 bytes xx xx	0.551	0.01256
33	0.249857	IP_232	IP_200	66	REQ	43034	1	3	01 03 00 00 00 06		
34	0.250275	IP_200	IP_232	75	RES	43034	1	3	01 03 0c xx xx xx 12 bytes xx xx	0.418	0.01128
35	0.265397	IP_232	IP_200	66	REQ	43035	1	3	01 03 00 28 00 0E		
36	0.265845	IP_200	IP_232	91	RES	43035	1	3	01 03 1c xx xx xx 28 bytes xx xx	0.448	0.01256
37	0.280584	IP_232	IP_200	66	REQ	43036	2	3	02 03 00 00 00 06		

Fonte: Próprio Autor (2023).

De acordo com a Tabela 15, a coluna “time” mostra o tempo capturado pelo software. A coluna “source” e “dest” mostra o final dos endereços IPs da origem e destino da mensagem, respectivamente. Neste caso é possível notar somente a comunicação entre o

CLP S71200 (IP 192.168.1.232) e o gateway Modbus (IP 192.168.1.200). A coluna “len” representa o tamanho da mensagem. A coluna “REQ/RES“ indica se a mensagem é uma pergunta do mestre (REQ) ou uma resposta do escravo (RES). A coluna “Trans” mostra a transação da mensagem. E, por fim, as colunas “Unit”, “Func” e “Frame” identificam a mensagem modbus.

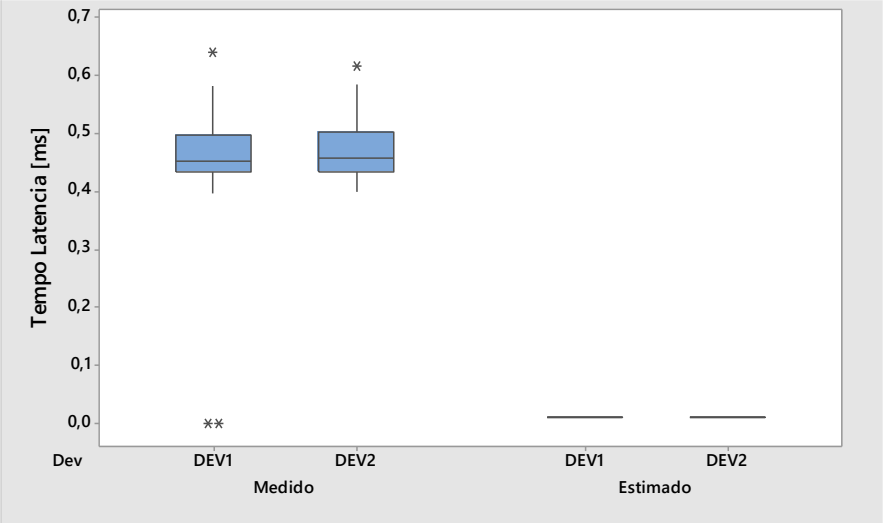
Baseado nestas informações foi criado a coluna “latência medida” que representa o tempo medido entre a pergunta do mestre e a resposta do escravo. Para validação do sistema de análise foi feita uma comparação dos valores medidos e estimados da rede. Como o modbus TCP utiliza a rede ethernet a 100Mbps, o tempo de byte considerado foi o seguinte:

$$T_{Byte} = T_{Bit} * 8 = \frac{1}{100000} * 8 = 8\ e^{-5} \qquad (7)$$

E desta forma, a latência da mensagem seria a quantidade de *bytes* multiplicada pelo T_{byte}. Neste caso, é possível notar que o mestre faz duas perguntas de leitura de *Holding Register* para cada um dos escravos e este ciclo se repete sempre da mesma forma, indicando que não tem problemas no algoritmo no tratamento das mensagens.

O gráfico boxsplot da Figura 72 mostra uma comparação entre os tempos estimado e medido. O valor estimado tem um valor médio de 0,011920 ms para os dois *devices*. Enquanto que os valores medidos tem média de 0,45888 ms com desvio padrão de 0,07053 para o *device* 1 e 0,46795 ms com desvio padrão de 0,04309 para o *device* 2. É possível notar que o valor medido tem um erro em relação ao estimado, mas isso é devido as características do sistema de captura (*switch*, computador, etc.) e dos tempos pequenos envolvidos, mas que podemos considerar os valores medidos como aproximados.

