



UNIVERSIDADE FEDERAL DO RIO GRANDE - FURG
CENTRO DE CIÊNCIAS COMPUTACIONAIS - C3
GRADUAÇÃO EM ENGENHARIA DE AUTOMAÇÃO
SISTEMAS DE AUTOMAÇÃO II



WONKA - FÁBRICA DE CHOCOLATES

ANDRESSA CAVALCANTE DA SILVA, 140594
MARINA ZANOTTA ROCHA, 140592

RIO GRANDE
2023



UNIVERSIDADE FEDERAL DO RIO GRANDE - FURG
CENTRO DE CIÊNCIAS COMPUTACIONAIS - C3
GRADUAÇÃO EM ENGENHARIA DE AUTOMAÇÃO
SISTEMAS DE AUTOMAÇÃO II



WONKA - FÁBRICA DE CHOCOLATES

Apresentado como parte da avaliação da disciplina de Sistemas de Automação II no Curso de Engenharia de Automação, na Universidade Federal do Rio Grande - FURG .

Prof: Msc. Gabriel Lavoura dos Santos

ANDRESSA CAVALCANTE DA SILVA, 140594
MARINA ZANOTTA ROCHA, 140592

RIO GRANDE
2023

LISTA DE FIGURAS

Figura 1 – Fluxograma do processo de fabricação do chocolate	2
Figura 2 – Primeira Máquina de Estados Finitos	3
Figura 3 – Segunda Máquina de Estados Finitos	4
Figura 4 – Cena criada no <i>Factory I/O</i>	5
Figura 5 – Atuador <i>pusher</i> para separação dos diferentes moldes de chocolate . . .	5
Figura 6 – Sensor para detectar os diferentes moldes de chocolate	6
Figura 7 – Sensor óptico	6
Figura 8 – Painel utilizado	6
Figura 9 – Representação de dois moldes diferentes de chocolate	7
Figura 10 – Lista das variáveis globais - Codesys	8
Figura 11 – Código em Ladder - Codesys	8
Figura 12 – Comunicação OPC	9
Figura 13 – Erro drive GUtil.dll OPC	10
Figura 14 – Erro drive GClient.dll OPC	10
Figura 15 – Adicionando dispositivo Ethernet no Codesys	11
Figura 16 – Adicionando dispositivo Modbus - slave no Codesys	12
Figura 17 – Telas de configuração Modbus do Codesys	12
Figura 18 – Endereçamento das variáveis no Codesys	13
Figura 19 – Protocolo Modbus no Factory I/O	13
Figura 20 – Tela Menu	15
Figura 21 – Tela <i>Control</i>	15
Figura 22 – Tela <i>Viewer</i>	16
Figura 23 – Motores Acionados, Tela <i>Viewer</i>	16
Figura 24 – Configuração botões para troca de telas	17
Figura 25 – Telas de configuração do driver Modbus do ElipseE3	18
Figura 26 – Lista de <i>tags</i> do ElipseE3	18

LISTA DE ABREVIATURAS E SIGLAS

CLP	Controlador Lógico Programável
OPC	Open Platforms Communications
NA	Normalmente Aberto
NF	Normalmente Fechado
SCADA	Sistema de Supervisão e Aquisição de Dados

SUMÁRIO

1 – INTRODUÇÃO	1
2 – METODOLOGIA	2
2.1 Processo de Fabricação do Chocolate	2
2.1.1 Máquina de Estados Finitos	3
2.2 Criação cena Factory I/O	4
2.3 Programação em Ladder - Codesys	7
2.3.1 Desafios encontrados	7
2.4 Implementação da Comunicação OPC	9
2.4.1 Desafios Encontrados	9
2.5 Implementação da Comunicação Modbus	10
2.5.1 Desafios Encontrados	11
2.6 Sistema Supervisório - Elipse	14
2.6.1 Dinâmica implementada nas trocas de telas	14
2.6.2 Configuração das tags	14
2.6.3 Desafios Encontrados	17
3 – Resultados	19
4 – CONCLUSÃO	20
4.1 Trabalhos Futuros	20
Referências	21

1 INTRODUÇÃO

O presente artigo consiste em descrever o processo de desenvolvimento do trabalho proposto como método avaliativo da disciplina de Sistemas de Automação II do curso de Engenharia de Automação. O projeto consiste em fazer um sistema de automação de uma planta industrial no *Factory I/O*, a qual a planta foi inteiramente construída do zero, desenvolvemos a estrutura em *ladder* no *software Codesys* para controlar o CLP, e o desenvolvimento de um Sistema de Supervisão e Aquisição de Dados *SCADA* no *software ElipseE3*.

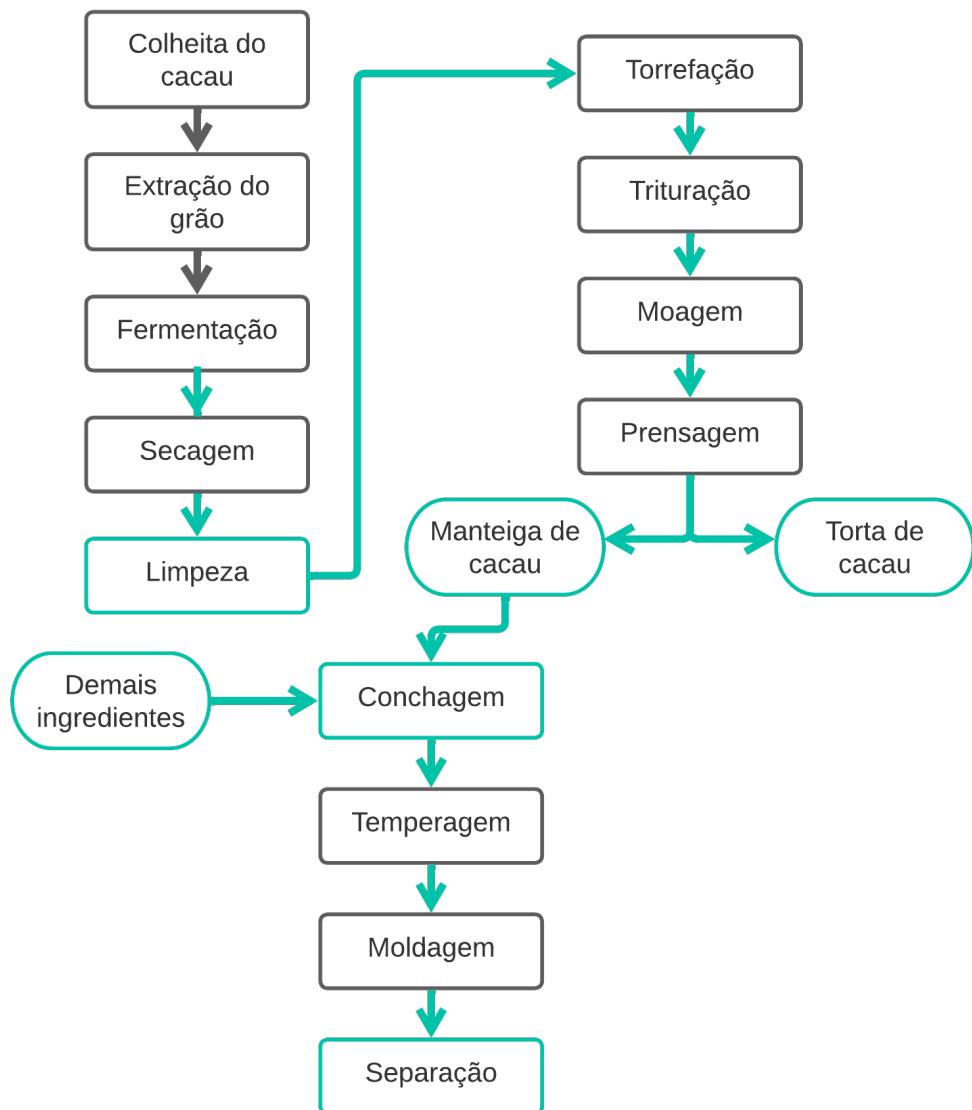
Propô-se então, simular parte de um processo dentro de uma industria que fabrica chocolate. Ao descorrer deste trabalho, detalham-se os diversos problemas superados, assim como as configurações que fizeram-se necessárias para realizar a simulação de uma comunicação entre os três *softwares* utilizados. Inicialmente estudou-se sobre as etapas da fabricação de chocolate a partir da colheita dos frutos, dos quais são retiradas as polpas onde se encontram as sementes, parte que realmente dá origem ao produto desejado. As sementes são levadas para fermentação e depois para secagem, o que permite então que os grãos sejam armazenados (CHOCOLAT, 2023). Então, encaminha-se para moagem dos grãos, seguida da prensagem da qual são obtidas a torta e a manteiga de cacau utilizadas, respectivamente, para produção de chocolate em pó e das barras. Após, tem-se a maxalação em que ocorre a mistura dos demais ingredientes, e por fim chega-se na temperagem, responsável pela aparência e qualidade do produto final (OLIVEIRA, 2023). Com isso, o produto pode ser moldado e separado para as embalagens, parte que a dupla selecionou para simular.

