CENTRO PAULA SOUZA

FATEC SANTO ANDRÉ

Tecnologia em mecatrônica industrial

Gabriel Bauer de Assis

Leticia de Lourdes Camaforto da Silva

Suzana do Nascimento Gobo

SISTEMA INTEGRADO DE MANUFATURA

Santo André

2024

GABRIEL BAUER DE ASSIS

LETICIA DE LOURDES CAMAFORTO DA SILVA

SUZANA DO NASCIMENTO GOBO

SISTEMA INTEGRADO DE MANUFATURA

Trabalho de Conclusão de Curso apresentado a FATEC SANTO ANDRÉ orientado pelo Prof. Me. Pedro Adolfo Gallani como requisito parcial para obtenção do título de Tecnólogo em Mecatrônica Industrial.

Santo André

2024

Fatec CENTRO PAULA SOUZA GOVERNO DO ESTADO DE SÃO PAULO

Faculdade de Tecnologia de Santo André

LISTA DE PRESENÇA

Santo André, 29 de junho de 2024.

LISTA DE PRESENÇA REFERENTE À APRESENTAÇÃO DO TRABALHO DE CONCLUSÃO DE CURSO COM O TEMA: **"SISTEMA INTEGRADO DE MANUFATURA"** DOS ALUNOS DO 6º SEMESTRE DESTA U.E.

BANCA

PRESIDENTE:
PROF. PEDRO ADOLFO GALLANI _____

MEMBROS:
PROF. FERNANDO GARUP DALBO _____

PROF. PAULO TETSUO HOASHI _____

ALUNOS:

GABRIEL BAUER DE ASSIS Gabriel Bauer

LETICIA DE LOURDES CAMAFORTO DA SILVA Leticia L. Camaforto

SUZANA DO NASCIMENTO GOBO Suzana N. Gobo

Aos nossos familiares, amigos e professores dedicamos este projeto.

AGRADECIMENTO

Agradecemos aos nossos familiares por todo suporte e apoio, aos nossos professores Me. Fernando Garup Dalbo, Me. Pedro Adolfo Gallani, Me. Moacyr da Silva Caminada e a toda equipe Fatec Santo André por disponibilizar parte dos recursos necessários para a construção deste trabalho.

“A maior recompensa para o trabalho do homem não é o que ele ganha com isso, mas o que ele se torna com isso.”

John Ruskin

RESUMO

Este trabalho de conclusão de curso aborda o desenvolvimento e construção de um sistema integrado de manufatura que simula as etapas de pintura e secagem da carroceria de carros. Para demonstração deste sistema, foi utilizado uma esteira para construir um protótipo que simule as etapas de uma linha de produção. A representação dos dados importantes envolvidos no processo será estabelecida pelas receitas, que através de um sistema de leitura por radiofrequência (RFID), serão detectadas e enviadas para o sistema de Supervisão e Aquisição de Dados (SCADA) conforme cada modelo de carro. Desse modo, através da quantidade de camadas de primer, cor do esmalte e as camadas de verniz, o operador poderá identificar qual processo será executado para aquele modelo. Para que todas estas etapas do processo sejam efetuadas, será utilizado o protocolo ModBus RTU, que padroniza a comunicação serial via cabo USB entre o Esp-32 (remota de entradas/saídas) e a Unidade Central de Processamento (UCP) e para o sistema de supervisão, o Protocolo de Controle de Transmissão/Protocolo da Internet (TCP/IP) via Ethernet que comunica o supervisório com o SoftPLC.

Palavras-chave: RFID. Supervisório. SoftPLC. ModBus. Sistema.

ABSTRACT

This final paper addresses the development and construction of an integrated manufacturing system that simulates the painting and drying stages of car bodies. To demonstrate this system, a prototype of an industrial conveyor belt for automotive painting and drying was used. The representation of important data related to the process will be established by recipes, which through a Radio Frequency Identification (RFID) reading system, will be detected and sent to the Supervisory Control and Data Acquisition (SCADA) system according to each car model. Thus, through the number of primer layers, enamel color, and varnish layers, the operator will be able to identify which process will be executed for that model. To ensure all these process stages are carried out, the ModBus-Remote Terminal Unit (RTU) protocol will be used, standardizing serial communication between the ESP32 (remote inputs/outputs) via Universal Serial Bus (USB) cable and the Central Processing Unit (CPU), and for the supervisory layer, the Transmission Control Protocol/Internet Protocol (TCP/IP) protocol via Ethernet, which communicates the supervisory system with the SoftPLC.

Keywords: RFID. Supervisory. SoftPLC. ModBus. System.

LISTA DE ILUSTRAÇÕES

LISTA DE ABREVIATURAS E SIGLAS

RFID	*Radio Frequency Identification*
SoftPLC	*Soft Programmable Logic Controller*
SPI	*Serial Peripheral Interface*
CC	Corrente Contínua
DC	*Direct Current*
SSC	Sistema de Supervisão e Controle
LED	*Light Emitting Diode*
IoT	*Internet of Things*
TCC	Trabalho de Conclusão de Curso
CLP	Controlador Lógico Programável
PLC	*Programmable Logic Controller*
SCADA	*Supervisory Control and Data Acquisition*
IHM	Interface Homem Máquina
I/O’s	*Inputs e Outputs*
E/S’s	Entradas e Saídas
UPC	Unidade Central de Processamento
CPU	*Central Processing Unit*
USB	*Universal Serial Bus*
RTU	*Remote Terminal Unit*
TCP/IP	*Transmission Control Protocol/Internet Protocol*
PWM	*Pulse Width Modulation*

SUMÁRIO

INTRODUÇÃO

Como dito por Pasquini (2020) o mercado industrial constantemente vem se adaptando com o surgimento de novas tecnologias, surgindo de forma periódica grandes inovações, onde sua implementação resulta em relevantes mudanças ao setor industrial. Esse acontecimento recebe o nome de revolução industrial, até hoje houve quatro grandes revoluções industriais, sendo elas, resumidamente:

1° Revolução industrial, ocorreu durante a metade do século XVIII, marcada pela introdução da máquina a vapor e mecanização (PASQUINI, 2020).

2° Revolução industrial, no decorrer do século XIX, conhecida por era da eletricidade e do petróleo e pelo início da produção em massa (PASQUINI, 2020).

3° Revolução industrial, ao longo do século XX, onde começou a introdução dos computadores, automação e eletrônicos dentro das indústrias (PASQUINI, 2020).

4° Revolução industrial, presentemente, conhecida também como Indústria 4.0, conceito dito pela primeira vez pelo alemão Klaus Schwab, que se trata da incorporação de tecnologias como Internet das coisas (IoT), sistemas cibernéticos e a utilização de redes no meio fabril. A indústria 4.0 contém dez pilares como está sendo representado da Figura 1 (ALTUS, 2021).

Figura 1: os dez pilares da indústria 4.0

Fonte: altus, 2024.

Neste trabalho iremos abranger sobre o pilar de integração de sistemas, focando em controle dos dispositivos de campo, do controle e supervisão do processo.

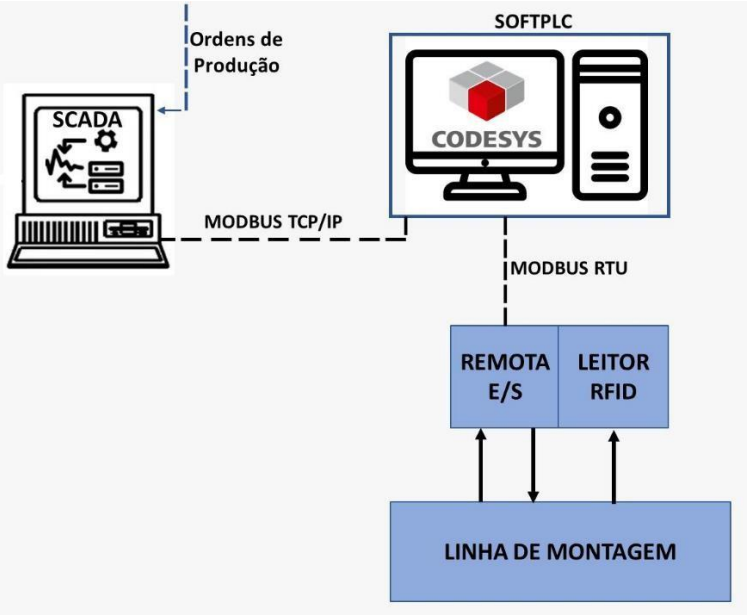
1.1. Objetivo

Este Trabalho de Conclusão de Curso visa demonstrar a simulação de um sistema integrado de manufatura composto pelas etapas de pintura e secagem da carroceria de carros. Para tal, será utilizado um SoftPLC de baixo custo para fins didáticos. O SoftPLC executará diferentes receitas da linha de montagem a partir de identificadores RFID, onde o ESP32 será a remota de entradas e saídas, além de identificar as diferentes tags RFID.

Para monitorar o sistema, será utilizado um software supervisório. O SoftPLC estará interligado ao ESP32 na rede Modbus RTU e ao supervisório na Modbus TCP/IP.

Abaixo, a Figura 2 apresenta a arquitetura geral do projeto a partir do objetivo principal.

Figura 2: arquitetura geral do projeto



Fonte: dos próprios autores, 2024.

2. FUNDAMENTAÇÃO TEÓRICA

Neste capítulo, são abordadas as teorias e estudos necessários para a sustentação do desenvolvimento deste projeto intitulado "Sistema Integrado de Manufatura".

2.1. Sistema integrado de manufatura

Um sistema integrado de manufatura, como o próprio nome sugere, procura conectar toda a área de produção aos demais setores da indústria por meio da tecnologia, tornando os processos mais ágeis, confiáveis e evitando os famosos “gargalos” (SKA, 2023).

Os atuais sistemas integrados de manufatura presentes nas indústrias são formados por abundantes subsistemas responsáveis por realizar tarefas específicas para a realização do sistema total. Um exemplo de subsistema típico seria o armazenamento de materiais, a movimentação de uma esteira ou mesa giratória, o controle de supervisão, entre outros (GROOVER, 2001 apud PAULA; SANTOS, 2008).

2.2. Linha de produção

Como apontado anteriormente, um dos marcos que caracterizou a segunda revolução industrial foi o início da produção em massa. A visão de um método industrial que permitisse aumentar significativamente a quantidade de peças produzidas, tornando o produto final mais barato, foi de Henry Ford, conhecido como o pai do Fordismo (SZEZERBICKI et al., 2004).

Fordismo trata-se de um sistema de produção em série, criado por Ford durante o século XX, que previa a inclusão de linhas de montagem com movimentos contínuos, padronização do maquinário e da mão de obra. Isto é, Ford teve a ideia inovadora de, em vez dos operários irem até o produto que está sendo produzido, o produto ir até os trabalhadores de forma contínua, ou seja, o produto que está sendo fabricado é feito de forma sequencial em várias etapas, parando em estações de trabalho específicas até que o produto esteja devidamente

finalizado, o que diminuía significativamente o tempo de produção. Esse sistema é conhecido como linha de produção (SZEZERBICKI et al., 2004).

2.2.1. Linha de pintura e secagem

Como explicado pela empresa Gans (2014), dentro do processo de fabricação em empresas metalúrgicas, as linhas de pintura e secagem têm um papel crucial. Após a montagem da carroceria, o automóvel passa pela linha de pintura, responsável por aplicar as seguintes etapas:

- Primer: fluido responsável por preparar a carroceria para as seguintes camadas.
- Esmalte: determina a cor final do veículo.
- Verniz: protege o esmalte.

Após a pintura, a carroceria segue para a linha de secagem, onde os fluidos aplicados anteriormente são secos, garantindo assim a proteção e o acabamento adequado ao produto final (GANS, 2014).

2.3. Receita

As receitas, também conhecidas como ordens de produção, são documentos essenciais para a organização e controle do processo de fabricação industrial, onde contêm informações como quantidade a ser produzida, especificações do produto entre outros (ERPFLEX, 2021).

Existem dois tipos principais de receitas, sendo elas:

- Produção contínua: utilizada em fabricações em massa de produtos padronizados, em que a receita é emitida conforme a programação da produção (ERPFLEX, 2021);
- Produção por encomenda: utilizada para fabricações de produtos não padronizados, em que a receita é gerada a partir do pedido de vendas (ERPFLEX, 2021).

Em vista disso, as receitas são recursos indispensáveis em indústrias, para haver controle e organização da produção de forma eficiente, evitando problemas de atrasos, perdas e falhas no estoque (ERPFLEX, 2021).

2.4. Pirâmide de automação

Conforme o fabricante Altus (2024), a pirâmide de automação visa demonstrar de forma gráfica os cinco níveis de controle e de trabalho que há no setor industrial, exibindo de modelo hierárquico que o nível mais baixo contém uma maior quantidade de itens e informações que o topo. Entretanto, ao ir subindo os níveis da pirâmide, em cada etapa, as informações e o fluxo de dados são mais bem trabalhados, ocorrendo um aumento de qualidade. Na Figura 3, temos uma representação da pirâmide de automação, com os nomes de cada campo e com exemplos do que pode ser encontrado neles.