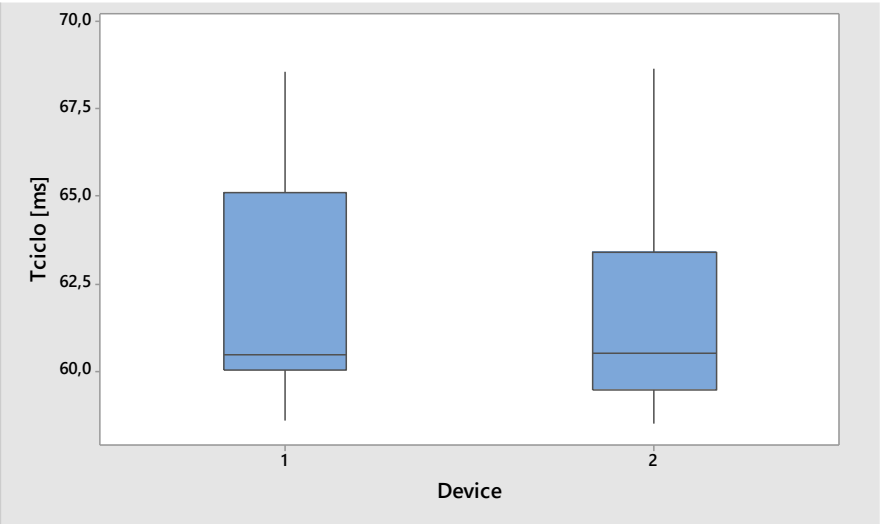
Figura 72 – Comparação entre tempo estimado e tempo medido



Fonte: Próprio Autor (2023).

A Figura 73 mostra a análise do ciclo de scan da rede quando todos os equipamentos estão funcionando. Foi determinado o tempo de ciclo para os dois *devices* em separado, ou seja, o tempo que o mestre demorava para repetir a pergunta para o *device* 1 e depois para o *device* 2.

Figura 73 – Análise do ciclo de scan da rede

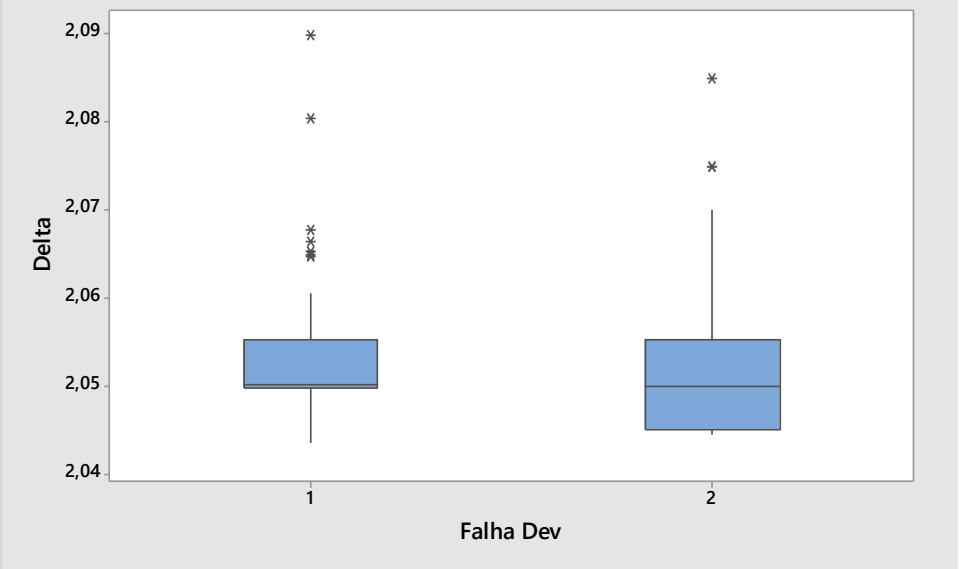


Fonte: Próprio Autor (2023).

O tempo de ciclo para o *device* 1 é de 61,95 ms com desvio padrão de 2,697, enquanto que para o *device* 2 foi de 61,66 ms com desvio padrão de 2.786. Os dois valores mostraram valores próximos onde o tempo em media foi de 30.90 ms por comando.

Considerando agora a analise de falha do sistema, é importante saber o que acontece com os tempos quando se perde um *device* na rede. O gráfico da Figura 74 mostra o tempo entre *requests* quando um *device* falha. No modbus este tempo é determinado pelo parâmetro *timeout*. Neste caso, o *timeout* do mestre está configurado em 2 segundos (*device* 1 = 2,0536 com desvio padrão de 0,00821 e para o *device* 2 2,0523 com desvio padrão de 0,00813). Desta forma, para cada *request*, o mestre espera ate 2 segundos pela resposta do escravo. E no final este valor é comum para os dois *devices*.