2 METODOLOGIA

2.1 Processo de Fabricação do Chocolate

Como dito anteriormente, processo escolhido foi o de uma fábrica de produção de chocolate, desde a colheita do fruto até a separação de produtos para as embalagens. Inicialmente organizou-se um fluxograma com os processos de uma indústria completa dessa categoria, apresentado na figura 1.

Figura 1 – Fluxograma do processo de fabricação do chocolate



Fonte: Elaborada pelas Autoras.

Destaca-se o fato de que devido a diversas limitações de *software* e tempo disponível,

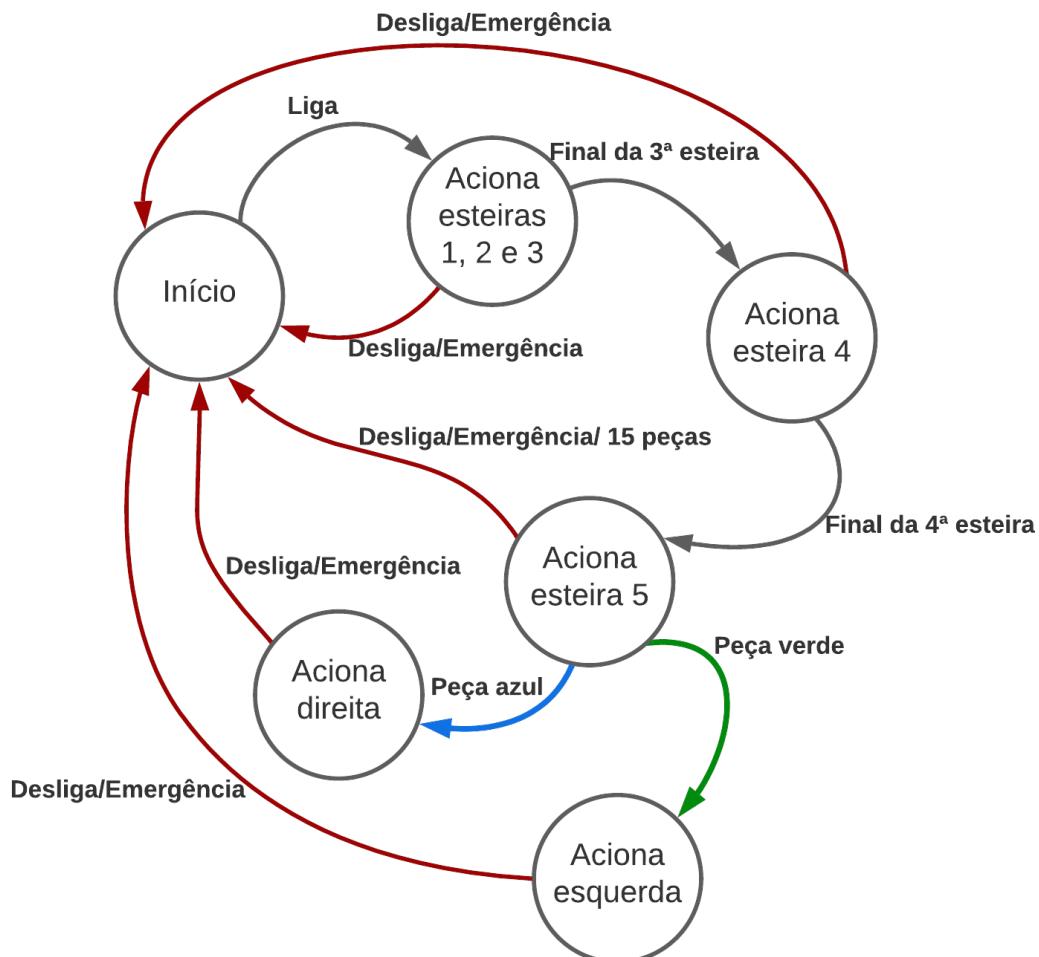
optou-se por simular apenas a última etapa de todo o processo, a separação dos diferentes moldes do chocolate. Portanto, para desenvolvermos melhor a proposta, criamos uma máquina de estados.

2.1.1 Máquina de Estados Finitos

Para a criação da máquina de estados, planejamos de acordo com alguns ítems selecionados préviamente, na subsessão 2.2, devido às limitações do ambiente de simulação utilizado.

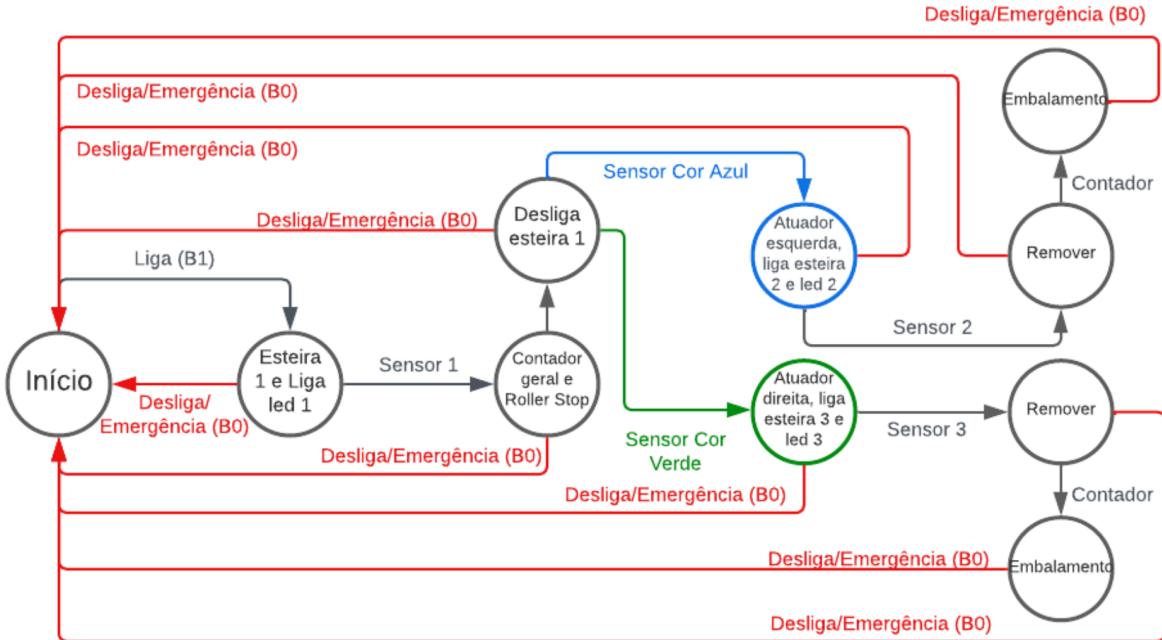
O objetivo foi simular algo simples, porém que estivesse o mais próximo possível do processo de uma empresa real. Por isso, priorizou-se o uso do botão de emergência e *stop*, como elementos de segurança presentes em indústrias. O processo foi esquematizado diversas vezes, o que resultou nas figuras 2 e 3, em que ambas descrevem a proposta idealizada inicialmente.

Figura 2 – Primeira Máquina de Estados Finitos



Fonte: Elaborada pelas Autoras.

Figura 3 – Segunda Máquina de Estados Finitos



Fonte: Elaborada pelas Autoras.

Ressalta-se o fato de que ao decorrer do desenvolvimento do trabalho, esses diagramas sofreram algumas alterações a fim de satisfazer os novos objetivos conforme as necessidades surgiam, por exemplo a decisão de abstrair alguns estados devido a falta de tempo, uma vez que se encontrou muita dificuldade em fazer a comunicação.