Figura 3: pirâmide de automação industrial

Fonte: altus, 2024.

2.5. Controlador Lógico Programável

Como explicado por Coelho (2011), o controlador lógico programável, também conhecido como CLP ou *Programmable Logic Controller (PLC)*, surgiu em 1968 na General Motors, visando substituir os painéis de comando utilizados na época (painéis de relé) para o controle na linha de montagem.

A necessidade de criar um equipamento ocorreu pela grande dificuldade de mudar a lógica de controle da linha de montagem quando necessário, pois sua complexidade gerava altos gastos de tempo e dinheiro. Assim, surgiu um equipamento de maior versatilidade que vem se

O CLP pode ser representado por três partes, sendo elas: os módulos de entrada, a unidade de processamento e os módulos de saída (SILVA, 2011, p. 1), como demonstrado na Figura 4.

Figura 4: representação das partes que compõe um CLP

Fonte: Campus Pelotas, 2024.

Durante o funcionamento do PLC, é realizada uma sequência de operações que leva o nome de ciclo de varredura. Ao ser ligado, o CLP realiza uma verificação geral de alguns itens, como o reconhecimento dos módulos de entrada e saída que estão ligados a ele, e o estado da memória (inspecionando se há um programa de usuário instalado). Esta etapa leva o nome de inicialização. Se não houver nenhum problema na parte física do controlador lógico programável e se o programa do usuário estiver instalado, o programa de inicialização inicia o programa do usuário. A partir desse momento, inicia-se a realização do ciclo repetitivo que leva o nome de ciclo de varredura, que consiste em verificar os estados das entradas e saídas, armazenar o que foi lido na memória, efetuar a comparação do estado das entradas e saídas com o programa do usuário e utilizar as saídas conforme definido pelo programa (SILVA, 2011, p. 2).

A seguir, a Figura 5 apresenta um diagrama explicativo sobre o funcionamento do PLC.

Figura 5: diagrama explicativo do funcionamento do PLC

Fonte: Campus Pelotas, 2024.

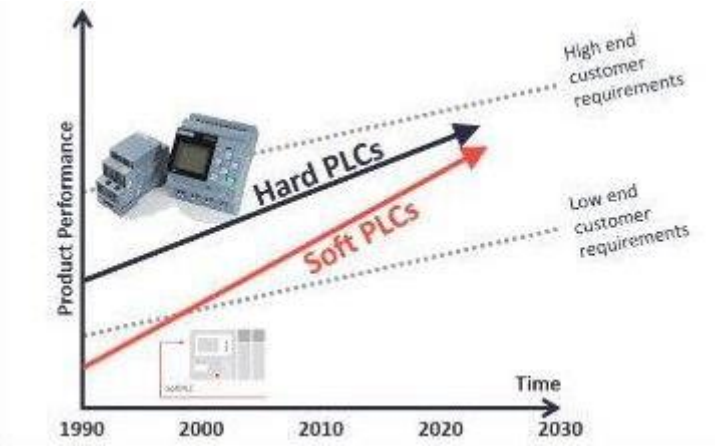
2.6. SoftPLC

Segundo Almeida et al. (apud ASSOCIATION FOR ADVANCING AUTOMATION, 2017), o SoftPLC é uma tecnologia de software que tem como objetivo transformar um computador industrial em um controlador lógico programável. Ou seja, ele permite que o computador realize o controle de E/S, a manipulação dos dados, os recursos computacionais, execute o programa do usuário e se conecte a uma ampla gama de sistemas de E/S e redes, entre outros dispositivos, seguindo todas as normas do padrão IEC 61131, assim como um CLP.

A norma IEC 61131 é o primeiro esforço real para padronizar assuntos relacionados aos controladores lógicos programáveis, como os elementos comuns que se referem aos tipos de dados, variáveis, configurações, recursos e tarefas. Além disso, a norma IEC 61131 também padroniza os programas, os blocos funcionais, as funções e as linguagens de programação (sequenciamento gráfico de funções, diagrama ladder, lista de instruções, diagrama de blocos funcionais, texto estruturado) que um CLP precisa ser capaz de processar (ALTUS, 2024).

Desde os anos 90, quando os SoftPLCs surgiram no mercado, eles têm evoluído significativamente graças às melhorias nos sistemas operacionais, tecnologias de virtualização e hardware de computação, tornando os SoftPLCs atuais muito mais confiáveis, poderosos e flexíveis do que no passado, representando uma ameaça real aos CLPs convencionais (IOT ANALYTICS, 2020). Como demonstrado graficamente na Figura 6.

Figura 6: PLC comparado ao SoftPLC



Fonte: Iot Analytics, 2024.

2.6.1. Remotas I/O

Segundo BRYAN (1997), as remotas de I/O (Input/Output), em português Entradas e Saídas (E/S), são elementos que permitem a expansão de sistemas de automação industrial, transmitindo informações para uma única CPU de um controlador lógico programável que realiza o controle de diferentes áreas da empresa. A remota transmite os sinais das entradas, que podem ser botões, sensores, entre outros equipamentos, para a CPU de um controlador lógico programável e, a partir dele, os sinais são enviados para as saídas conforme os comandos do controlador. As saídas podem ser atuadores, motores, entre outros dispositivos (BRYAN, 1997).

Segundo o fabricante Pepperl+Fuchs (2023), essa tecnologia ajuda a reduzir o tempo de inatividade do processo, pois eventuais manutenções são realizadas com mais praticidade por ser um sistema não centralizado.

2.6.2. Redes de campo

Redes de campo, também conhecidas como Fieldbus, são sistemas de comunicação digital utilizados na automação industrial para interconectar diferentes tipos de dispositivos de campo, como sensores, atuadores e controladores (CAVALHEIRO, 2021).

Albuquerque e Alexandria (2009) apontam que um dos principais fatores para o desenvolvimento das redes de comunicação foi a redução do volume de cabos necessários para interligar os equipamentos de controle na manufatura. Hoje em dia, em alguns casos, é possível transmitir os dados utilizando apenas dois fios.

Alguns exemplos das principais redes são Modbus, Profinet, Profibus DP, entre outras.

2.7. Sistemas supervisórios (SCADA)

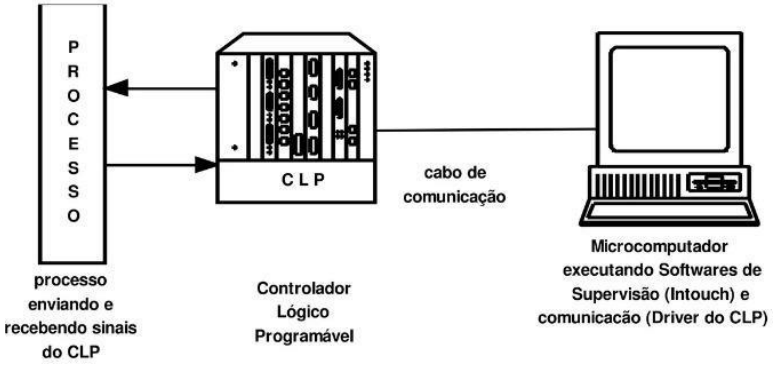
Coelho (2010) informa que os sistemas supervisórios, também conhecidos como SCADA, permitem o monitoramento e rastreamento das informações presentes em um processo com instalações físicas. Essas informações são manipuladas, analisadas, armazenadas e, em seguida, apresentadas ao usuário.

Os sistemas SCADA atualmente utilizam tecnologias de computação e comunicação para automatizar a monitoração e controle dos processos industriais, permitindo a coleta de dados em ambientes complexos, frequentemente geograficamente distantes entre si. A apresentação é realizada de forma fácil de entender para o operador, por meio de recursos gráficos e multimídia (COELHO, 2010).

Para que isso seja possível, os sistemas supervisórios identificam todas as variáveis presentes na aplicação por meio de tags, podendo realizar tanto operações computacionais quanto representar os pontos de entrada e saída de dados do processo que estão sendo controlados. A apresentação ao usuário é feita apenas com os valores das tags dos dados coletados (COELHO, 2010).

A Figura 7 mostra uma representação de um esquema básico do sistema de supervisório.

Figura 7: esquema básico para um sistema de supervisório



Fonte: documento do Instituto Federal Fluminense, 2024.

2.8. Protocolo ModBus

O protocolo Modbus, criado pela empresa Modicon, é um protocolo de comunicação serial aberto, ou seja, qualquer fabricante pode utilizá-lo gratuitamente. O Modbus transmite informações entre dispositivos eletrônicos através de linhas seriais, onde o dispositivo que solicita a informação é chamado de mestre e os dispositivos que fornecem a resposta são denominados escravos (PETRUZELLA, 2014).

Protocolo Modbus fornece comunicação cliente/servidor entre os dispositivos conectados em diferentes tipos de barramentos ou redes, permitindo uma fácil comunicação em todos os tipos de arquiteturas de redes. Além disso, todos os tipos de dispositivos, como PLCs, Interface Homem Máquina (IHM), painéis de controle, drivers, entre outros, podem utilizar o protocolo Modbus para iniciar uma operação remota (MODBUS, 2012).

2.8.1. Modbus RTU

Conforme o fabricante Altus (2021), o protocolo Modbus RTU foi a primeira versão criada. O padrão permite a transmissão de dados e utiliza endereços e valores que são transmitidos de maneira binária.

O Modbus RTU opera de maneira assíncrona, utilizando os meios físicos seriais RS-232 e RS-485. Além disso, cada mestre pode suportar até 247 dispositivos escravos (ALTUS, 2021).

Segundo Kobayashi (2009), o modo RTU contém o campo de início representado por um intervalo silencioso, os campos de endereço e função são representados por 8 bits. O campo de dados, que depende da função utilizada, também é representado por 8 bits. A verificação de erros ocorre através do algoritmo Cyclic Redundancy Check (CRC), representado por 16 bits. Para finalizar, há o campo de fim, que é igual ao campo de início, com um intervalo de silêncio.

2.8.2. Modbus TCP

Segundo Kobayashi (2009), devido à facilidade de modificação do protocolo Modbus, ao longo do tempo foram incorporados novos meios de comunicação dentro do protocolo original, criando variações como o Modbus TCP, que introduz a comunicação Ethernet. A utilização da pilha de protocolos TCP/IP permite a comunicação entre dispositivos com maior conectividade, possibilitando a interligação de diferentes redes e o compartilhamento da mesma infraestrutura.

O Modbus TCP contém apenas dois campos do pacote Modbus tradicional, sendo eles o de função e o de dados. A verificação de erros de transmissão ocorre através das camadas inferiores do Modbus TCP (KOBAYASHI, 2009).

2.9. Protocolo SPI

Conforme explicado por Pinheiro (2018), o protocolo *Serial Peripheral Interface (SPI)* é um método de comunicação utilizado para interconectar diferentes dispositivos ou chips, permitindo a troca de informações entre eles.

O protocolo SPI adota um esquema de comunicação mestre/escravo e pode operar no modo full duplex, ou seja, cada componente utilizando a comunicação SPI pode receber e transmitir dados simultaneamente.

Vale destacar que este protocolo não é padronizado por nenhuma entidade, o que pode resultar em variações de uso dependendo do fabricante. Uma das principais vantagens do SPI é sua capacidade de comunicação rápida, mesmo quando vários dispositivos estão interconectados em um barramento compartilhado. Ao contrário de outros protocolos similares, como o I2C, o SPI não requer um esquema de endereçamento de dispositivos tão elaborado (PINHEIRO, 2018).

2.10. ESP32

O microcontrolador ESP32 foi projetado pela empresa desenvolvedora de tecnologia Espressif Systems, destacando-se no mercado pela sua velocidade de processamento, acessibilidade e conectividade (ESPRESSIF, 2024).

Algumas características do módulo incluem saída de tensão de 3,3 volts, 38 pinos, função *Pulse Width Modulation (PWM)*, conectividade Wi-Fi e Bluetooth, entre outros. Para programar o código, utiliza-se a plataforma IDE do Arduino, sendo o código enviado por meio de um cabo micro USB para o microcontrolador (ESPRESSIF, 2023).

Na Figura 8, há uma representação do ESP32 e suas pinagens.

Figura 8: pinagem do ESP32

Fonte: FVM learning, 2024.

2.11. Sensores

A WEG (2024) explica que os sensores são dispositivos que detectam estímulos do ambiente e os transformam em sinais ou dados, podendo ser tanto digitais quanto analógicos, dependendo do tipo de sensor. Esses dados ou sinais são capazes de ser interpretados por um sistema.

Os sensores em uma indústria são utilizados para extrair dados específicos em tempo real de uma parte do processo. Esses dados são essenciais para garantir o controle eficaz da produção, possibilitando a detecção de possíveis falhas, posicionamento do produto em desenvolvimento, medição de dimensões, entre várias outras aplicações (WEG, 2024).