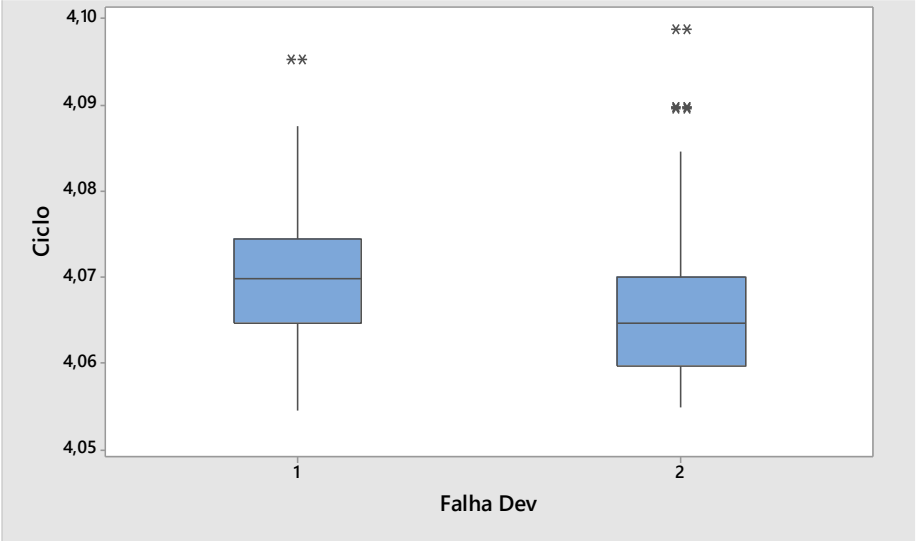
Figura 74 – Tempo do ciclo de *scan* quando o *device* 1 falha



Fonte: Próprio Autor (2023).

O gráfico da Figura 75 mostra o tempo de ciclo do *device* 1 quando o *device* 2 está com problemas de comunicação.

Figura 75 – Tempo do ciclo de *scan* quando o *device* 2 falha



Fonte: Próprio Autor (2023).

Os tempos obtidos de ciclo de *scan* foram de 4,0698 s com desvio padrão de 0,00882 para o *device* 1, e 4,0680 s com desvio padrão de 0,0108 para o *device* 2. Ou seja, o valor é coerente pois como existem duas mensagens para cada *device*, ele espera dois *timeouts* para cada *device* para então fazer o próximo *scan*.

Conclui-se que o sistema tem um bom funcionamento, com um tempo de ciclo de *scan* em torno de 60 ms. Com a falha de algum dos equipamentos, o sistema se comporta de acordo com o esperado, tendo em média 4 s e 60 ms por ciclo, uma vez que o *timeout* de comunicação é de 2 s. Outro ponto positivo é que ao retornar um escravo que estava fora da rede, o sistema reconhece automaticamente e volta para o seu funcionamento normal, sem maiores complicações.

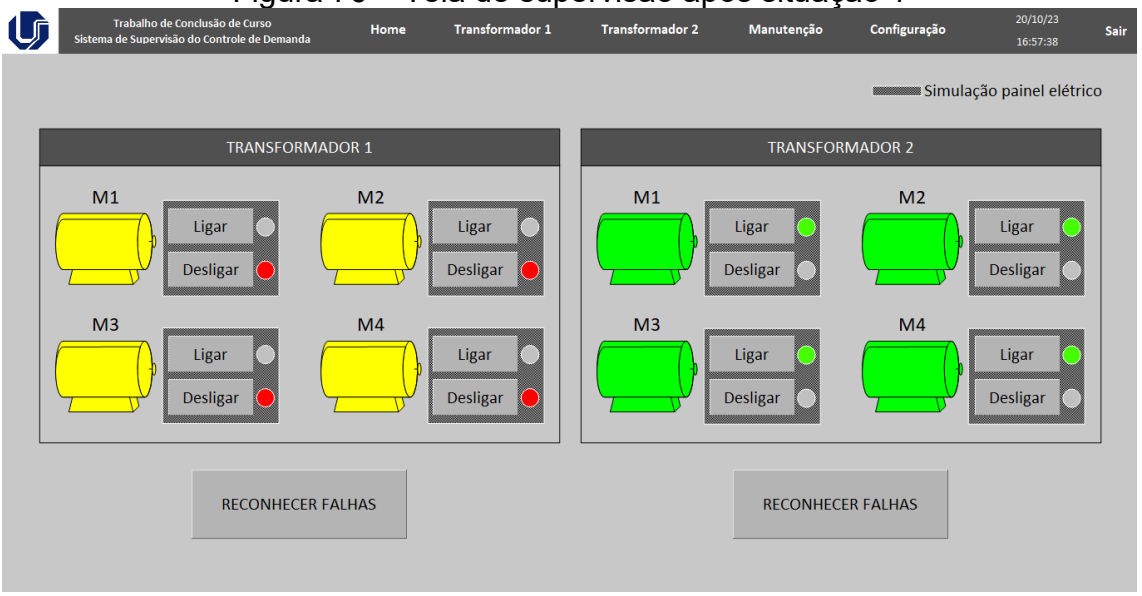
4.2. Análise do sistema de controle de demanda

Para validação do sistema de controle de demanda, foram executadas todas as possibilidades de desligamento de cargas do projeto original, mas no sistema simulado, totalizando 6 situações.