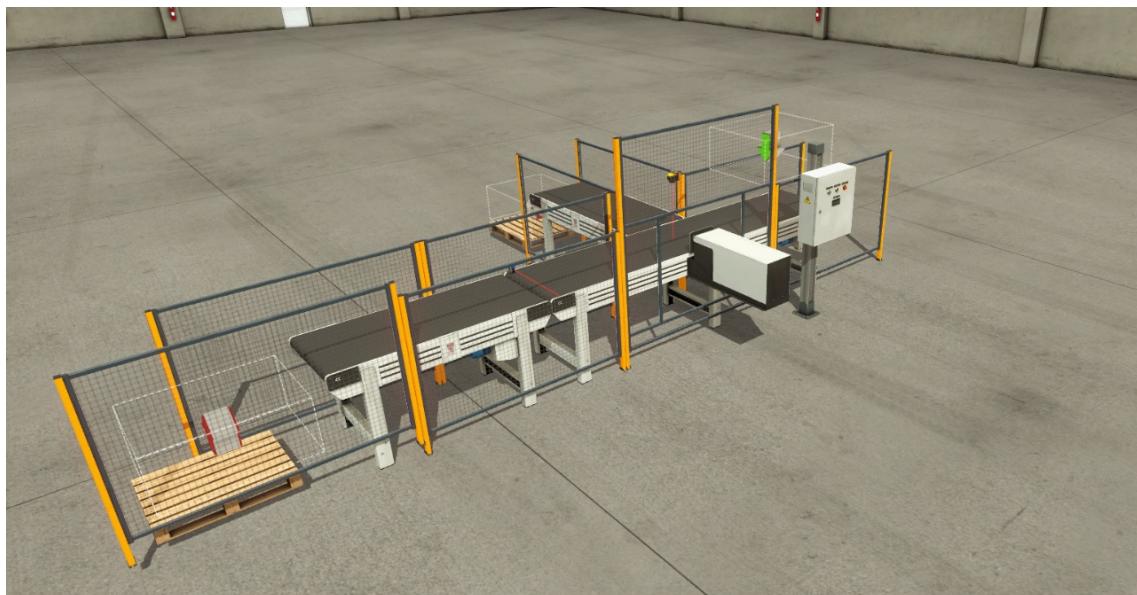
2.2 Criação cena Factory I/O

Para simular a fábrica de chocolate de modo a conter todos os estados propostos na subseção 2.1.1, adaptou-se alguns itens disponibilizados pelo *software* (I/O, 2023a), que serão detalhados posteriormente. Em geral pode-se observar na figura 4, a planta completa, a qual é composta pelos seguintes elementos:

- Atuadores
 - 1 esteira de 4m
 - 2 esteiras de 2m
 - 1 *pusher* (figura 5): empurra as peças azuis
- Sensores
 - Sensor de visão (figura 6): configurado como *boolean*, muda para estado lógico 1 ao detectar peças azuis
 - Sensor óptico (figura 7): também *boolean*, seu estado altera ao detectar qualquer peça, nesse caso apenas as verdes chegam até ele

- Acionamento
 - Botão Liga: aciona a esteira 1 de 4m
 - Botão Stop: para todas as esteiras, encerra o processo
 - Botão de Emergência: diferente do botão Stop, depois de pressionado é necessário destravá-lo para que se possa ligar o sistema novamente
- Demais elementos
 - Painel de controle (figura 8): quadro onde estão os botões e o *display*
 - *Display*: recebe e mostra o valor do contador
 - Grades de proteção: assim como tem-se o botão de Emergência para simular a segurança de uma indústria real, incluiu-se esses elementos
 - Peças (figura 9): representam os moldes dos chocolates

Figura 4 – Cena criada no *Factory I/O*



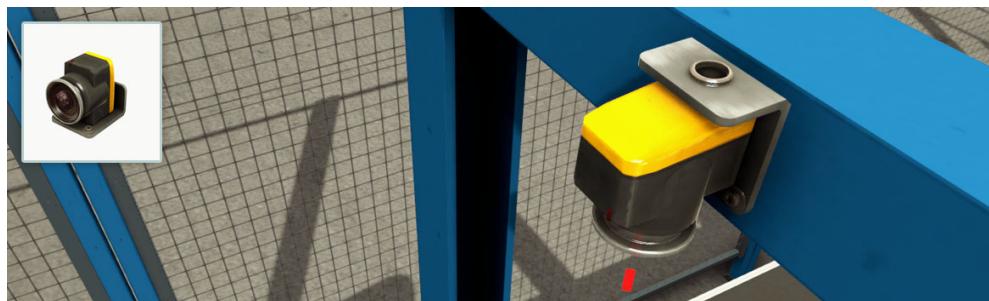
Fonte: Elaborada pelas Autoras.

Figura 5 – Atuador *pusher* para separação dos diferentes moldes de chocolate



Fonte: Factory IO, 2023.

Figura 6 – Sensor para detectar os diferentes moldes de chocolate



Fonte: Factory IO, 2023.

Figura 7 – Sensor óptico



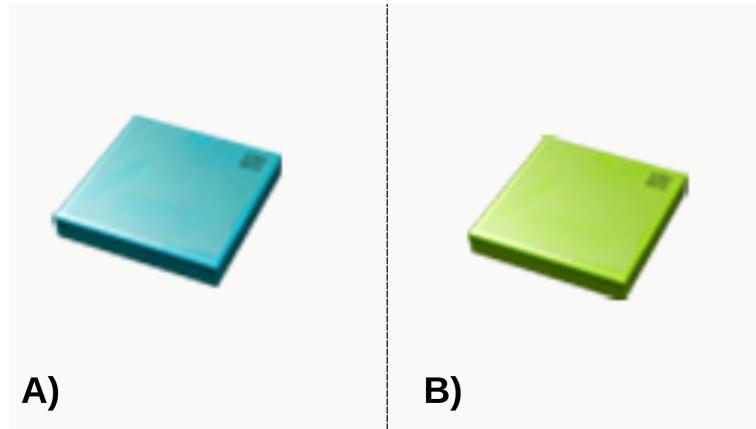
Fonte: Factory IO, 2023.

Figura 8 – Painel utilizado



Fonte: Elaborada pelas Autoras.

Figura 9 – Representação de dois moldes diferentes de chocolate



Fonte: Factory I/O, 2023.

2.3 Programação em Ladder - Codesys

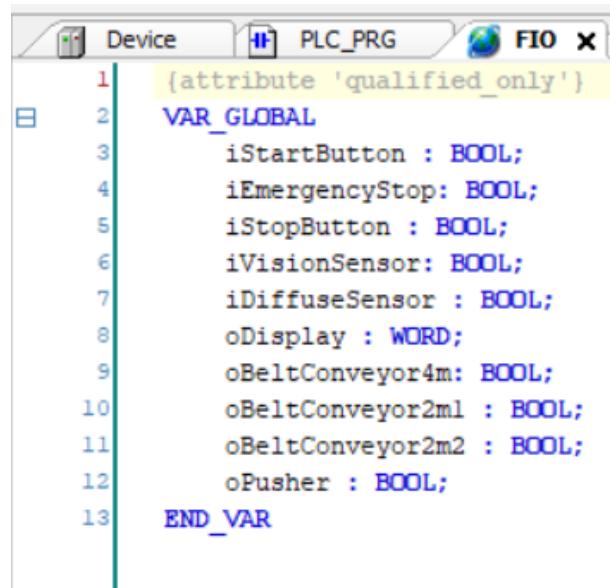
Definido o diagrama de estados a ser implementado e com a seleção dos elementos a serem utilizados na cena *Factory I/O* elaborou-se o código de controle das variáveis na linguagem Ladder no *software* Codesys. O primeiro passo foi criar uma lista de variáveis globais (figura 10), isso é necessário para que se possa utilizá-las nos demais *softwares*. Na figura 11 tem-se a programação desenvolvida em 5 linhas:

1. Representa o acionamento do sistema, com as variáveis dos botões LIGA, EMERGÊNCIA e STOP como contatos NA e uma bobina SET para a esteira 1;
2. Linha com a lógica condicional do sensor de visão, quando essa variável é verdadeira e ambos os botões EMERGÊNCIA e STOP não estiverem pressionados a segunda esteira é acionada simultaneamente com o *pusher*;
3. Contém a variável do sensor óptico como contato NA ligado em uma bobina SET para a esteira 3;
4. Linha com a lógica de parada do sistema, com os contatos NF das variáveis dos botões EMERGÊNCIA e STOP;
5. Última que contém o bloco contador, o qual atualiza a variável de contagem de acordo com o estado do sensor de visão e reinicia com o botão LIGA, após atingir o valor 15 aciona a bobina da variável AUX.