2.11.1. Sensor infravermelho

O sensor infravermelho é um dispositivo capaz de detectar a radiação infravermelha emitida pelos objetos dentro de seu campo de visão. Ao detectar essa energia infravermelha, ele a converte em energia elétrica, que pode ser interpretada por um sistema. Existem dois tipos principais de sensores infravermelhos, sendo eles ativo e passivo (SERVO, 2023).

O sensor infravermelho ativo emite sua própria radiação infravermelha e detecta a quantidade de radiação refletida pelo objeto dentro de seu campo de visão. Já o sensor infravermelho passivo detecta apenas a radiação emitida pelo objeto dentro de seu campo de visão, sem emitir nenhuma radiação própria (SERVO, 2024).

2.12. Motores DC

Conforme a KALATEC (2024), o motor de corrente contínua, também conhecido como motor CC, é acionado por uma fonte de alimentação de corrente contínua, como o nome pressupõe. Trata-se de um equipamento que converte energia elétrica em energia mecânica, sendo capaz de variar a velocidade do equipamento ao alterar a corrente.

Além disso, a maioria dos motores CC possuem um componente interno que permite a inversão periódica da corrente (KALATEC, 2024).

Por apresentar facilidade em controlar a velocidade e possuir versatilidade para atender desde pequenas aplicações domésticas até grandes projetos industriais, o motor CC acaba se destacando entre as opções de motores (KALATEC, 2024).

2.13. RFID

O RFID, sigla para *Radio Frequency Identification*, que em português significa "identificação por radiofrequência", como o próprio nome sugere, trata-se de uma tecnologia que utiliza ondas de radiofrequência. Para o seu funcionamento são necessários dois elementos: as tags, que contêm um chip com um número de série ou conjunto de dados particular, tornando cada tag única; e o leitor, que possui uma antena capaz de identificar os sinais eletromagnéticos das tags. Dessa forma, quando um objeto com a tag RFID entra no campo de leitura da antena do leitor, este envia sinais eletromagnéticos ao dispositivo capaz de identificar e, se necessário, enviar estes sinais para um sistema de controle (TOTVS, 2022).

Na Figura 9, é demonstrada uma das aplicações que utiliza a tecnologia RFID, um exemplo muito presente no nosso cotidiano.

Figura 9: exemplo de um sistema RFID

Fonte: etnews, 2024.

3. METODOLOGIA

Neste capítulo, serão abordadas as etapas que seguem para o desenvolvimento deste trabalho, estarão escritas de forma sequencial o passo a passo seguido para se chegar ao objetivo final de desenvolver um protótipo de um sistema integrado de manufatura.

Serão apresentados os métodos mais eficazes e viáveis utilizados, definidos através de estudos realizados, pesquisas de mercado e planejamentos, para o desenvolvimento e a construção deste projeto.

Cabe dizer que, o planejamento eletrônico, mecânico e lógico são elaborados a fim de executar funções específicas no protótipo, garantindo que, obtenha-se um projeto tecnológico bem desenvolvido e com todas as partes planejadas implementadas.

3.1. Fluxograma geral a partir do objetivo central

Tendo como objetivo central apresentar um projeto que represente de forma clara o controle de um sistema industrial, este trabalho é um protótipo de uma linha de produção, na qual, é caracterizada por dois processos envolvidos, sendo um deles a pintura e outro a secagem de carrocerias de carros, é comum ser identificado como um sistema integrado de manufatura.

Este é um processo visto como complexo e de alto custo, devido a sua dimensão quando visualizado em uma planta industrial real. Portanto, a criação deste projeto originou-se a partir da oportunidade de demonstrar um meio para realizar o controle deste processo de maneira menos robusta e de menor custo.

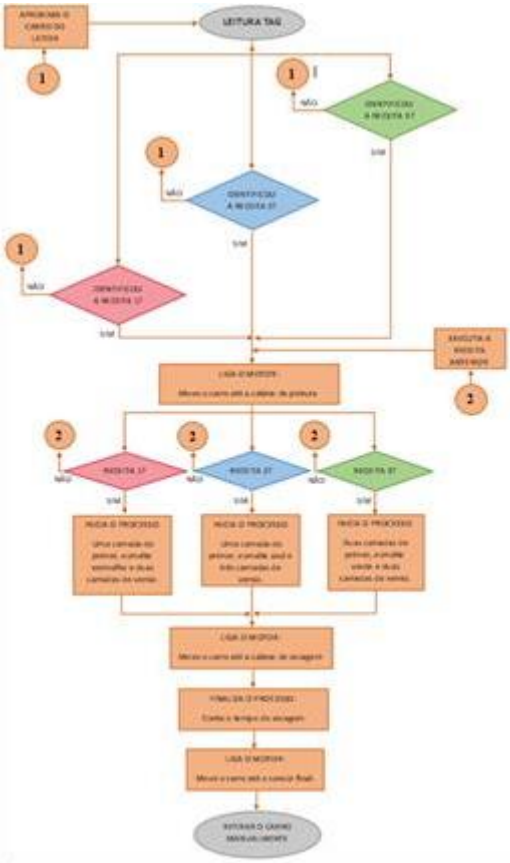
O projeto desenvolvido neste trabalho, funcionará a partir do controle realizado por dispositivos tecnológicos voltados para a área da automação industrial, que são capazes de armazenar informações e/ou compartilhar dados entre outros dispositivos.

A partir de diferentes modelos de carrocerias, o processo simulado desenvolvido será capaz de identificar quais modelos estão sendo incluídos no processo de pintura e secagem.

São processos divididos em cabines, neste caso, uma cabine executa a simulação da pintura e outra cabine, a secagem, porém, ambas são localizadas no decorrer da linha de produção de maneira sequencial.

A seguir, na Figura 10 está apresentado o fluxograma geral que demonstra o funcionamento do protótipo, possibilitando uma visualização clara do que será executado em cada etapa do processo simulado.

Figura 10: fluxograma do funcionamento geral



Fonte: dos próprios autores, 2024.

Para melhor visualização, a Figura 10 está anexada como apêndice A ao final deste trabalho.

3.2. Materiais

Para dar sequência na criação do sistema, é de extrema importância definir e apresentar todos os componentes que serão utilizados para o desenvolvimento do projeto antes de iniciar as etapas que seguirão, portanto, a seguir encontra-se a relação destes materiais:

a) esteira;

b) motor 12V;

c) sensores de presença;

d) módulo leitor RFID Mfrc522;

e) microcontrolador Esp-32;

f) fios de borne;

g) placa de ensaio;

h) fonte 12V;

i) módulo relé 5V;

j) tags RFID programáveis.

3.3. Planejamento inicial para o desenvolvimento do trabalho

Antes de tudo, foi elaborado um cronograma que possui todas as etapas que foram sendo desenvolvidas com o decorrer dos meses, desde os primeiros planejamentos relacionados à construção e desenvolvimento do protótipo, até pesquisas realizadas a fundo sobre estes tipos de processos, além de outros temas relevantes que agregassem aos conhecimentos obtidos neste período.

Este cronograma pode ser visto a seguir, na Figura 11.

Figura 11: sequência das etapas teóricas do desenvolvimento do trabalho

CRONOGRAMA	2023				2024			
	9	10	11	12	1	2	3	4
Conceito e Pesquisa								
Discussão Técnica								
Desenvolvimento								
Teste e Validação								
Publicação								
Apresentação Final								

Fonte: dos próprios autores, 2024.

A partir do cronograma inserido acima, nota-se que os primeiros momentos do trabalho foram as pesquisas e a conceitualização do projeto, ou seja, foi o período em que as ideias ainda estavam sendo discutidas e pontuadas a fim de chegar a uma decisão final.

Seguindo, as discussões técnicas iniciaram após ter uma ideia concreta, este período foi importante para embasar toda a elaboração e desenvolvimento do trabalho, além de que, neste período também foram colocadas em ordem as prioridades de execução e a sequência de criação.

Ao iniciar o ano seguinte, o desenvolvimento se iniciou, tanto teórico como prático, compras de materiais e dispositivos ao mesmo tempo que a parte textual do projeto também estava sendo desenvolvida, tabelas de custos, fundamentação bibliográfica, mais pesquisas de mercado. Até que pudesse obter um bom resultado das discussões realizadas no período anterior a este.

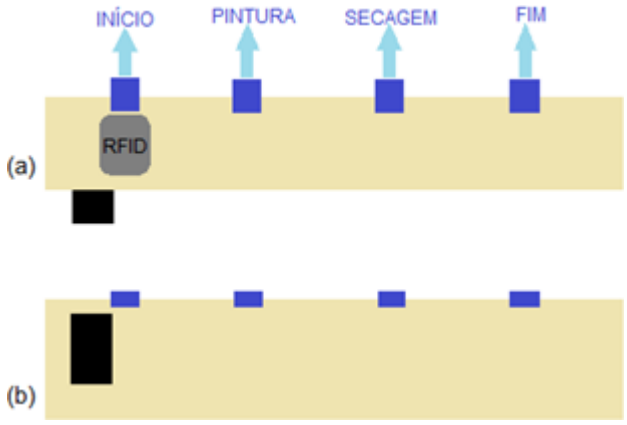
Com tudo implementado, seguem com os testes e validações realizados, tanto pelos próprios alunos como pelos professores responsáveis por orientar e liderar o projeto, isto, até que o trabalho esteja pronto para ser publicado e apresentado posteriormente.

3.4. Planejamento prático

A seguir, estão apresentados todos os planos desenvolvidos pelos integrantes do grupo, que devem iniciar todo o desenvolvimento do projeto, estes planos são divididos em três temas, o mecânico, o eletrônico e o lógico.

Mecânico: o primeiro passo é identificar um local para fixar o leitor RFID e todos os outros dispositivos que serão utilizados, como o ESP32, a fonte de alimentação e os sensores de presença, a Figura 12 apresenta em croqui, uma das maneiras possíveis de posicionamento para cada um destes dispositivos no decorrer da esteira.

Figura 12: (a) vista superior da esteira; (b) vista frontal da esteira



Fonte: dos próprios autores, 2024.

Visualizando a Figura 12, os retângulos azuis representam os sensores que devem ser posicionados de forma estratégica para que realizem a leitura do carro corretamente; o retângulo preto representa a posição na qual o motor deve estar; o leitor RFID deve ser fixado no início

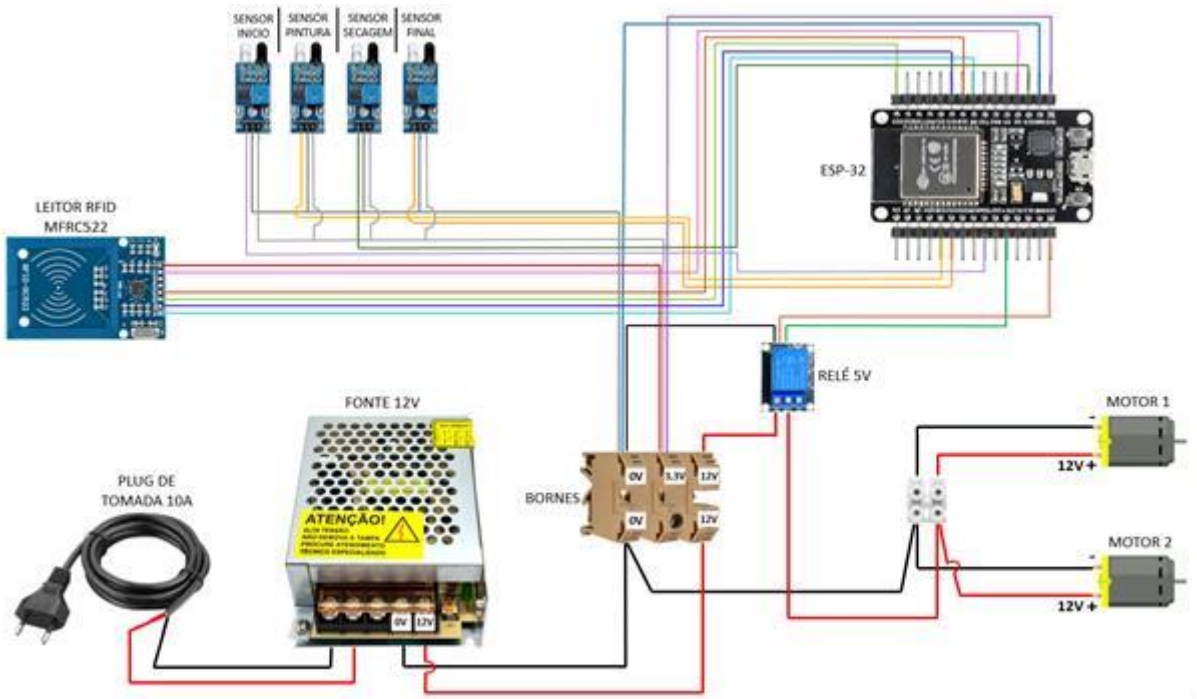
da esteira para identificar o processo a ser executado antes de acionar o motor.

Eletrônico: é importante realizar o dimensionamento do projeto, bem como a quantidade e o comprimento dos fios que serão utilizados, pois assim, evita-se desperdícios de fios ou outros materiais elétricos.