A primeira situação é quando o sistema está com a demanda total dentro dos limites estabelecidos, mas o escravo 1 está com a demanda local excedida. Sendo assim, a automação deve desligar sequencialmente as cargas presentes no escravo 1, independentemente da prioridade de desligamento escolhida pelo operador. Para esta situação, Figura 76, a demanda será mantida alta para que todas as cargas do escravo 1

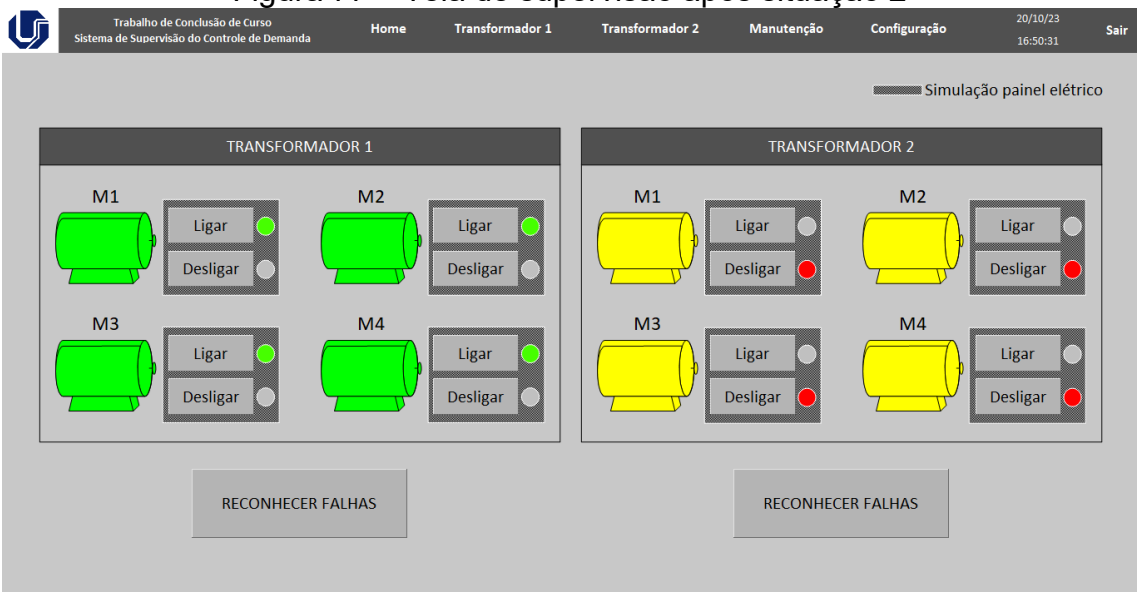
sejam desligadas. Na situação dois, Figura 77, o sistema repete a situação um, mas com o escravo 2 extrapolando a demanda local, mantendo a demanda total dentro dos limites.

Figura 76 – Tela de supervisão após situação 1



Fonte: Próprio Autor (2023).

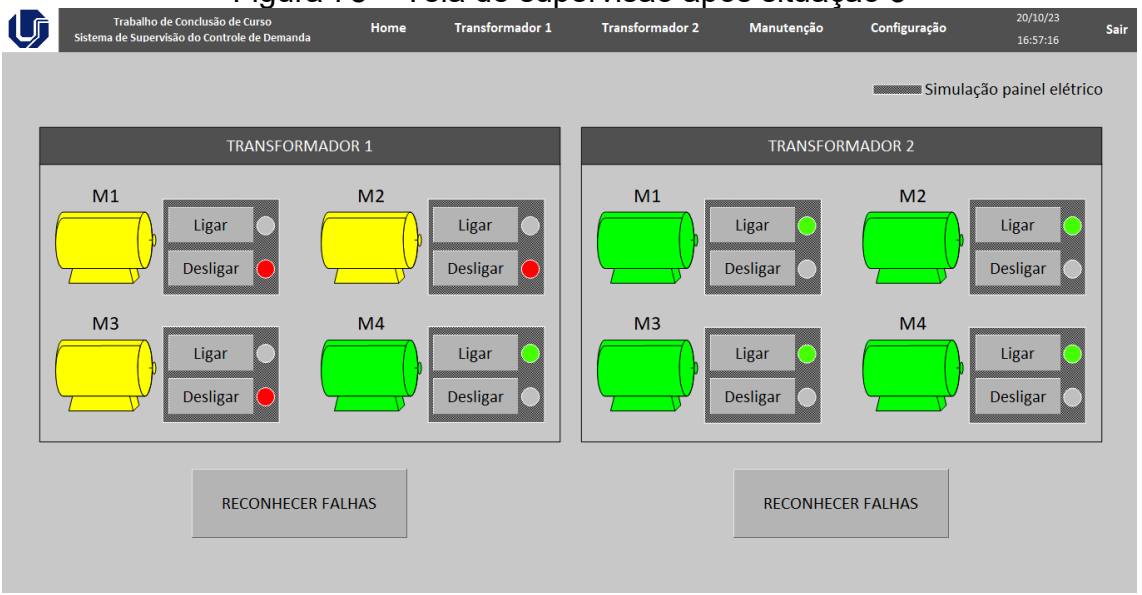
Figura 77 – Tela de supervisão após situação 2



Fonte: Próprio Autor (2023).