2.3.1 Desafios encontrados

Nota-se que os contatos das variáveis dos botões EMERGÊNCIA e STOP operam contrários à lógica convencional, pois o *Factory I/O* já possui esses elementos configurados de modo que ao serem pressionados os estados alternam para valor *boolean* 0. Assim, teve-se que considerar esse fator ao fazer o código, pois no Codesys, quando essas variáveis são contatos NA, os seus valores lógicos são 1 quando não acionadas, e vice-versa.

Figura 10 – Lista das variáveis globais - Codesys



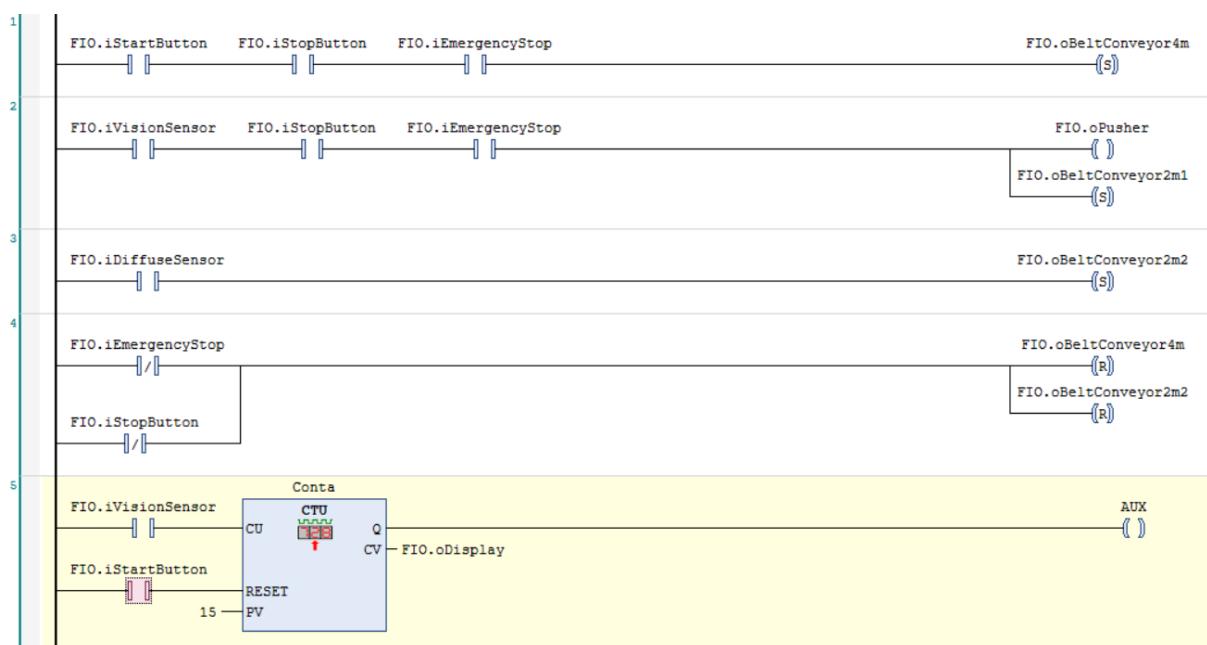
```

1  {attribute 'qualified_only'}
2  VAR_GLOBAL
3      iStartButton : BOOL;
4      iEmergencyStop: BOOL;
5      iStopButton : BOOL;
6      iVisionSensor: BOOL;
7      iDiffuseSensor : BOOL;
8      oDisplay : WORD;
9      oBeltConveyor4m: BOOL;
10     oBeltConveyor2ml : BOOL;
11     oBeltConveyor2m2 : BOOL;
12     oPusher : BOOL;
13 END_VAR

```

Fonte: Elaborada pelas Autoras.

Figura 11 – Código em Ladder - Codesys



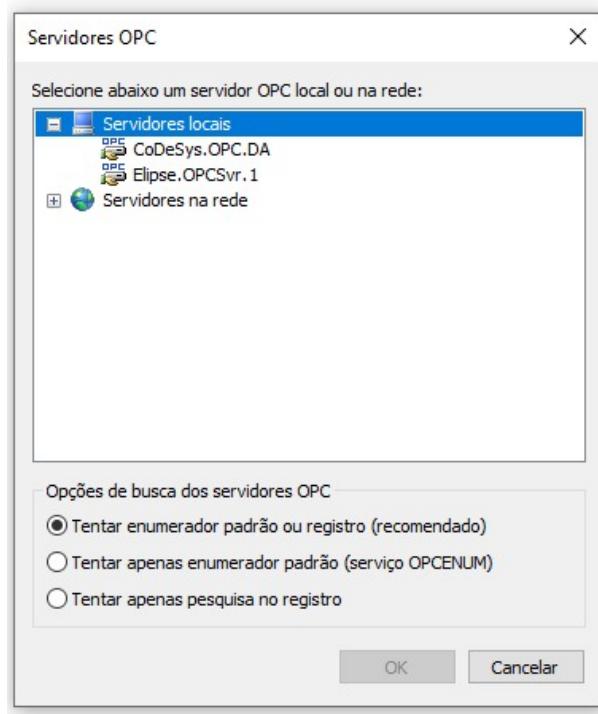
Fonte: Elaborada pelas Autoras.

Além disso, estavamos com versões diferentes do *software*, um mais atual do que o outro, e por consequencia o que era desenvolvido no mais atual, não podia ser aberto na outra versão. O que dificultou o trabalho em dupla.

2.4 Implementação da Comunicação OPC

As primeiras tentativas para estabelecer uma comunicação entre os *softwares*, foram pelo protocolo OPC, porém encontramos diversos desafios descritos melhor na subsessão 2.4.1. Em geral, a comunicação com esse protocolo deve ser simples e rápida. Inicialmente, deve-se checar se o OPC *Configuration* está ligado. A partir dele, adiciona-se apenas um drive de comunicação específico para OPC (figura 12), tanto no ElipseE3 quanto no *Factory I/O* essa configuração é mais direta que as demais, por exemplo, no primeiro não é necessário acrescentar arquivos externos para tal, diferentemente do protocolo Modbus explicado na subsessão 2.5.

Figura 12 – Comunicação OPC



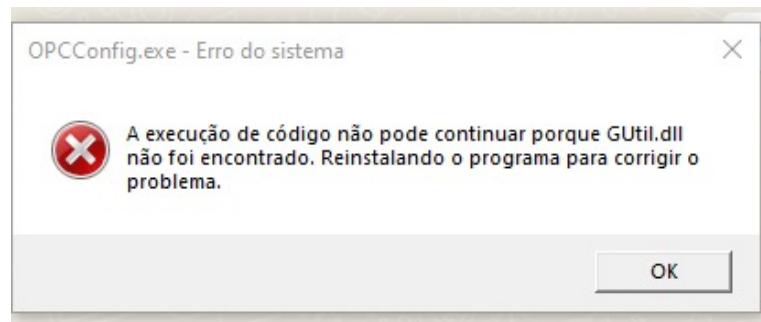
Fonte: Elaborada pelas Autoras.

2.4.1 Desafios Encontrados

Este protocolo de comunicação que em teoria deveria ser simples, tornou-se o mais complicado, utilizou-se 3 versões diferentes do Codesys, em duas delas, a versão possui um *bug* com os drives do protocolo, como pode ser observado nas figuras 13 e 14. A terceira versão que disponível, era mais antiga (V3.5.11) do que a que foi desenvolvido o

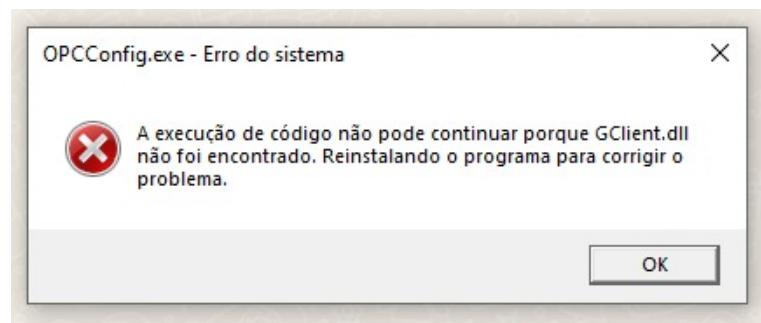
programa (V3.5.15), logo, abrir a programação era inviável, inclusive tentou-se instalar a versão V3.5.15.BF2, a qual tem correção de diversos *bugs*. Mesmo com essas versões mais atualizadas a opção de selecionar o Codesys como servidor OPC, figura 12, simplesmente não aparecia, devido aos erros de drives. Os drivers que garantiam o funcionamento dessa comunicação tiveram que ser instalados separadamente, porém eles funcionavam exponencialmente. O que se mostrou impraticável, e foi de grande influência para migração para outro protocolo de comunicação. Por fim, após diversas tentativas foi possível fazer funcionar, figura 12, em um computador do laboratório, o qual era acessado apenas de modo remoto, o que com os três simuladores em operação simultaneamente tinha muito *lags*, e tornava o desenvolvimento do projeto inviável.