Ao final desta etapa de ligação elétrica, é indispensável a realização de testes utilizando um instrumento de medição que permita verificar se as ligações estão corretas, outro detalhe crucial é, verificar antes de qualquer ligação, no *datasheet* ou no manual dos componentes a tensão máxima que podem receber ou transmitir, para evitar danos aos dispositivos.

A Figura 13 mostra o esquema eletrônico desenvolvido para realizar a ligação dos componentes de forma correta e funcional.

Figura 13: plano de ligação dos componentes



Fonte: dos próprios autores, 2024.

Lógico: neste momento, serão apresentados os cinco passos que seguem para realizar a comunicação correta entre os dispositivos.

Passo 1: comunicar o ESP32 com o leitor RFID, para isto, é necessário criar um programa através da plataforma do Arduíno IDE, que apresente a maneira, na qual, foi possível comunicar o microcontrolador e o módulo leitor.

Esta comunicação deve ser realizada através do protocolo SPI e a partir dele, inserir bibliotecas para o ESP32 e para o leitor RFID, possibilitando localizar estes dispositivos no código e identificá-los.

No programa criado, deve conter a configuração do leitor RFID para identificar uma tag assim que for aproximada ao módulo e ao identificar, deve realizar alguma receita que será executada no decorrer do processo.

São três tags, elas conterão as receitas criadas e por se tratar de uma linha de pintura e secagem de carrocerias, devem ser programadas de acordo com os diferentes tipos de carros que serão representados neste protótipo.

Passo 2: após comunicar o ESP32 com o leitor RFID, é necessário integrar o supervisório, pois nele, dispositivos podem ser monitorados, além de ter todos os dados que estão sendo transmitidos supervisionados de forma remota. Para esta integração, utiliza-se a comunicação serial via cabo USB, a partir do protocolo Modbus RTU entre o microcontrolador e o computador, que é onde está o supervisório.

Na Figura 14 observa-se um croqui de como pode ser desenvolvido o layout das telas criadas para o sistema de supervisão, lembrando que este é apenas um modelo de layout, portanto, pode ser alterado de acordo com a necessidade.

Figura 14: croqui da tela de monitoramento do supervisório

Fonte: dos próprios autores, 2024.

Na tela criada, devem ser inseridos todos os sinais discretos e/ou analógicos que serão monitorados no decorrer do processo executado.

Importante ressaltar que, para cada tela criada no supervisório podem ser utilizados Led’s que representam os dispositivos do processo.

Passo 3: após finalizar os testes com o ESP32 sendo utilizado apenas como um microcontrolador, seguir com os testes utilizando o ESP32 como uma remota de entradas e saídas para o software de programação para projetos de automação, o Codesys. Desta forma, possibilitará a integração destes dois recursos disponíveis a fim de migrar todos os testes feitos anteriormente para serem feitos com o SoftPLC.

O protocolo de comunicação utilizado segue sendo o mesmo, Modbus RTU com meio físico o cabo USB via comunicação serial entre o ESP32 e o Codesys. Pode-se aproveitar a mesma programação utilizada nos testes anteriores, sendo necessário somente realizar alguns ajustes para esta aplicação.

Os sinais trocados entre o ESP32 e a CPU do SoftPLC são sinais digitais, ou seja, não precisam de um monitoramento de tempo em tempo como um sinal analógico, pois uma vez transmitidos, já são analisados e executados os comandos pelo mestre, que é a CPU.

No Codesys, deve haver o mapeamento Modbus de todas as variáveis e para isto, é importante especificar a função 2 para a leitura de entradas discretas ou digitais e a função 5, para a escrita em uma única saída.

Já as faixas de registro, são estabelecidas de acordo com os tipos de variáveis que serão controladas, portanto, deve-se, antes de tudo, ter bem definidas todas as variáveis envolvidas no processo.

Passo 4: para finalizar os testes com o SoftPLC, a partir de um programa localizado no software Codesys, que pode ser criado nas seguintes linguagens, texto estruturado, grafset ou ladder, todos os comandos do processo devem estar inseridos dentro do próprio SoftPLC, sendo controlados pela CPU e executados no sistema.

Realize testes, simulando o acionamento de cada variável através da simulação do software de programação, verifique o de todas as variáveis.

Passo 5: deve ser acrescentado ao projeto o supervisório, a partir do protocolo de comunicação Modbus TCP/IP via ethernet será possível integrar o Codesys ao supervisório.

Realizar o mapeamento Modbus TCP/IP das variáveis do SoftPLC com as que foram criadas no supervisório e atentar-se com a configuração delas.

Conclua a comunicação realizando a simulação do processo, em que o acionamento das variáveis possa ser visto em tempo real por meio do supervisório.

3.5. Receitas

Devem ser definidos três padrões diferentes nomeados como receitas, que representarão os três processos executados para a linha de pintura e secagem de carrocerias. Nestas receitas devem conter as características dos três tipos de carros existentes no sistema.

Para pintura, cada receita deverá exibir as camadas de primer, cor do esmalte (termo técnico para tinta) e as camadas de verniz, isto é, para cada carro.

Para secagem, as receitas devem mostrar o tempo de secagem ideal, visto que, o tempo que o carro ficará no processo de secagem deve ser respectivo à quantidade de camadas aplicadas na carroceria.

No quadro 1, estão apresentadas as receitas escolhidas, diferenciadas de acordo com a necessidade que a simulação requer para ser executada, aplicadas a cada tag:

Quadro 1: identificação de cada receita

Identificação	Processo
Receita 1	Uma camada de primer, esmalte vermelho e duas camadas de verniz.
Receita 2	Uma camada de primer, esmalte azul e três camadas de verniz.
Receita 3	Duas camadas de primer, esmalte verde e duas camadas de verniz.

Fonte: dos próprios autores, 2024.

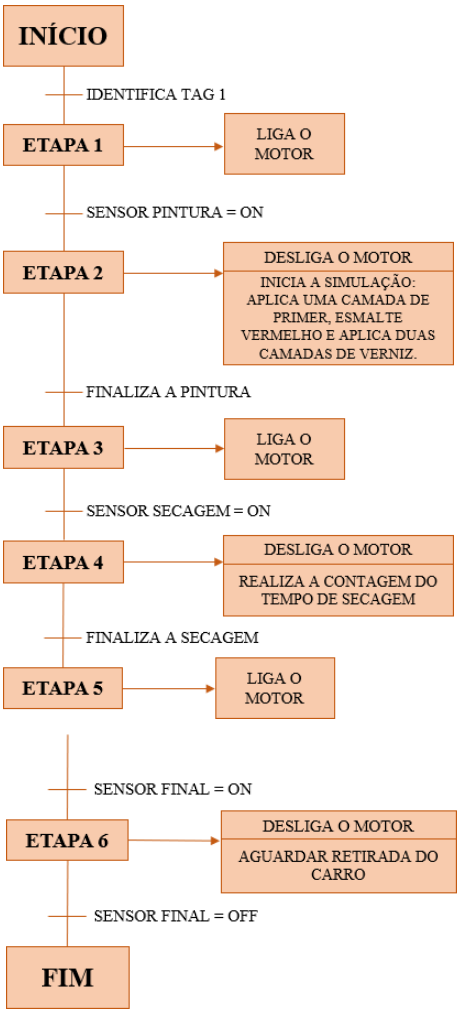
Como dito anteriormente, cada receita será aplicada a uma tag de identificação RFID diferente, neste caso, deve haver três tags, que serão respectivas a cada carro existente no processo.

Nos tópicos seguintes, estarão apresentadas as etapas que cada receita executará, a partir do processo armazenado em cada tag identificada pelo módulo RFID.

3.5.1. Receita 1

A Figura 15 contém a sequência que as etapas da receita 1 devem seguir.

Figura 15: etapas da receita 1

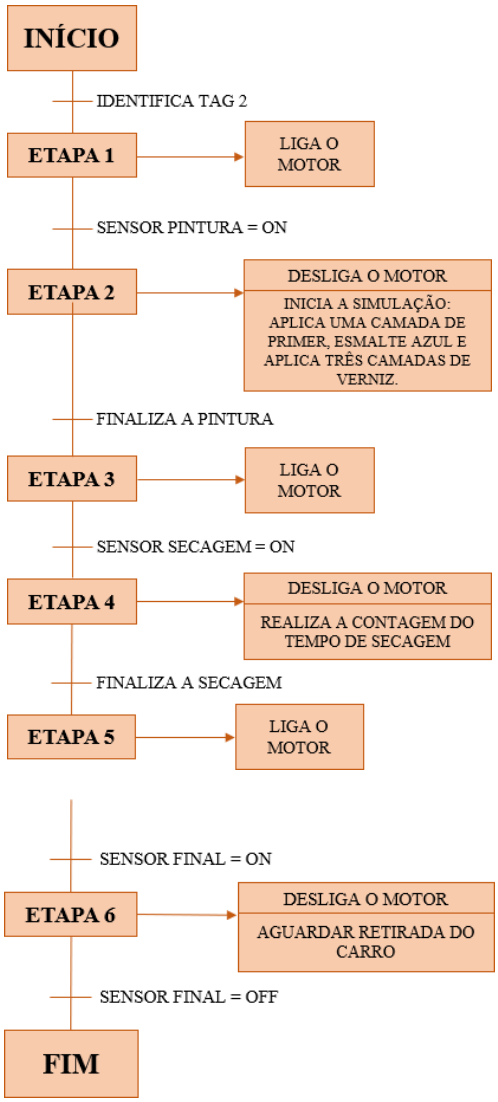


Fonte: dos próprios autores, 2024.

3.5.2. Receita 2

A Figura 16 contém a sequência que as etapas da receita 2 devem seguir.

Figura 16: etapas da receita 2

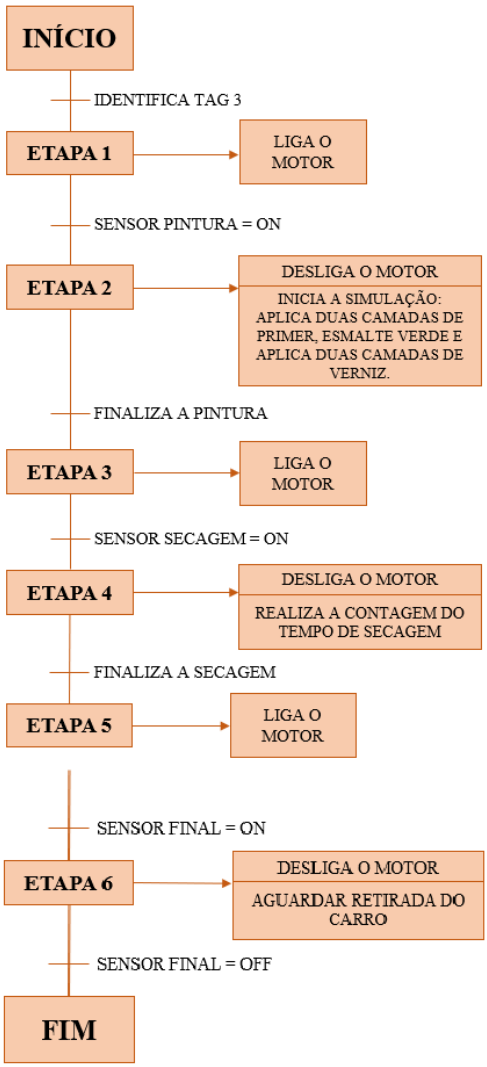


Fonte: dos próprios autores, 2024.

3.5.3. Receita 3

A Figura 17 contém a sequência que as etapas da receita 3 devem seguir.

Figura 17: etapas da receita 3



Fonte: dos próprios autores, 2024.

4. DESENVOLVIMENTO

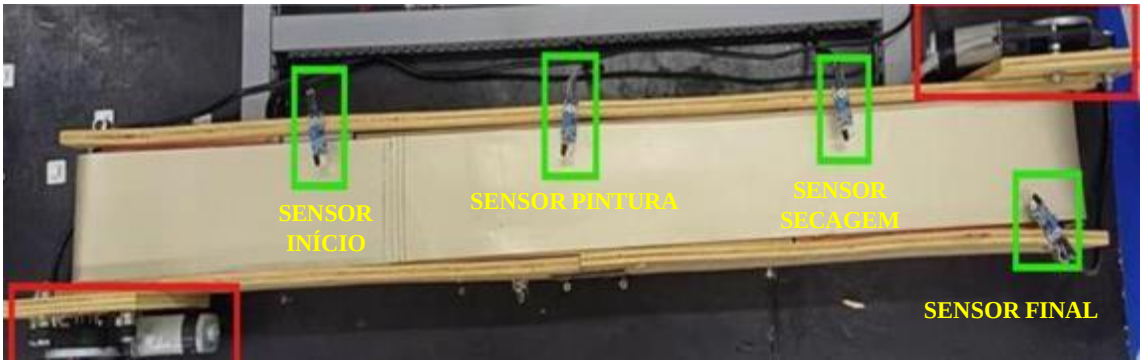
Dado que o foco principal do projeto é demonstrar a simulação de um sistema integrado de manufatura, composto por diferentes receitas predefinidas, foram empregadas diversas tecnologias para essa construção, além de uma construção mecânica adequada para garantir o bom funcionamento e a precisão nos movimentos relacionados aos processos envolvidos.