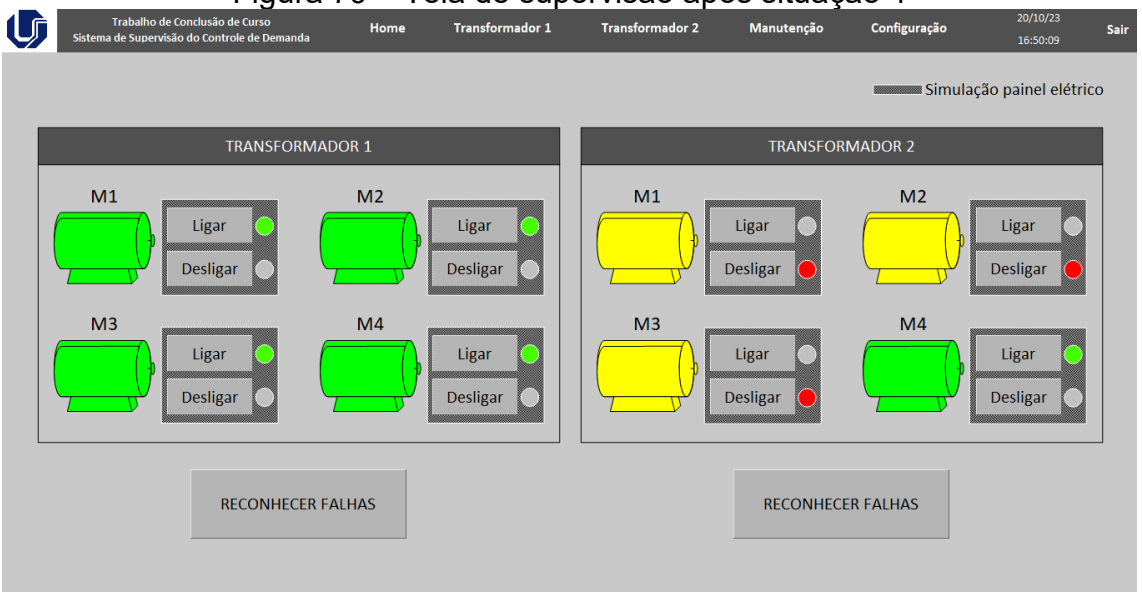
Para a terceira situação, Figura 78, testa-se o desligamento parcial das cargas do escravo 1, seja para demanda local ou demanda total excedidas, fazendo com que a demanda seja reestabelecida dentro dos limites antes que todas as cargas deste escravo sejam desligadas. Sendo assim, tem-se de uma a três cargas desligadas no escravo 1. Já na quarta situação, Figura 79, esse cenário se repete, mas para o escravo 2.

Figura 78 – Tela de supervisão após situação 3



Fonte: Próprio Autor (2023).

Figura 79 – Tela de supervisão após situação 4



Fonte: Próprio Autor (2023).

As últimas duas situações contam com a demanda total acima do limite e as demandas locais dentro dos limites estabelecidos, onde a prioridade escolhida pelo operador estabelecerá qual escravo terá suas cargas sequencialmente desligadas, sendo a situação 5 para prioridade de desligamento o escravo 1 e a situação 6, prioridade de desligamento o escravo 2. O resultado dessas simulações seguem as figuras 76 e 77 e um exemplo da tela de configuração durante a simulação está ilustrada na Figura 80.

Figura 80 – Exemplo de tela de configuração utilizada durante a simulação

Fonte: Próprio Autor (2023).

Posto isso, todas as possibilidades de demanda alta, seja a demanda total ou a demanda local do escravo, foram testadas e, após os testes, foi possível observar que todas estão com pleno funcionamento e sempre atuando da maneira esperada, desligando as cargas sequencialmente sempre que ocorre alguma condição de demanda excedida.