Figura 13 – Erro drive GUtil.dll OPC



Fonte: Elaborada pelas Autoras.

Figura 14 – Erro drive GClient.dll OPC



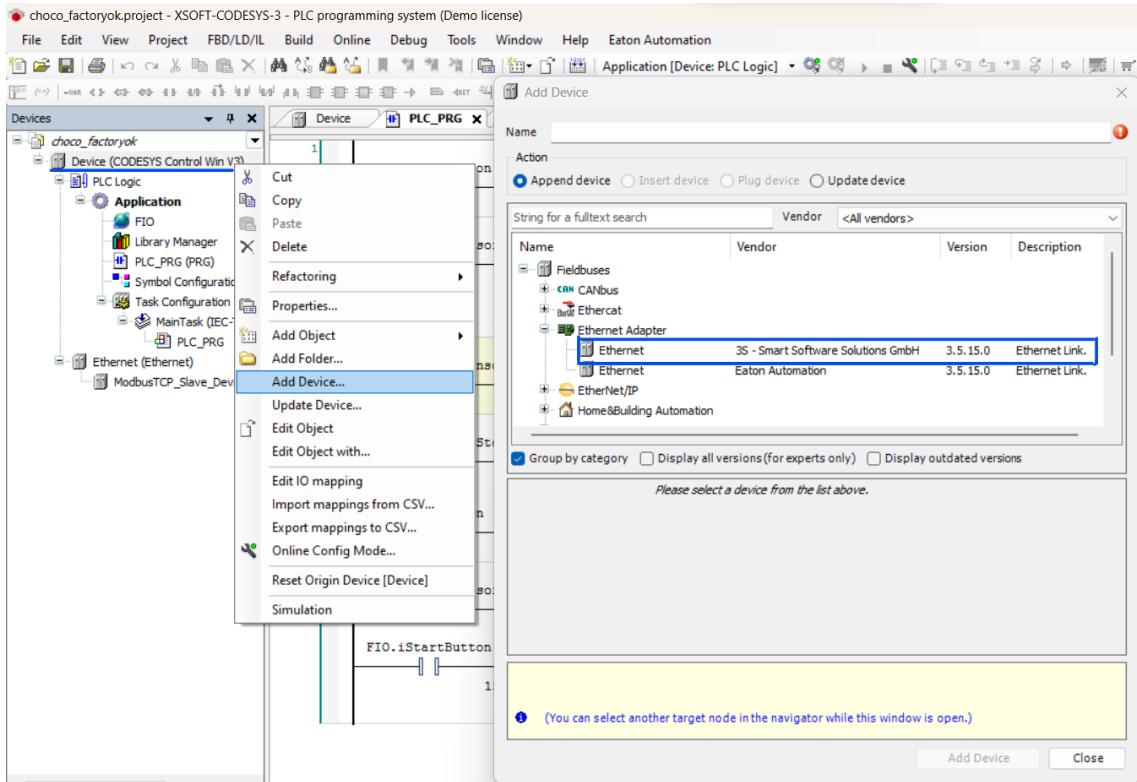
Fonte: Elaborada pelas Autoras.

2.5 Implementação da Comunicação Modbus

Devido às dificuldades em realizar a comunicação OPC citadas anteriormente, optou-se por implementar em paralelo o protocolo Modbus TCP/IP, em que o ElipseE3 opera como mestre e ambos o Codesys e o *Factory I/O* são escravos. Para isso, configurou-se o Codesys conforme o passo a passo descrito na documentação do *Factory I/O* (I/O,

2023b), adicionou-se um dispositivo de Ethernet, como mostra a figura 15, dentro desse elemento acrescentou-se um dispositivo de Modbus escravo (figura 16).

Figura 15 – Adicionando dispositivo Ethernet no Codesys



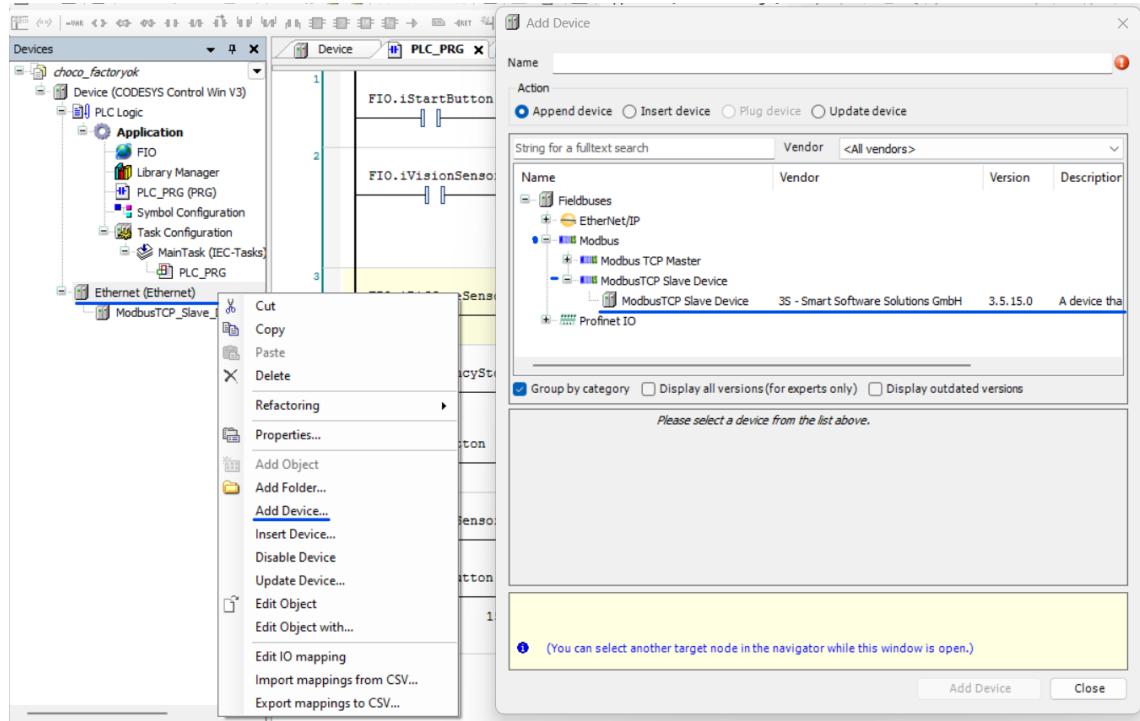
Fonte: Elaborada pelas Autoras.

Com esses itens inseridos, restou configurá-los com os dados de endereço IP e tipo de interface (WiFi, Ethernet ou Bluetooth) no módulo de Ethernet (figura 17a), no elemento escravo Modbus definiu-se a porta de leitura (502 - porta padrão de comunicação TCP Modbus (TOUCH et al., 2023)) e a quantidade de *Holding* e *Input registers*, como mostra a figura 17b. Após essa etapa, realizou-se o endereçamento das variáveis (figura 18) com relação direta à ordem do *Factory I/O* (figura 19, observa-se que a única do tipo *word* (valor inteiro de 16 bits) é a que recebe os valores do contador, as demais são do tipo *bool*.