Portanto, neste capítulo, descreveremos de forma detalhada todas as etapas necessárias para alcançar o objetivo final deste trabalho.

4.1. Construção mecânica

Para a demonstração do projeto, foi utilizada uma esteira de 110 x 13 cm, a qual foi adquirida devidamente montada com dois motores de corrente contínua, um em cada extremidade da esteira. Em seguida, foram fixados quatro sensores infravermelhos na parte superior da esteira. Como pode ser visto na Figura 18, os motores são representados em vermelho e os sensores em verde.

Figura 18: esteira com os motores e sensores



Fonte: dos próprios autores, 2024.

Além disso, foi acrescentado um módulo RFID embaixo do couro, na frente do sensor de início. Por fim, foi adquirido um suporte de madeira de 135 x 45 cm para a fixação da esteira, do ESP32 e de toda a parte elétrica do projeto. A Figura 19 demonstra tanto o suporte com todas as devidas partes do projeto fixadas quanto a localização do módulo RFID em vermelho.

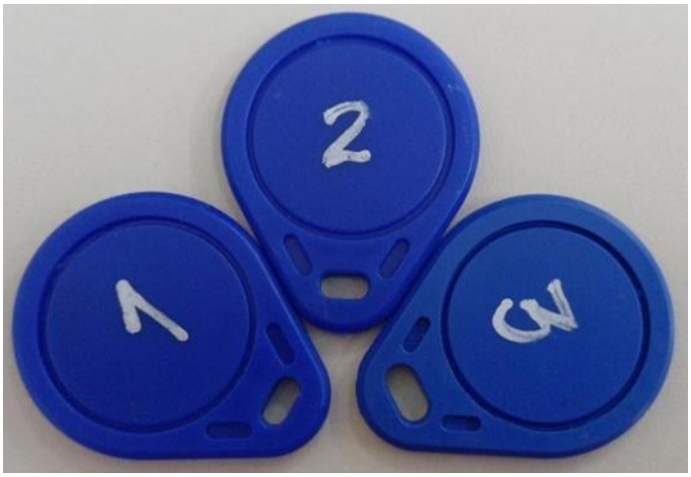
Figura 19: localização do RFID e visão total da parte física do projeto

Fonte: dos próprios autores, 2024.

4.2. Identificação do UID das tags RFID

Ao adquirir um leitor RFID para microcontroladores, é necessário obter tags RFID que tenham a mesma frequência que o leitor, para que seja possível realizar a identificação. O estilo de tag escolhido para o desenvolvimento deste projeto pode ser visto na Figura 20. Foi utilizada uma tag para cada receita de pintura, totalizando três tags.

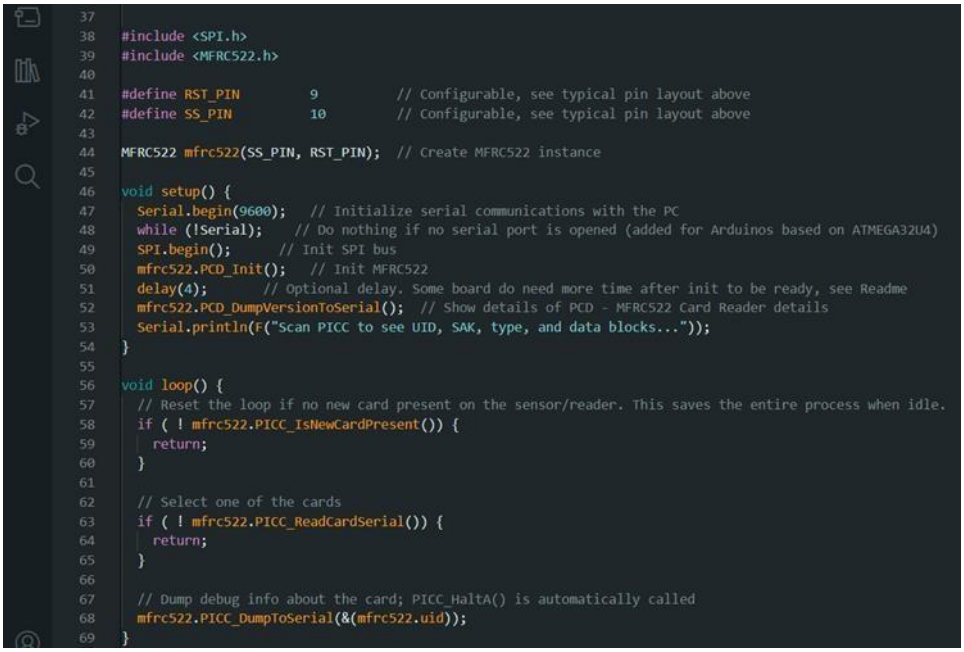
Figura 20: tags de cada modelo



Fonte: dos próprios autores, 2024.

Para que seja possível criar uma programação para algumas das tags, é preciso saber qual é a *User Identification (UID)* de cada uma delas. Para isso, foi utilizado o código DumpInfo, que mostra o UID de cada tag ao aproximá-las do leitor RFID. Esse código está disponível como exemplo na biblioteca MFRC522 do software Arduino IDE, conforme ilustrado na Figura 21.

Figura 21: código DumpInfo



```
#include <SPI.h>
#include <MFRC522.h>

#define RST_PIN         9          // Configurable, see typical pin layout above
#define SS_PIN          10         // Configurable, see typical pin layout above

MFRC522 mfrc522(SS_PIN, RST_PIN);  // Create MFRC522 instance

void setup() {
  Serial.begin(9600);   // Initialize serial communications with the PC
  while (!Serial);    // Do nothing if no serial port is opened (added for Arduinos based on ATMEGA32U4)
  SPI.begin();      // Init SPI bus
  mfrc522.PCD_Init();   // Init MFRC522
  delay(4);       // Optional delay. Some board do need more time after init to be ready, see Readme
  mfrc522.PCD_DumpVersionToSerial();  // Show details of PCD - MFRC522 Card Reader details
  Serial.println(F("Scan PICC to see UID, SAK, type, and data blocks..."));
}

void loop() {
  // Reset the loop if no new card present on the sensor/reader. This saves the entire process when idle.
  if ( ! mfrc522.PICC_IsNewCardPresent()) {
    return;
  }

  // Select one of the cards
  if ( ! mfrc522.PICC_ReadCardSerial()) {
    return;
  }

  // Dump debug info about the card; PICC_HaltA() is automatically called
  mfrc522.PICC_DumpToSerial(&(mfrc522.uid));
}
```

Fonte: dos próprios autores, 2024.

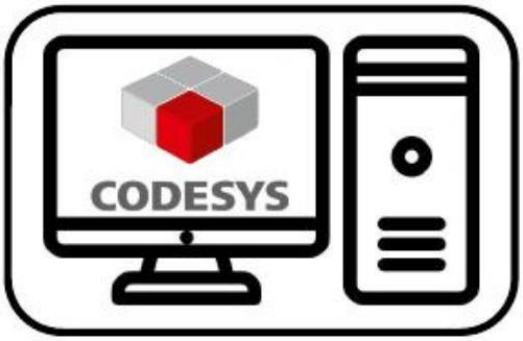
Após a identificação do UID, é possível criar uma lógica de programação que atribua a função desejada a cada tag, como será visto posteriormente neste trabalho.

4.3. SoftPLC de baixo custo

Como mencionado anteriormente neste trabalho, o SoftPLC visa transformar um computador industrial em um CLP através de software. Por questões financeiras, foi desenvolvido um SoftPLC de baixo custo, utilizando um computador pessoal em vez de um modelo industrial, e como software de runtime foi utilizada a plataforma Codesys, que permite a programação do CLP em todas as linguagens definidas na norma IEC 61131. Para melhorar o entendimento.

A Figura 22 representa a arquitetura de um SoftPLC de baixo custo.

Figura 22: SoftPLC de baixo custo

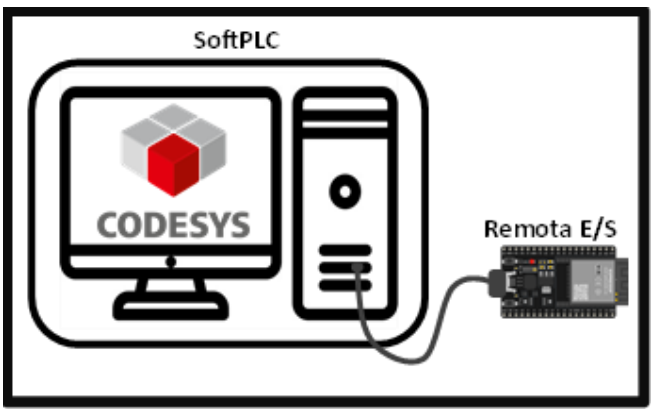


Fonte: dos próprios autores, 2024.

4.3.1. Remota (ESP32)

Como já informado no presente trabalho, o ESP32 possui uma alta capacidade de processamento em comparação com outros modelos de microcontroladores disponíveis no mercado. Esse alto poder de processamento permitiu atribuir duas funções diferentes ao microcontrolador ESP32: uma como remota de E/S do SoftPLC de baixo custo, onde, para a realização da comunicação entre eles, usou-se como meio físico um cabo USB; e outra como responsável por realizar a leitura e comparação das tags do módulo RFID, o que resultou em uma economia significativa para o projeto ao evitar o uso de outro microcontrolador. A Figura 23 demonstra uma representação de um SoftPLC de baixo custo com um ESP32 como remota de entradas e saídas.

Figura 23: SoftPLC de baixo custo com remota de E/S



Fonte: dos próprios autores, 2024.

4.4. Comunicações

Para a implementação do sistema integrado de manufatura, foram utilizados três tipos de comunicação com diferentes protocolos: SPI, Modbus RTU com meio físico USB e Modbus TCP/IP utilizando Ethernet. Cada protocolo possui uma função específica, como será detalhado a seguir.

4.4.1. Comunicação ESP32 e leitor RFID

Para realizar a comunicação entre o ESP32 e o leitor RFID, foi utilizado o protocolo de comunicação SPI. Para que os dispositivos consigam se comunicar utilizando o protocolo escolhido, primeiramente é necessário realizar a ligação do meio físico, que foi por meio de cabos conectados aos pinos correspondentes do ESP32 aos pinos MISO, MOSI, SCLK e SS do leitor. Após realizar a ligação física de ambos os dispositivos, foi preciso utilizar a plataforma Arduino IDE, que compila programas feitos em linguagem de programação C++, para elaborar o programa.

Com a finalidade de iniciar a programação responsável pela comunicação dos dispositivos utilizando o protocolo SPI, viu-se a indispensabilidade da inclusão das bibliotecas SPI e MFRC522, como representado na Figura 24.

Figura 24: bibliotecas RFID

```
#include <SPI.h>                          // Comunicação módulo RFID
#include <MFRC522.h>
```

Fonte: dos próprios autores, 2024.

Além das bibliotecas, é necessário definir no programa os pinos aos quais o RFID será conectado ao ESP32, conforme demonstrado na Figura 25.

Figura 25: definição dos pinos RFID

```
//CONFIG RFID
#define SS_PIN 5    // Pino do ESP32 conectado ao SS do RC522
#define RST_PIN 2   // Pino do ESP32 conectado ao RST do RC522
```

Fonte: dos próprios autores, 2024.

Após identificar o UID de cada tag, como demonstrado anteriormente, foram criadas quatro variáveis do tipo string: uma para armazenar o UID lido pelo RFID durante o processo e outras três para atribuir o valor do UID das tags que definem qual receita será executada. Além das variáveis do tipo string, tornou-se indispensável a criação de três variáveis do tipo bool, visto que o protocolo Modbus não envia variáveis do tipo string para a CPU. Para finalizar esta etapa, foi criada uma função void para realizar a parte lógica da leitura e comparação das tags lidas pelo módulo RFID. Na Figura 26, são demonstradas as variáveis criadas.

Figura 26: variáveis RFID

```
String tagUID = "";  // String para armazenar o UID da tag
String tag1 = "c3f5840d";  // Primeira string de UID a ser comparada
String tag2 = "d362a60d";  // Segunda string de UID a ser comparada
String tag3 = "e3f757a6";  // Terceira string de UID a ser comparada
bool RFID1, RFID2, RFID3;
void readAndCompareTag();
```

Fonte: dos próprios autores, 2024.

A Figura 27 demonstra a lógica utilizada para a leitura e comparação das tags RFID, bem como a lógica empregada para atribuir os valores das variáveis do tipo string às variáveis do tipo bool.