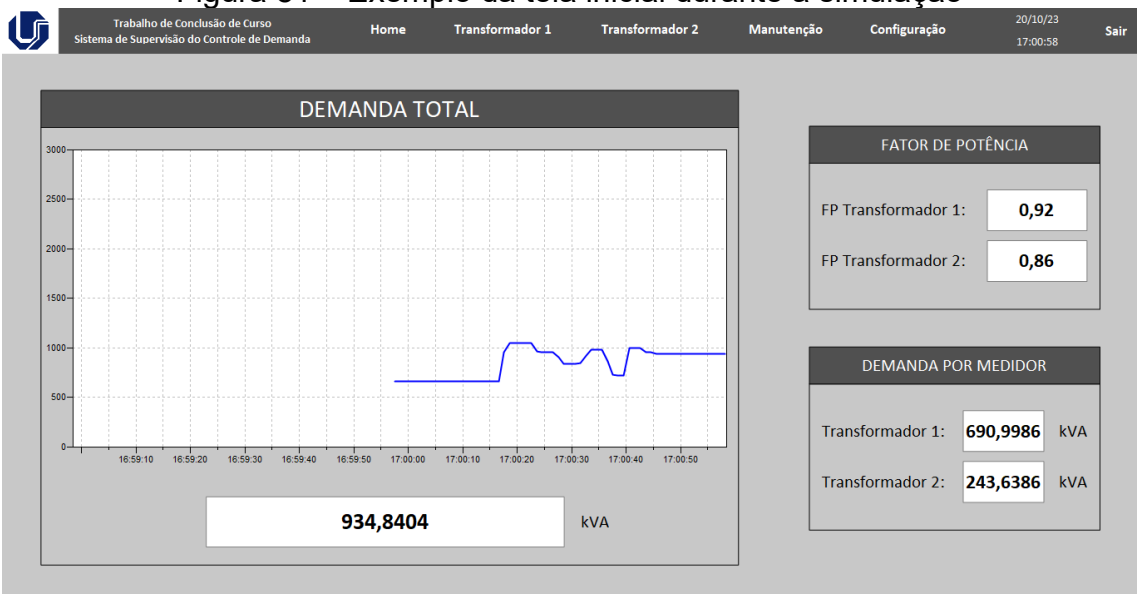
Portanto, a operação terá uma previsibilidade certeira durante o processo – uma vez que o operador poderá supervisionar a demanda e estabelecer ordens de prioridade de desligamento – e a companhia não sofrerá com multas fora de seu orçamento. Todas as possibilidades e resultados dos testes estão evidenciados na Tabela 16. Por fim, tem-se exemplos das telas inicial e do transformador 1 durante a simulação, ilustradas nas Figura 81 e 82, respectivamente.

Tabela 16 – Testes executados no sistema simulado de controle de demanda

Situação	Teste	Prioridade	Resultado
1	Desligamento total das cargas do escravo 1 com demanda local alta	N/A	Desligado Corretamente
2	Desligamento total das cargas do escravo 2 com demanda local alta	N/A	Desligado Corretamente
3	Desligamento parcial das cargas do escravo 1 em qualquer condição de demanda alta	N/A	Desligado Corretamente
4	Desligamento parcial das cargas do escravo 2 em qualquer condição de demanda alta	N/A	Desligado Corretamente
5	Desligamento total das cargas do escravo 1 com demanda total alta	Prioridade 1	Desligado Corretamente
6	Desligamento total das cargas do escravo 2 com demanda total alta	Prioridade 2	Desligado Corretamente

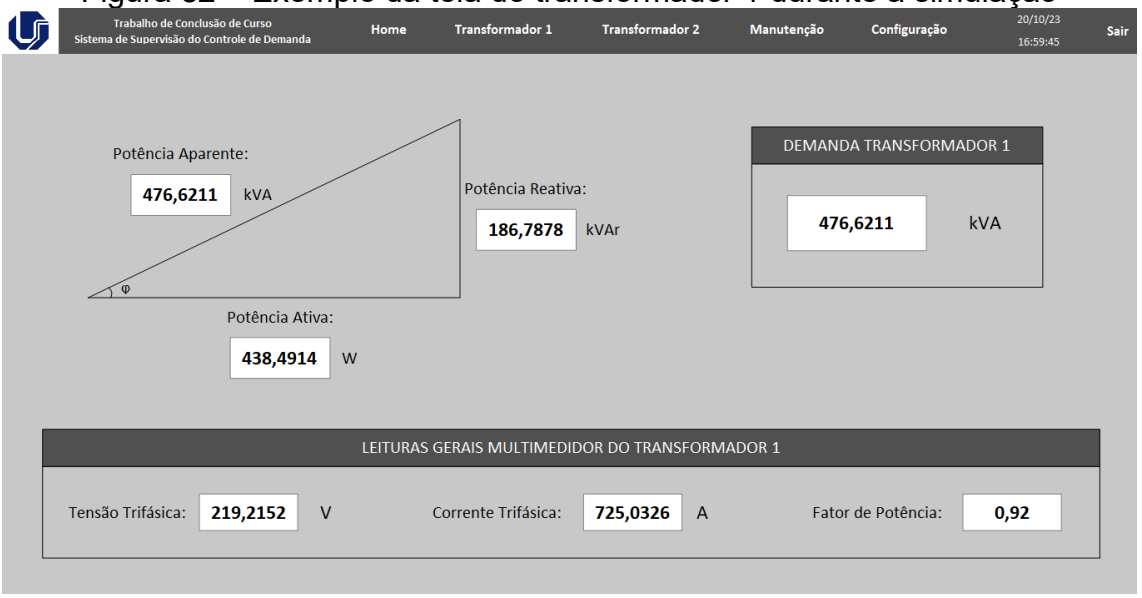
Fonte: Próprio Autor (2023).