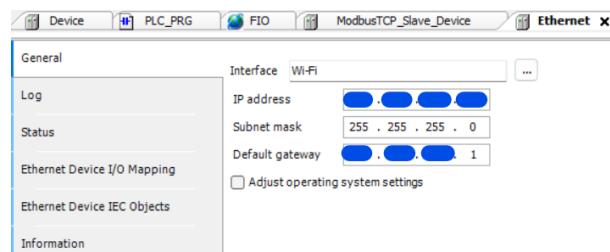
2.5.1 Desafios Encontrados

Ao longo da implementação do protocolo de comunicação Modbus as integrantes enfrentaram diversas dificuldades, tanto em relação a determinar qual *software* seria o mestre e quais seriam os escravos, quanto no endereçamento das variáveis. Mesmo utilizando alguns tutoriais como guias, algumas etapas foram complicadas de solucionar, principalmente porque vários erros nos testes ocorreram sem que houvessem alterações nos parâmetros de versões anteriores, sendo apenas falhas apresentadas pelos *softwares* de maneira aleatória, pelo que se percebeu. Além desses,

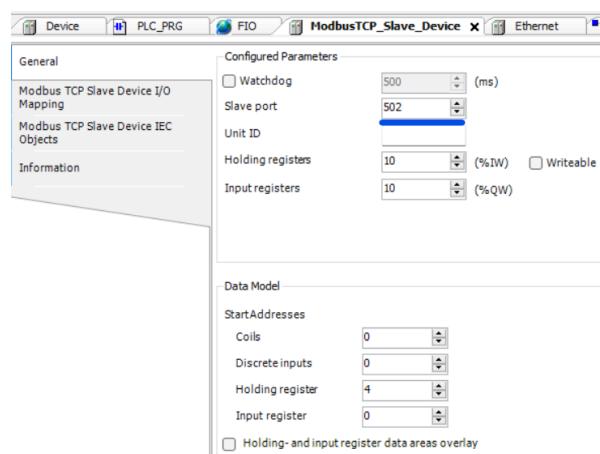
Figura 16 – Adicionando dispositivo Modbus - slave no Codesys



Fonte: Elaborada pelas Autoras.



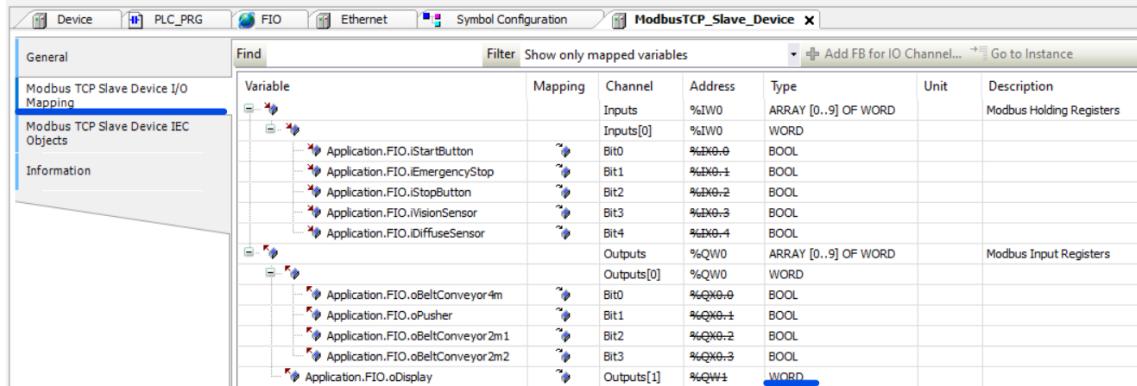
(a) Configuração endereço IP no Codesys



(b) Configuração da porta do dispositivo escravo Modbus

Figura 17 – Telas de configuração Modbus do Codesys

Figura 18 – Endereçamento das variáveis no Codesys



Fonte: Elaborada pelas Autoras.

Figura 19 – Protocolo Modbus no Factory I/O



Fonte: Elaborada pelas Autoras.

2.6 Sistema Supervisório - Elipse

O objetivo de implementar o Sistemas de Supervisão e Aquisição de Dados (SCADA) é monitorar as variáveis e os dispositivos que utiliza-se na simulação do processo de separação dos chocolates. Por isso priorizou-se que o máximo possível que todas as variáveis pudessem ser identificadas da forma mais visual e organizada possível.

Para isso, construiu-se 3 telas diferentes, a primeira delas, figura 20, é uma tela de menu inicial que possui 2 botões que direcionam o usuário para as outras telas. A dinâmica de mudanças de telas pode ser observado na subsessão 2.6.1.

A segunda tela, figura 21, possui o objetivo de observar ativamente o painel de controle, o botão de emergência funcional, que quando acionado pisca em forma de sinalização. Os botões de *start* e *stop* também funcionais que acendem a cor de forma mais vibrante quando pressionados, e por fim, adicionou-se um contador para que se possa visualizar a quantidade de peças azuis, que representam o Molde 1, foram produzidas.

Como pode ser observado na figura 22, a terceira tela é de visualização do processo em geral, para que se possa acompanhar o correto funcionamento de todos sensores e atuadores da planta. Portanto, tem-se a luz azul central acendendo a partir do *vision sensor* quando detecta pela cor o Molde 1.

Também se considerou de extrema importância, saber quando e qual esteira está sendo acionada ou não conforme deveria, o que permite um controle de custos energéticos e é possível observar se alguma delas está com defeito pois não está operando corretamente. Um exemplo de quando algumas estão acesas, figura 23, ou seja de forma perceptível observa-se qual dos motores está acionado.

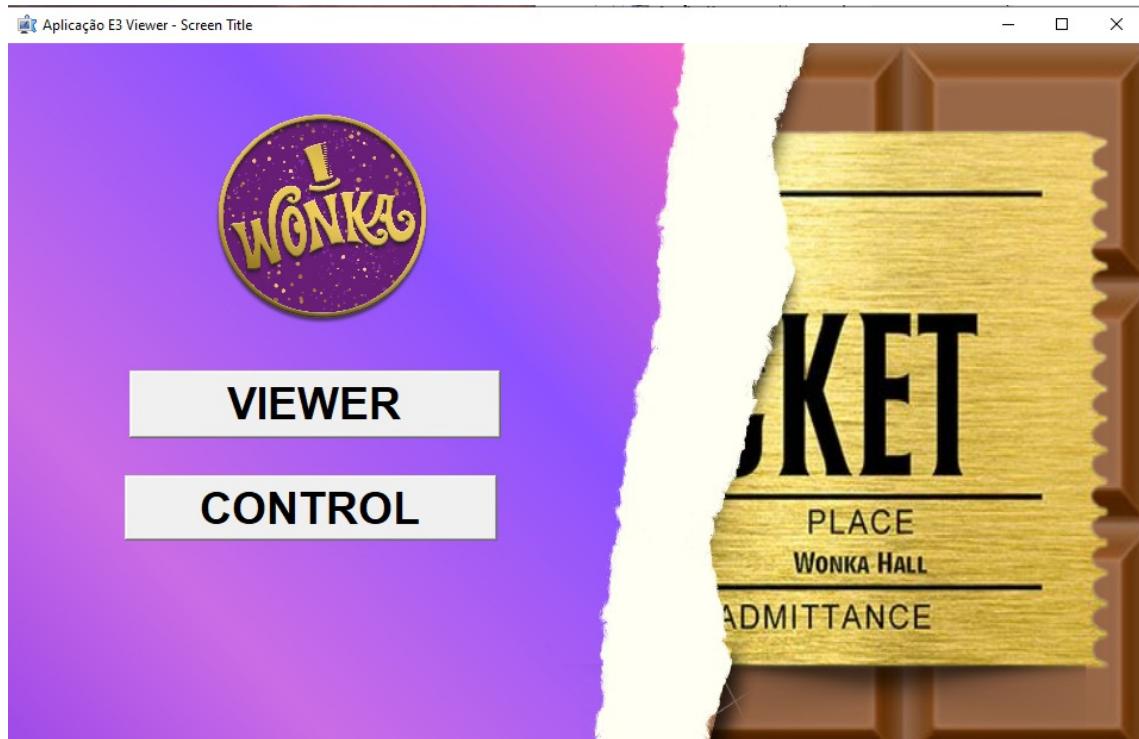
2.6.1 Dinâmica implementada nas trocas de telas

Para configurar uma tela para cada objetivo de visualização, como descrito anteriormente, é necessário configurar cada um dos botões de forma individual. Para isso, como mostrado na 24, é necessário escrever a tela de destino e alguns padrões de alinhamento. Todos foram configurados com o tipo *click* embora existam outras opções. É importante destacar que neste supervisório, se qualquer tela for renomeada depois dos botões configurados, é necessário exclui-lo, pois o *software*, neste caso específico, não consegue entender essa mudança.