Figura 27: lógica de leitura e comparação do módulo RFID

```
void readAndCompareTag() {
  // Verifica se há uma nova tag presente
  if (mfrc522.PICC_IsNewCardPresent() && mfrc522.PICC_ReadCardSerial()) {
    // Limpa a string do UID antes de ler um novo
    tagUID = "";

    // Lê o UID da tag e armazena na string
    for (byte i = 0; i < mfrc522.uid.size; i++) {
      // Adiciona os bytes do UID à string (formata com 0 à esquerda se necessário)
      tagUID += (mfrc522.uid.uidByte[i] < 0x10 ? "0" : "");
      tagUID += String(mfrc522.uid.uidByte[i], HEX);
    }
   }
}
    // Compara o UID com os possíveis UIDs
    if (tagUID == tag1) {
       RFID1=1;
       RFID2=0;
       RFID3=0;

    }
    else if (tagUID == tag2) {
       RFID1=0;
       RFID2=1;
       RFID3=0;
    }
    else if (tagUID == tag3) {
       RFID1=0;
       RFID2=0;
       RFID3=1;
    }
  }
```

Fonte: dos próprios autores, 2024.

4.4.2. Comunicação remota E/S (ESP32) e CPU

Após identificar o UID de cada tag, como demonstrado anteriormente, foram criadas quatro variáveis do tipo string. Para a realização dessa fase, foi primeiramente definido o protocolo Modbus RTU para realizar a comunicação entre o ESP32 e a CPU do SoftPLC. Para isso, foi utilizado meio físico USB para conectar a remota de E/S (ESP32) à CPU, e a plataforma Arduino IDE, onde foi desenvolvida uma lógica para configurar a parte da comunicação da remota de E/S, e o CODESYS para realizar a configuração da comunicação da CPU.

Inicialmente, foi feita a programação na configuração da comunicação da remota de entradas e saídas utilizando a plataforma Arduino IDE. Deu-se como primordial inserir as bibliotecas Modbus, ModbusDevice, ModbusRegBank e ModbusSlave no programa, conforme demonstrado na Figura 28.

Figura 28: bibliotecas ModBus RTU

```
//BIBLIOTECAS
#include <modbus.h>                          // Comunicação modbus
#include <modbusDevice.h>
#include <modbusRegBank.h>
#include <modbusSlave.h>
```

Fonte: dos próprios autores, 2024.

Após a inclusão das bibliotecas, como pode ser visualizado na Figura 29, foi necessário definir o ID do escravo e determinar quais pinos seriam entradas ou saídas, associando-os aos respectivos pinos utilizados na remota.

Figura 29: configurando o ID do escravo e definindo saídas e entradas

```
void setup() {
  regBank.setId(1); //configura o ID do escravo = 1
  pinMode(DI1, INPUT_PULLUP);   //define quais pinos são (entradas ou saidas)
  pinMode(DI2, INPUT_PULLUP);
  pinMode(DI3, INPUT_PULLUP);
  pinMode(DI4, INPUT_PULLUP);
  pinMode(DO1, OUTPUT);
  pinMode(DO2, OUTPUT);
  pinMode(DO3, OUTPUT);
  pinMode(DO4, OUTPUT);
  pinMode(DO5, OUTPUT);
  pinMode(DO6, OUTPUT);
  pinMode(DO7, OUTPUT);
```

Fonte: dos próprios autores, 2024.

Como demonstrado na Figura 30, ainda no void setup, é essencial acrescentar os registros das entradas e saídas. Para este projeto, foram utilizadas apenas variáveis do tipo discreto, no entanto, caso seja necessário, é possível atribuir variáveis analógicas.

Figura 30: registros de entradas e saídas

```
//registros de entradas discretas = input status 1X
regBank.add(10001);// IDENTIFICA SENSOR DE INÍCIO
regBank.add(10002);// IDENTIFICA SENSOR DE PINTURA
regBank.add(10003);// IDENTIFICA SENSOR DE SECAGEM
regBank.add(10004);// IDENTIFICA SENSOR DE FINAL
regBank.add(10005);// IDENTIFICA A TAG RFID #1
regBank.add(10006);// IDENTIFICA A TAG RFID #2
regBank.add(10007);// IDENTIFICA A TAG RFID #3

//registros de saidas discretas = coils 0X
regBank.add(1);// IDENTIFICA MOTOR
regBank.add(2);// IDENTIFICA LED PRIMER
regBank.add(3);// IDENTIFICA LED VERMELHO
regBank.add(4);// IDENTIFICA LED VERDE
regBank.add(5);// IDENTIFICA LED AZUL
regBank.add(6);// IDENTIFICA LED VERNIZ
regBank.add(7);// IDENTIFICA LED SECAGEM
```

Fonte: dos próprios autores, 2024.

Para finalizar a parte de comunicação Modbus RTU utilizando a plataforma Arduino IDE, no void loop é crucial atribuir os endereços dos registros de entradas e saídas às suas respectivas variáveis, como representado na Figura 31. Somente assim será possível enviar e receber os dados corretamente para a CPU.

Figura 31: enviando os valores das variáveis para a CPU

```
void loop(){

 regBank.set(10001, !digitalRead(DI1));// envia ao Codesys o Status do Sensor Início
 regBank.set(10002, !digitalRead(DI2));// envia ao Codesys o Status do Sensor Pintura
 regBank.set(10003, !digitalRead(DI3));// envia ao Codesys o Status do Sensor Secagem
 regBank.set(10004, !digitalRead(DI4));// envia ao Codesys o Status do Sensor Final
 regBank.set(10005, RFID1);// envia ao Codesys o flag que identifica RfId1
 regBank.set(10006, RFID2);// envia ao Codesys o flag que identifica RfId2
 regBank.set(10007, RFID3);// envia ao Codesys o flag que identifica RfId3

 digitalWrite(DO1, regBank.get(1));//recebe do Codesys o Status do motor
 digitalWrite(DO2, regBank.get(2));//recebe do Codesys o Status do LED primer
 digitalWrite(DO3, regBank.get(3));//recebe do Codesys o Status do LED vermelho
 digitalWrite(DO4, regBank.get(4));//recebe do Codesys o Status do LED verde
 digitalWrite(DO5, regBank.get(5));//recebe do Codesys o Status do LED azul
 digitalWrite(DO6, regBank.get(6));//recebe do Codesys o Status do LED verniz
 digitalWrite(DO7, regBank.get(7));//recebe do Codesys o Status do LED secagem

    slave.run();
    delayMicroseconds (600);
    readAndCompareTag();

}
```

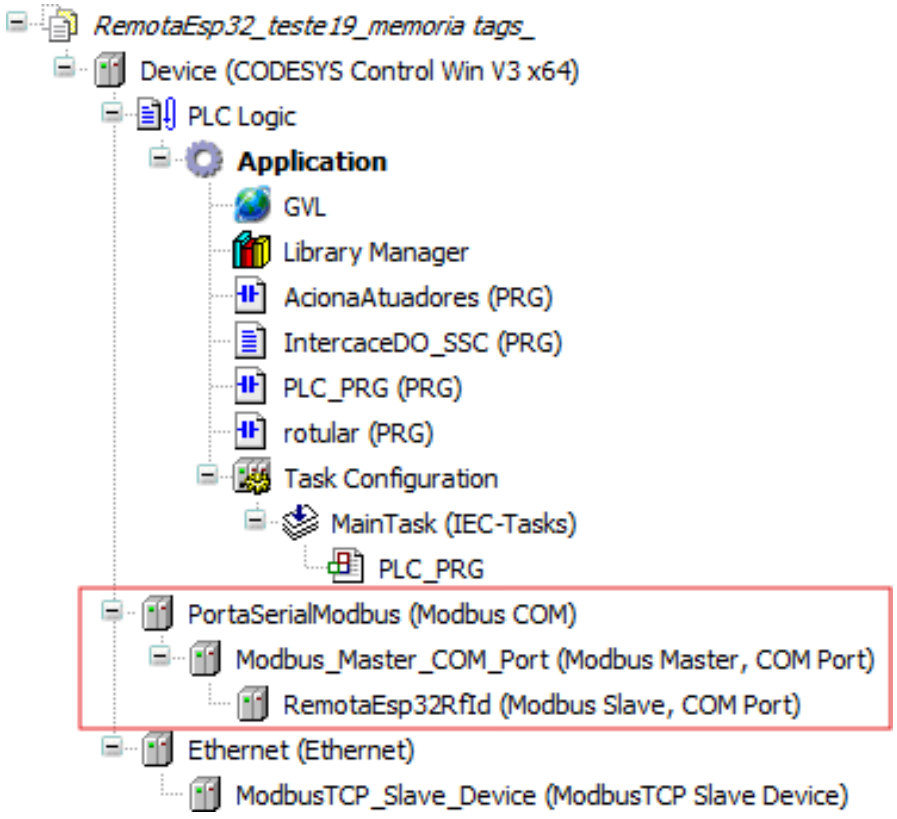
Fonte: dos próprios autores, 2024.

É possível visualizar a programação completa em C++ utilizada neste projeto no Apêndice B.

Ao finalizar o programa de comunicação da remota, é necessário configurar o programa no CODESYS para que a CPU seja capaz de se comunicar com o escravo (remota E/S). Para isso, é preciso adicionar o protocolo que será utilizado (Modbus) no programa runtime.

Ao adicionar o protocolo de comunicação Modbus, ele aparece na árvore do programa com três abas: a primeira sendo a porta serial (Modbus COM), a segunda uma sub-aba da porta serial chamada "Modbus_Master" e a terceira uma sub-aba da segunda chamada "Modbus_Slave", que foi renomeada para "RemotaESP32Rfid", conforme representado em vermelho na Figura 32.

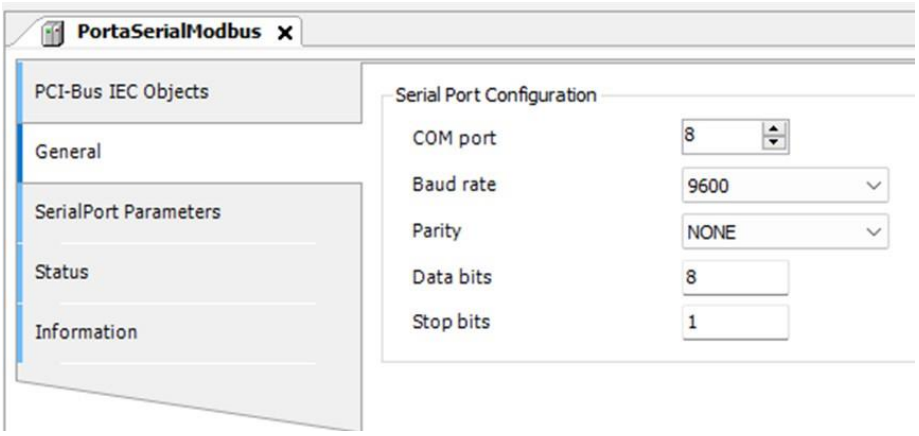
Figura 32: árvore do programa Codesys com o protocolo ModBus



Fonte: dos próprios autores, 2024.

Na aba Porta Serial, na seção geral, conforme demonstrado na Figura 33, foram configurados o baud rate e a porta COM, conforme definidos pelo computador ao executar o programa. É importante destacar que essas características precisam ser iguais às definidas no programa em C++.

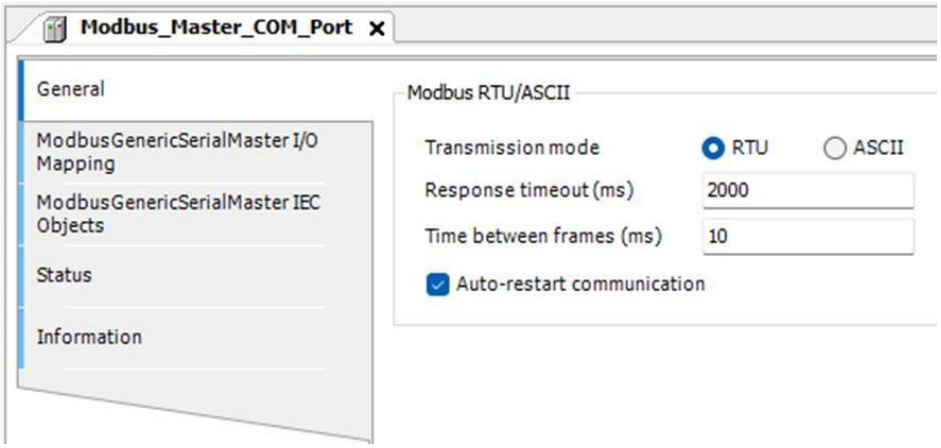
Figura 33: configuração “PortaSerialModbus”



Fonte: dos próprios autores, 2024.

Passando para a programação da aba Modbus Master, foi necessário configurar apenas a seção Geral. Neste ponto, atribui-se o tempo de resposta que melhor funciona para o projeto. É também neste momento que se configura o uso do protocolo de comunicação Modbus RTU, o qual foi escolhido para este projeto, como demonstrado na Figura 34.