Figura 81 – Exemplo da tela inicial durante a simulação



Fonte: Próprio Autor (2023).

Figura 82 – Exemplo da tela do transformador 1 durante a simulação



Fonte: Próprio Autor (2023).

5. Conclusão

O desenvolvimento deste trabalho contou com um sistema completo de automação para que uma planta industrial não extrapole o consumo de energia estabelecido pela concessionária de energia. O projeto primordial foi desenvolvido para um caso real há 12 meses – durante minha atuação como analista de automação – visando redução de gastos da empresa. Incluiu a elaboração de um sistema de supervisão via IHM, da lógica de controle através de um controlador industrial e de uma rede de comunicação completa pelo protocolo Modbus.

Baseado nisso, o sistema para validação dos resultados foi totalmente refeito no laboratório durante a execução deste trabalho, desde a lógica de controle para o cliente Modbus TCP, o sistema SCADA desenvolvido no software Elipse E3 e a lógica para os escravos simulados, buscando manter a originalidade do projeto base. Esta etapa permitiu a coleta de resultados e análises aprofundadas do sistema, como a análise da rede Modbus TCP.

Foi possível identificar que a lógica de controle desenvolvida foi eficiente naquilo que foi proposto controlar, além de uma eficaz máquina de estados para comunicação Modbus entre um cliente em TCP para com mais de um escravo em RTU, o que permitiu a economia ao utilizar apenas um gateway-conversor. Ademais, as telas de supervisão projetadas são simples e objetivas, garantindo fácil entendimento para aqueles que operam o sistema.

Como aprimoramento futuro, é importante a aplicação de capital em inversores de frequência para controle dos motores via rede de comunicação industrial, o que traz mais confiabilidade e segurança para o sistema.

Por fim, os objetivos propostos neste trabalho foram, satisfatoriamente, concluídos e foi possível aprofundar nos conceitos estudados no curso de engenharia de controle e automação, como redes de comunicação industrial, programação de CLP e desenvolvimento de sistemas supervisórios, os quais são assuntos indispensáveis para um profissional da área.

6. Referências Bibliográficas

BARROS, M. R. d. A. **ESTUDO DA AUTOMAÇÃO DE CÉLULAS DE MANUFATURA PARA MONTAGENS E SOLDAGEM INDUSTRIAL DE CARROCERIAS AUTOMOTIVAS**. 2006.

BOLESTAD, R. L. **Análise de Circuitos.** 2011.

BOLTON, W. ***Programmable Logic Controllers***. 2006.

BOLTON, W. ***Programmable logic controllers*.** [S.l.]: Newnes, 2015.

CASTELLANO, G. **Desenvolvimento de um Controlador de Demanda de Energia Elétrica de Baixo Custo para a Greylogix Brasil.** 2015

Detailed description of the Modbus TCP protocol with command examples. IPC2U. 2017. Disponível em: https://ipc2u.com/articles/knowledge-base/detailed-description-of-the-modbus-tcp-protocol-with-command-examples/.

ELIAS, M. C. [2002]. **Secagem e armazenamento de grãos de milho e de sorgo na propriedade rural.** Disponível em: https://www.agricultura.rs.gov.br/upload/arquivos/201811/23093823-armazenamento-em-prop-milho-e-sorgo.pdf

Elipse. **Manual do Usuário do Driver Siemens MProt.** 2022.

Elipse. **Elipse E3 Software.** 2015. Disponível em: https://www.elipse.com.br/produto/elipse-e3.

ESFERABLOG. **Demanda Contratada: O que é e a importância de uma boa estratégia.** 2022. Disponível em: https://blog.esferaenergia.com.br/gestao-empresarial/demanda-contratada.

FREITAS, C.M; **Redes de Comunicação em RS-485**. Embarcados, 2017. Disponível

em: https://embarcados.com.br/redes-de-comunicacao-em-rs-485/.

FRENZEL, L. **What's The Difference Between The RS-232 And RS-485 Serial Interfaces?** Electronic Design, 2013. Disponível em: https://www.electronicdesign.com/.

FILHO, J. M. **Instalações Elétricas Industriais.** 2010.

FOROUZAN, B.A. **Comunicação de Dados e Redes de Computadores**. 2008.