2.6.2 Configuração das tags

Foi necessário adicionar um *driver* Modbus, para isso é necessário que o diretório onde se encontram o domínio e projeto do ElipseE3 contenha um arquivo Modbus.dll, essencial para o protocolo de comunicação escolhido. Além disso, é preciso configurar conforme apresenta a figura 25, na figura 25b seleciona-se o protocolo (Modbus TCP) e ajusta-se para que os endereços das *tags* iniciem em 0.

Figura 20 – Tela Menu



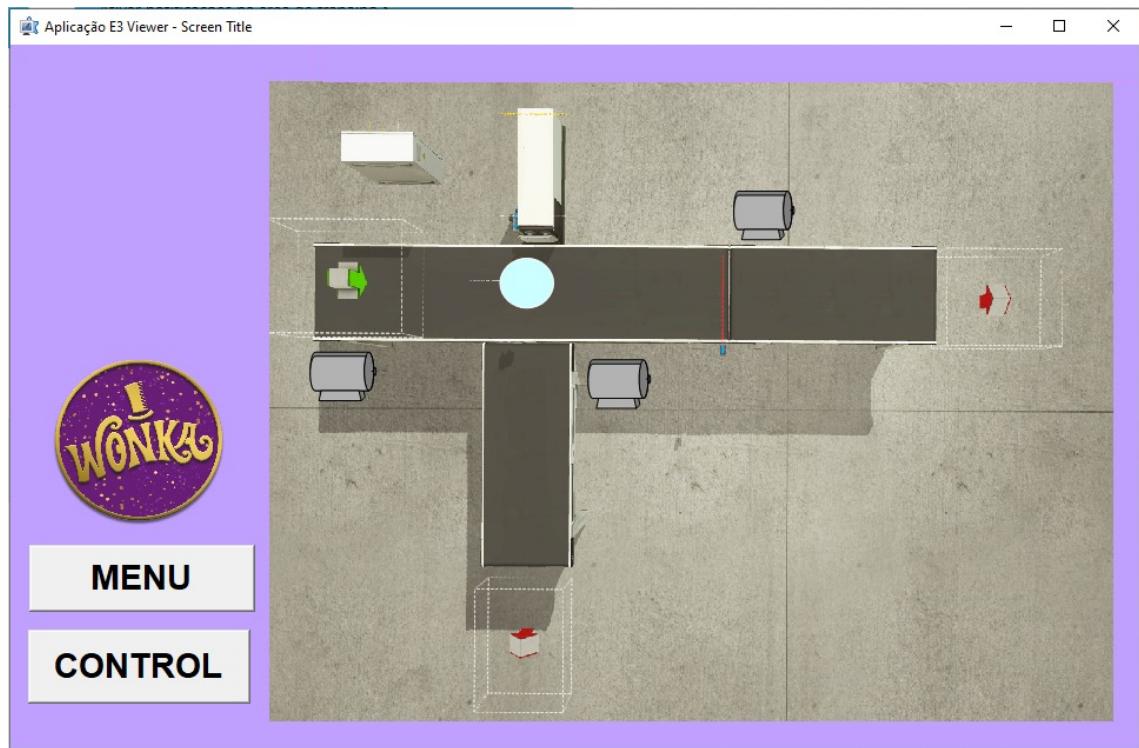
Fonte: Elaborada pelas Autoras.

Figura 21 – Tela Control



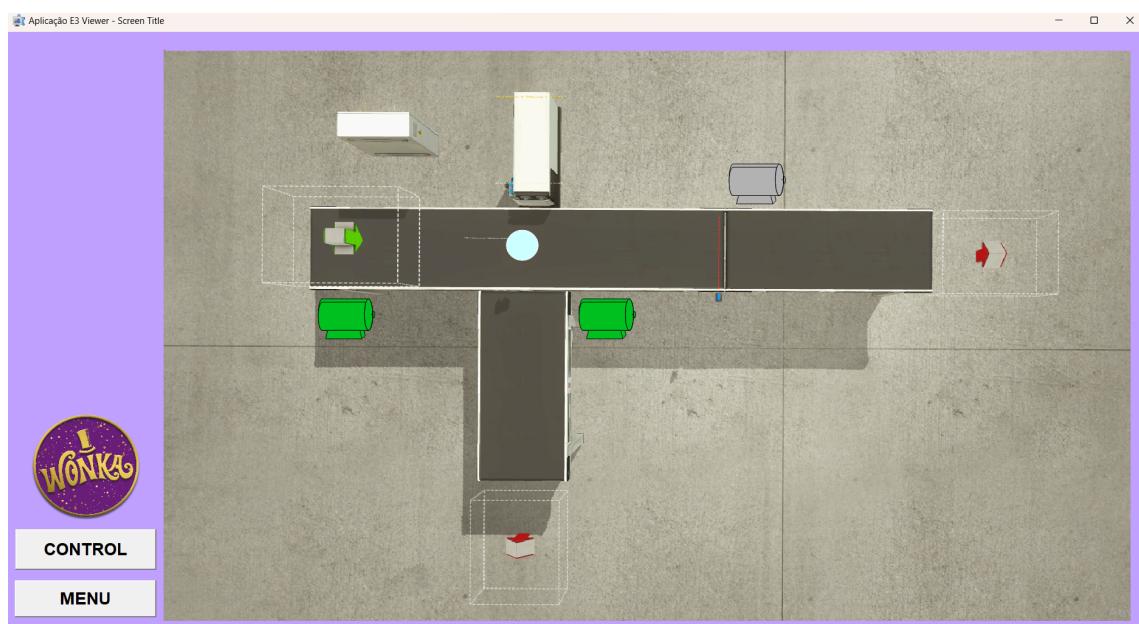
Fonte: Elaborada pelas Autoras.

Figura 22 – Tela Viewer



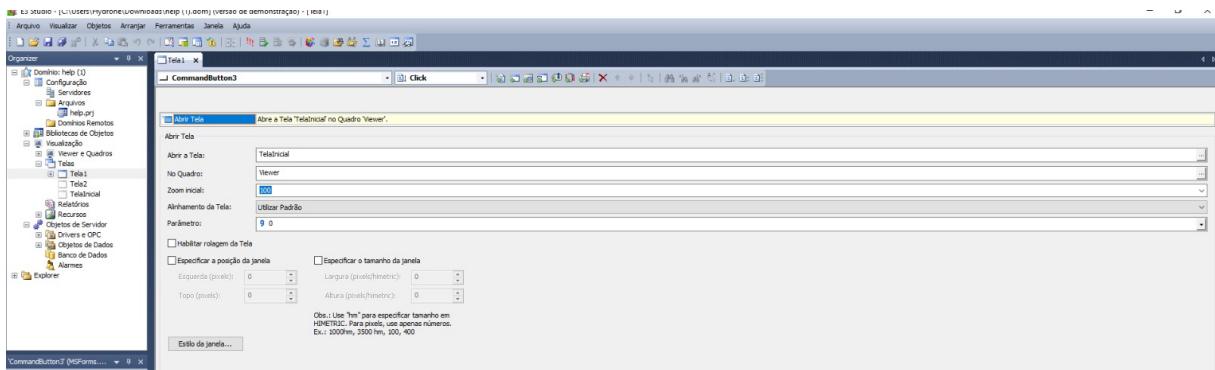
Fonte: Elaborada pelas Autoras.

Figura 23 – Motores Acionados, Tela Viewer



Fonte: Elaborada pelas Autoras.

Figura 24 – Configuração botões para troca de telas



Fonte: Elaborada pelas Autoras.

Na aba *Operations* tem-se os tipos de operação que podem ser atribuídos às *tags*, nesse caso incluiu-se a 8 com o modo *Read* em 4 (função de leitura de *word* (SOFTWARE, 2022b)) e o tipo de dado como *word*, isso se deu para que fosse possível ler a variável do contador. As demais *tags* foram definidas com as operações 6 e 7 de escrita e leitura, respectivamente, conforme o tipo de variável (*input* ou *output*) como mostra a figura 26 (SOFTWARE, 2022a).

Seguindo na configuração do *driver*, seleciona-se a opção Ethernet em *Physical Layer* na aba de *Setup* (figura 25c, e com isso, deve-se definir o IP, nesse caso escolheu-se utilizar o *localhost* (127.0.0.1) e a porta de leitura (502), além do transporte - TCP/IP, apresentado na figura 25d. Assim, o driver está pronto para ser conectado.