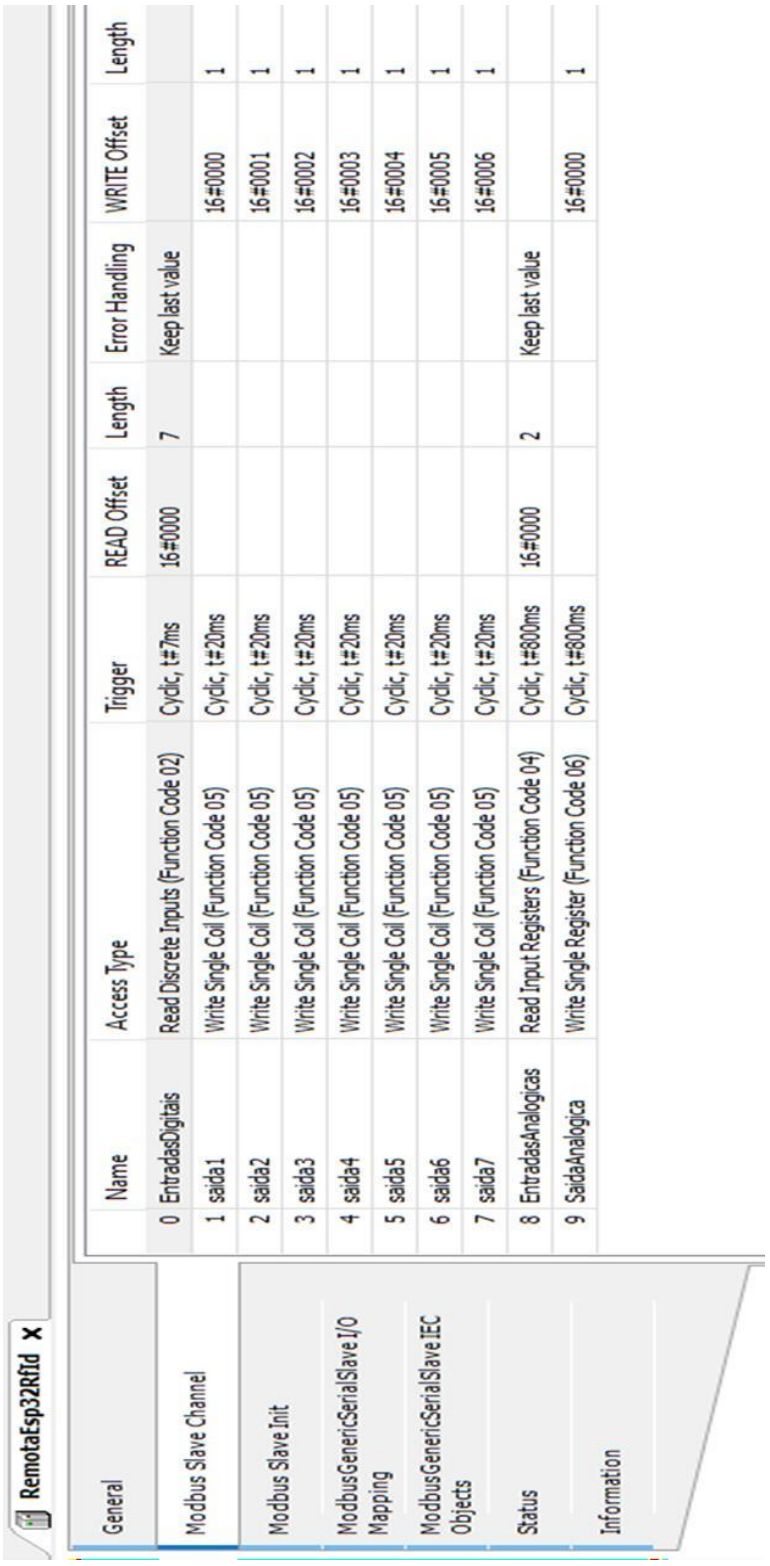
Figura 34: configuração “Modbus_Master_COM_Port”



Fonte: dos próprios autores, 2024.

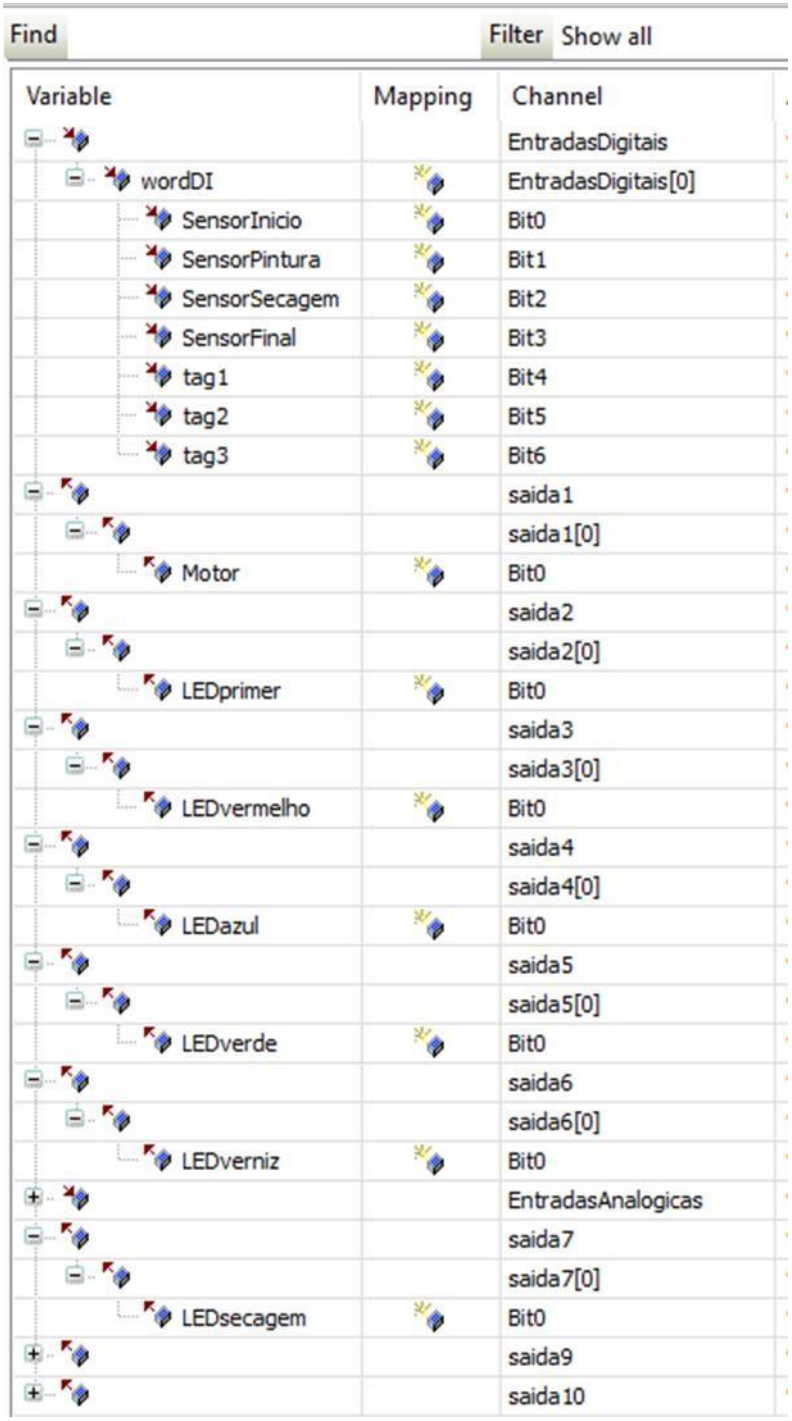
Por fim, na aba RemotaESP32RFID, responsável pela comunicação com o escravo, foi fundamental, na pasta "Modbus Slave Channel", como mostrado na Figura 35, incluir os canais nas funções compatíveis com o tipo de variável que foi utilizado, ou seja, incluir uma função "Read Discrete Inputs" com a possibilidade de atribuir sete variáveis de leitura e sete funções "Write Single Coil", possibilitando a atribuição na pasta “ModbusGenericSerialSlave I/O Mapping” das sete entradas e das sete saídas usadas no projeto (SensorInicio, SensorPintura, SensorSecagem, SensorFinal, tag1, tag2, tag3, Motor, LEDprimer, LEDvermelho, LEDazul, LEDverde, LEDverniz e LEDsecagem), conforme representado na Figura 36.

Figura 35: configuração “RemotaEsp32Rfid” na pasta “Modbus Slave Channel”



Fonte: dos próprios autores, 2024.

Figura 36: configuração “RemotaEsp32Rfid”



Fonte: dos próprios autores, 2024.

É importante ressaltar que essas variáveis são globais portanto, se forem atribuídos os mesmos nomes em alguma das abas presentes na Application, ocorrerá um conflito e o programa não será executado de forma coerente.

5. PROGRAMAÇÃO CODESYS

Após efetuar a configuração dos protocolos de comunicação, tanto da remota E/S quanto da CPU, foi feita a lógica de controle.

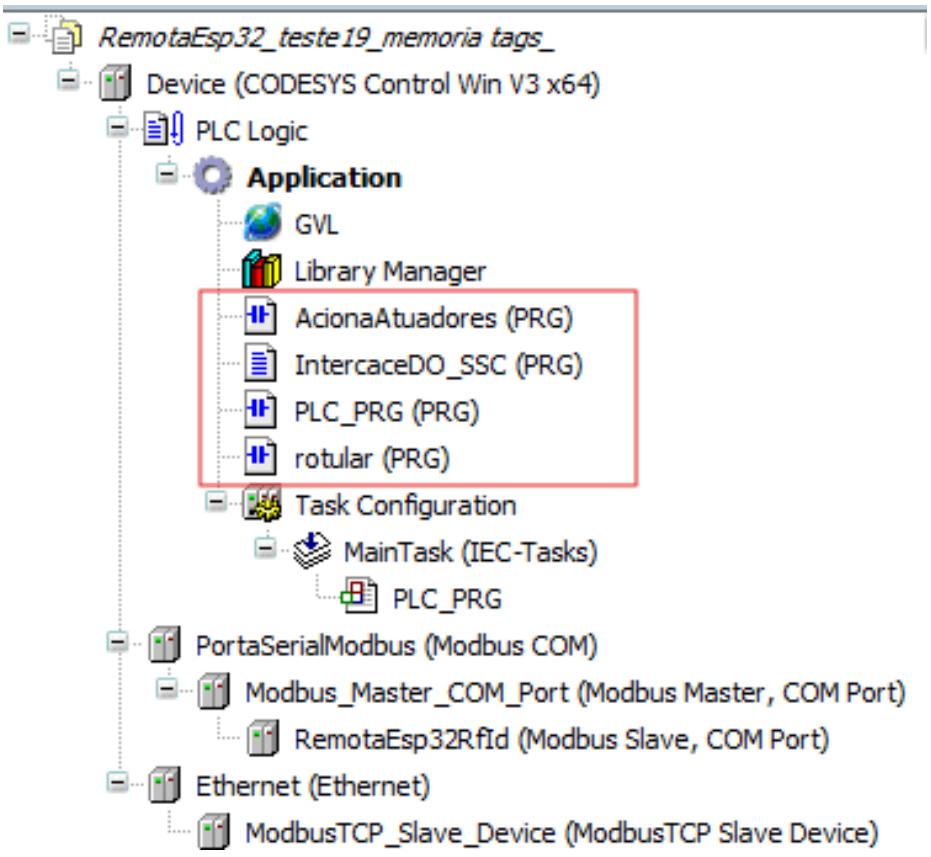
Garantindo o melhor funcionamento possível do processo, foi utilizado o método de programação conhecido como máquina de estados, que é um estilo de programação muito comum. Ele parte do princípio de armazenar o estado em que a programação se encontra, mudando apenas ao receber uma nova condição. Ao receber essa nova condição, o programa armazena sua nova posição, formando assim um ciclo que não permite que ocorra uma interferência fora de ordem, desse modo não prejudicando a execução do processo.

5.1. POU

Conforme o site da Schneider Electric (2019), uma Program Organizational Unit (POU) é um programa, um bloco funcional ou uma função usada para criar aplicações de controle. Para este trabalho, as POUs foram utilizadas com o intuito de facilitar o entendimento e as possíveis alterações do programa.

Neste projeto, foram criadas três POUs, como representado na Figura 37. A árvore do programa contém as POUs "AcionaAtuadores" e "Rotular" em linguagem de programação Ladder e a POU "InterfaceDO_SSC" em texto estruturado, além do programa principal "PLC_PRG", também em Ladder.

Figura 37: árvore do programa Codesys



Fonte: dos próprios autores, 2024.

Cada programa presente na árvore tem sua devida função, são elas:

Programa principal PLC (PLC_PRG): contém a lógica de máquina de estados para que o programa passe de um estado para outro, ou seja, define qual entrada deve ser acionada para que o programa avance para o próximo passo, além de chamar as outras POUs. O diagrama em Ladder está no Apêndice C.