FRANCHI, C. M.; CAMARGO, V.L.A. **Controladores Lógicos Programáveis Sistemas Discretos.** 2008.

SADIKU, M. N. O.; ALEXANDER, C K. **Fundamentos de Circuitos Elétricos.** 2003.

OZUR, F. S.; PEREIRA, T. H.; CORREA, J. D. A. d. S. **Controle de Demanda de Energia Elétrica.** 2011.

SENAI. **Linguagens de programação para controladores lógicos programáveis.** 2015.

ÛRZEDA, C. C. d. **SOFTWARE SCADA COMO PLATAFORMA PARA A RACIONALIZAÇÃO INTELIGENTE DE ENERGIA ELÉTRICA EM AUTOMAÇÃO PREDIAL.** 2006.

WORLD, L. L. **PLC Manufacturers: The Latest PLC Brands, Rankings & Revenues.** 2020.

PUPO, M. S. **Interface homem-máquina para supervisão de um CLP em controle de processos através da WWW.** Tese (Doutorado) — Universidade de São Paulo, 2002.

MATTEDE, H. **Fórmulas de potência, quais são?** Mundo da Elétrica, [2016]. Disponível em: https://www.mundodaeletrica.com.br/formulas-de-potencia-quais-sao/

THOMAS, G. **Introduction to the Modbus Protocol**. 2008.

SOUZA, E.L.S. **Redes de Comunicação Industrial.** 2010.

O que é potência Ativa, Reativa e Aparente? AlugaGera, 2019. Disponível em: https://alugagera.com.br/noticias/potencia-ativa-reativa-aparente

MACKAY, S.; WRIGHT, E.; REYNDERS, D.; PARK, J. **Practical Industrial Data Networks: Design, Installation and Troubleshooting.** 2004.

MACIEL, G. **Triângulo das Potências – Potência aparente, ativa e reativa.** Ligando os fios, 2021. Disponível em: https://ligandoosfios.com.br/triangulo-das-potencias-potencia-aparente-ativa-e-reativa/

MONSETTATE, R.E.Z; DE LA TORRE, C.P.C. **Diseño e Implementación de una Red Modbus/RTU entre dos Autómatas Programables S7-1200 Basado en el Estándar RS485.** 2018.

Linguagens de programação PLC. Learnchannel-TV, [2017]. Disponível em: https://learnchannel-tv.com/pt/clp/linguagens-de-programacao-clp/.

Introduction to Modbus TCP/IP. Acromag, [2005].

Modbus. **Modbus Messaging on TCP/IP Implementation Guide.** 2006. Disponível em: www.modbus.org/.

MORAES, C. C. d.; CASTRUCCI, P. d. L. 2006. **Engenharia de Automação Industrial.**

Modbus. **Modicon Modbus Protocol Reference Guide.** 1996. Disponível em: www.modbus.org/.

WEG. **Manual do Usuário do Analisador de Energia MMW03-M22CH.** 2019.

Ministério de Agricultura e Pecuária. **Produção de grãos brasileira deverá chegar a 390 milhões de toneladas nos próximos dez anos**. 2023. Dísponível em: https://www.gov.br/agricultura/pt-br/assuntos/noticias/producao-de-graos-brasileira-devera-chegar-a-390-milhoes-de-toneladas-nos-proximos-dez-anos

SOUZA, R. M. d.; FIGUEIREDO, R. S.; OLIVEIRA NETO, O. J. d. 2010. **Modelo para gerenciamento de custos relacionados à secagem e armazenagem de grãos para aplicação da metodologia *system dynamics*.**

OLIVEIRA, C. O. d. 2021. **Qualidade de grãos: saiba como garantir para a sua lavoura.** Disponível em: https://www.myfarm.com.br/qualidade-de-graos/

SULZBACHER, A. L.; VILELA, F. A.; ARAÚJO, A. S. **Monitoramento da secagem de milho em secador intermitente de coluna de cavalete.** [2013].

Secagem. Comil, [2022]. Disponível em: https://comil.com.br/pt/produtos/secadores/

MATTEDE, H. Motor trifásico. **Como funciona e qual a sua aplicação?** Munda da Elétrica, [2016]. Disponível em: https://www.mundodaeletrica.com.br/motor-trifasico-como-funciona-e-qual-sua-aplicacao/

UMANS, S. D. **Máquinas Elétricas de Fitzgerald e Kingsley.** 2014.

Secagem de grãos: Entenda como funciona esse processo. Fimaco, 2018. Disponível em: https://fimaco.com.br/secagem-de-graos-entenda-o-processo/

SANTOS, R. F. d. **Secagem e armazenamento de grãos: diferentes tipos e custos.** 2019. Disponível em: https://blog.aegro.com.br/secagem-e-armazenamento-de-graos/