2.6.3 Desafios Encontrados

A principal dificuldade que as autoras se depararam ao elaborar o sistema supervisório foi nos endereços das *tags*, pois em um dos testes iniciais utilizou-se os valores incorretos o que causou uma anomalia no ElipseE3 de modo que a variável recebia uma quantidade muito alta de dados resultando em uma sobrecarga no *software*. Até entender qual era o problema perdeu-se algum tempo nessa questão.

Outro ponto de adversidade enfrentado foi em relação ao compartilhamento dos arquivos entre as duas integrantes, pois ao abrir os arquivos em diferentes computadores observaram-se algumas inconsistências, o que era inconveniente para o trabalho em dupla.

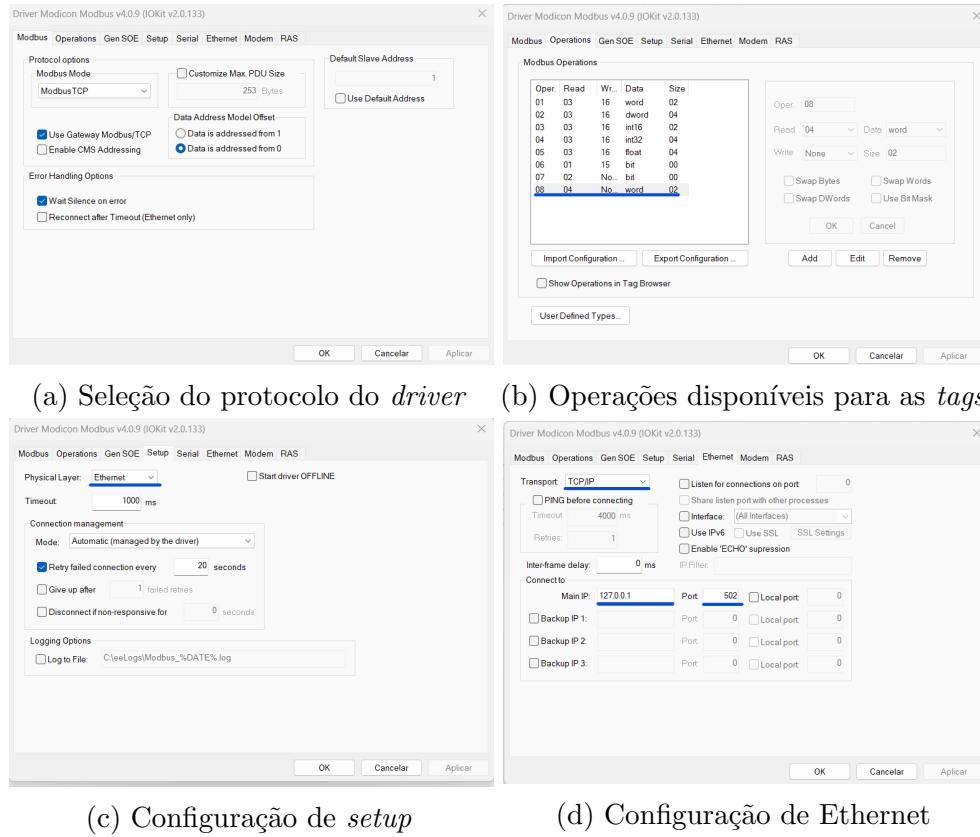


Figura 25 – Telas de configuração do driver Modbus do ElipseE3

Figura 26 – Lista de *tags* do ElipseE3

Nome	Disp...	Item	P1/N...	P2/N...	P3/N...	P4/N...	Ta...	Varr...	Leitur...	Escrit...
Driver1			0	0	0	0				
• LIGA			1	6	0	0	1000	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
• EMERGENCY			1	6	0	1	1000	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
• STOP			1	6	0	2	1000	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
• COLOR_SENSOR			1	7	0	1	1000	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
• LASER_SENSOR			1	6	0	4	1000	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
• ESTER4m			1	7	0	0	1000	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
• ESTER2m1			1	7	0	2	1000	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
• ESTER2m2			1	7	0	3	1000	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
• DISPLAY			1	8	0	1	1000	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	

Fonte: Elaborada pelas Autoras.

3 Resultados

Considera-se que o objetivo foi alcançado, tendo obtido como resultado uma simulação funcional de uma planta simplificada da etapa de separação de moldes de chocolate. Conta-se com a elaboração de uma cena original no *Factory I/O*, o desenvolvimento de um código em Ladder para CLP no Codesys e a implementação de um sistema supervisório através do ElipseE3, com o destaque da comunicação entre os três *softwares* ser pelo protocolo Modbus TCP/IP. O funcionamento final consiste no acionamento da primeira esteira ao apertar o botão Liga, assim que uma peça azul passa pelo sensor de visão o elemento *pusher* é ativado em conjunto com a segunda esteira, encaminhando o item para direita. Quando uma peça verde chega ao final da primeira esteira, ela é detectada pelo sensor óptico a iniciar a esteira 3 para que o produto chegue ao final da linha. Em relação aos elementos de segurança instalados, tem-se o botão Stop, que quando pressionado para todas as esteiras até o sistema ser reativado, e o botão de Emergência, o qual deve ser destravado antes da fabricar retomar operações. Além disso, tem-se um contador em sincronia com o sensor de visão, de modo a incrementar seu valor de acordo com a detecção de moldes azuis. É possível ver o processo em operação no vídeo elaborado pelas autoras no *link*: <<https://youtu.be/Y7egmpI6R4Q>>.

4 CONCLUSÃO

Conclui-se que embora a cena de simulação tenha sido simples, aplicou-se boa parte dos recursos disponíveis que se utiliza em empresas reais atualmente. Em relação a isso podem ser citados os itens fundamentais como um botão de emergência, e também o protocolo de comunicação Modbus, visto que esse é um dos mais utilizados em ambientes industriais. Logo, observa-se que a simulação desenvolvida nesse trabalho trouxe uma percepção de processos industriais dentro da área da Automação, uma vez que plantas maiores tendem a ter processos extremamente parecidos.

Em determinadas etapas, enfrentou-se diversos problemas que foram superados, e que resultaram na utilização do Modbus, por consequência a ideia de "mestre - escravo" ficou muito mais clara. Portanto, adquiriu-se também uma vasta experiência e conhecimento com programação em Ladder, e o funcionamento de um sistema supervisório (SCADA).

4.1 Trabalhos Futuros

Em geral propõem-se como aprimoramento do trabalho, a ampliação da planta, para que haja separação de mais cores, a implementação de mais contadores, para que se possa ter uma maior coleta de dados possíveis. Para isso, seria necessário dedicar-se a estudar melhor os recursos disponíveis nos *softwares* utilizados, ou até mesmo analisar outros simuladores que fossem mais adequados ao processo investigado.

Referências

- CHOCOLAT, E. **Lesson—How Chocolate is Made.** 2023. Disponível em: <<https://www.ecolechocolat.com/en/how-chocolate-is-made.html>>. Acesso em: 28 de janeiro de 2023. Citado na página 1.
- I/O, F. **Sensors.** 2023. Disponível em: <<https://docs.factoryio.com/manual/parts/sensors/>>. Acesso em: 28 de janeiro de 2023. Citado na página 4.
- I/O, F. **Setting up CODESYS Modbus TCP (SP15 or lower).** 2023. Disponível em: <<https://docs.factoryio.com/tutorials/codesys/setting-up/codesys-mb-sp15/>>. Acesso em: 28 de janeiro de 2023. Citado na página 11.
- OLIVEIRA, A. **7 etapas da fabricação de chocolate.** 2023. Disponível em: <<https://www.cpt.com.br/artigos/7-etapas-da-fabricacao-de-chocolate>>. Acesso em: 28 de janeiro de 2023. Citado na página 1.
- SOFTWARE, E. **Aba Operations.** 2022. Disponível em: <https://docs.elipse.com.br/documents/pt-br/driver/modbus/latest/modbus_config_property_operations_tab.html>. Acesso em: 28 de janeiro de 2023. Citado na página 17.
- SOFTWARE, E. **Funções Suportadas.** 2022. Disponível em: <https://docs.elipse.com.br/documents/pt-br/driver/modbus/latest/index.html?modbus_supported_function.html>. Acesso em: 28 de janeiro de 2023. Citado na página 17.
- TOUCH, J. et al. **Service Name and Transport Protocol Port Number Registry.** 2023. Disponível em: <<https://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xhtml?&page=10>>. Acesso em: 28 de janeiro de 2023. Citado na página 11.