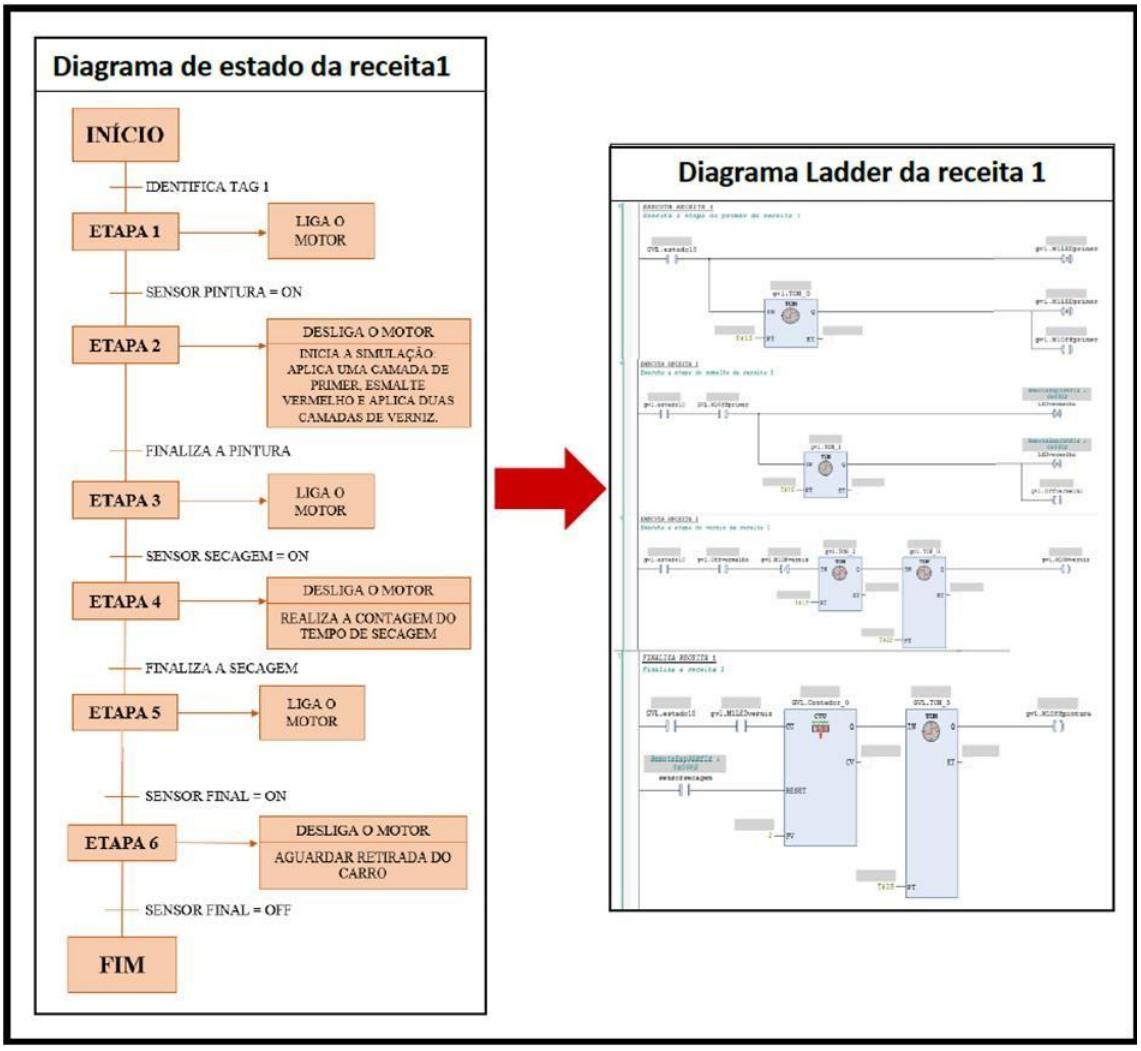
POU Rotular: tem o objetivo de atribuir um determinado valor à variável “rotularMensagens” ou à variável “TAG”, dependendo do estado em que o programa está executando no momento ou da receita determinada pelo RFID. Esses valores são enviados ao

supervisório, permitindo a exibição de diferentes mensagens que informam o estado do processo e qual receita está sendo executada. O diagrama em Ladder, para melhor visualização, está no Apêndice D.

POU AcionarAtuadores: contém a parte lógica do programa responsável por acionar cada atuador no momento adequado, de acordo com o estado do programa e a receita que está sendo executada. O diagrama da programação completa em Ladder presente nesta POU está disponível para melhor visualização no Apêndice E.

A fim de comparar o planejamento com a finalização, a Figura 38 apresenta o fluxograma da receita 1, de acordo com a metodologia, juntamente com a parte da programação em Ladder presente na "POU AcionarAtuadores", que representa a execução dessa etapa. Para melhor visualização da programação em Ladder, ela está disponível no Apêndice E.

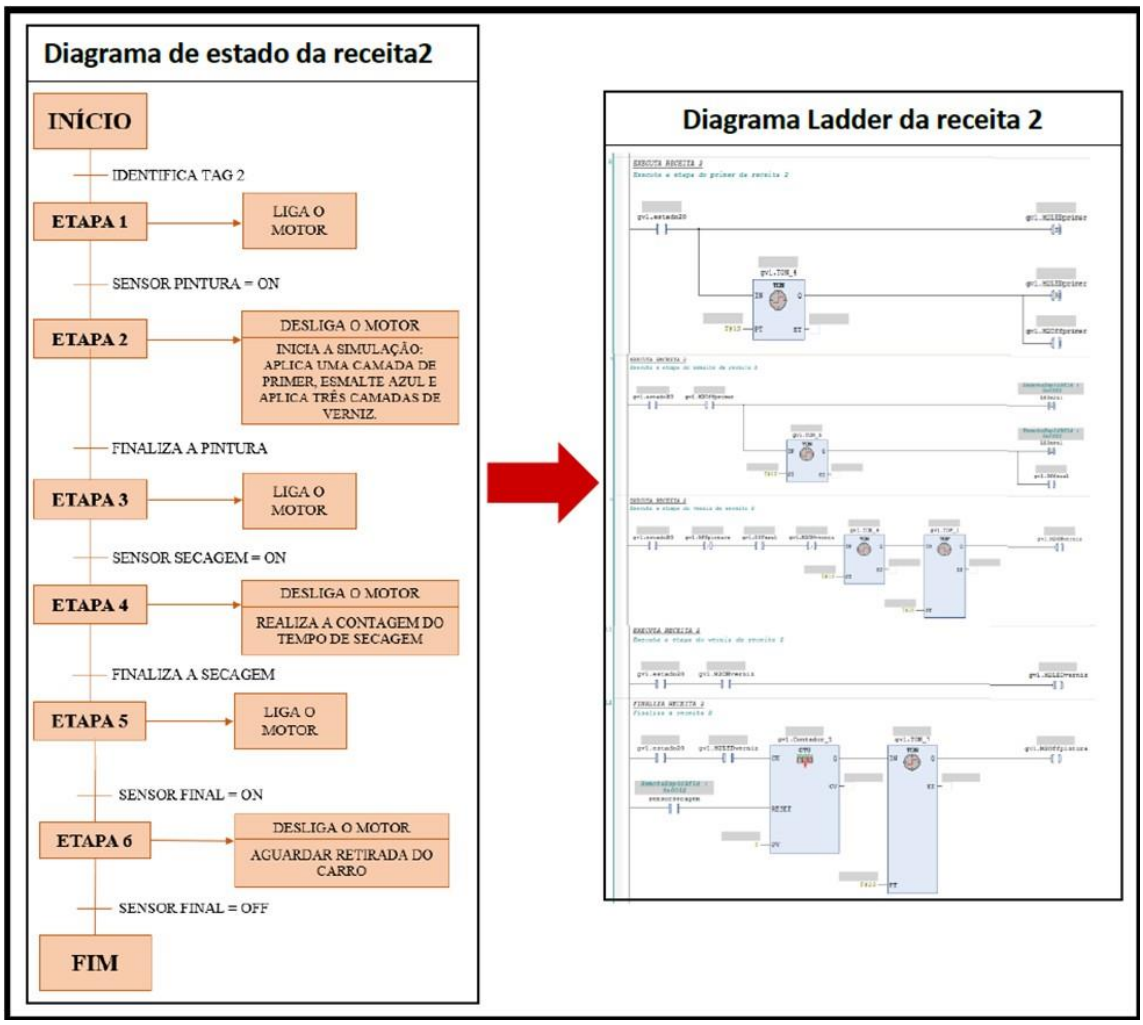
Figura 38: fluxograma e Ladder da receita 1



Fonte: dos próprios autores, 2024.

Na Figura 39, é apresentada a comparação entre o fluxograma da receita 2, conforme a metodologia e a programação em Ladder dessa etapa. Para melhor visualização da programação em Ladder, ela está disponível no Apêndice E.

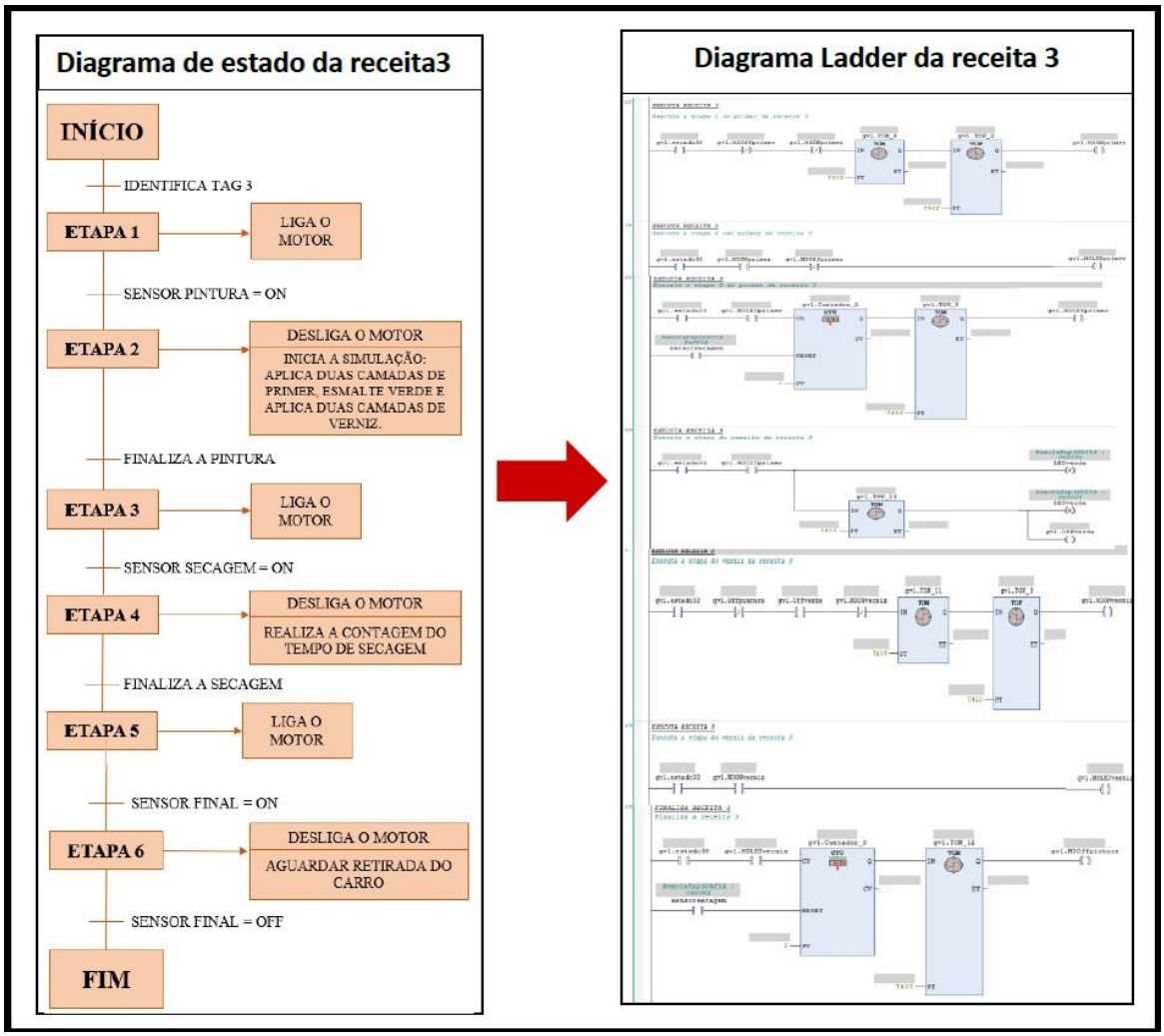
Figura 39: receita 2 (fluxograma e Ladder)



Fonte: dos próprios autores, 2024.

Na Figura 40, é apresentada a comparação entre o fluxograma da receita 3, conforme a metodologia e a programação em Ladder dessa etapa. Para melhor visualização da programação em Ladder, ela está disponível no Apêndice E.

Figura 40: receita 3 (fluxograma e Ladder)



Fonte: dos próprios autores, 2024.

POU IntercaceDO_SSC: escrita em texto estruturado, foi utilizada para agrupar todas as saídas discretas em uma lista, onde se atribui um endereço para cada uma, permitindo que o estado das saídas seja enviado ao supervisório. O diagrama em Ladder está no Apêndice F.

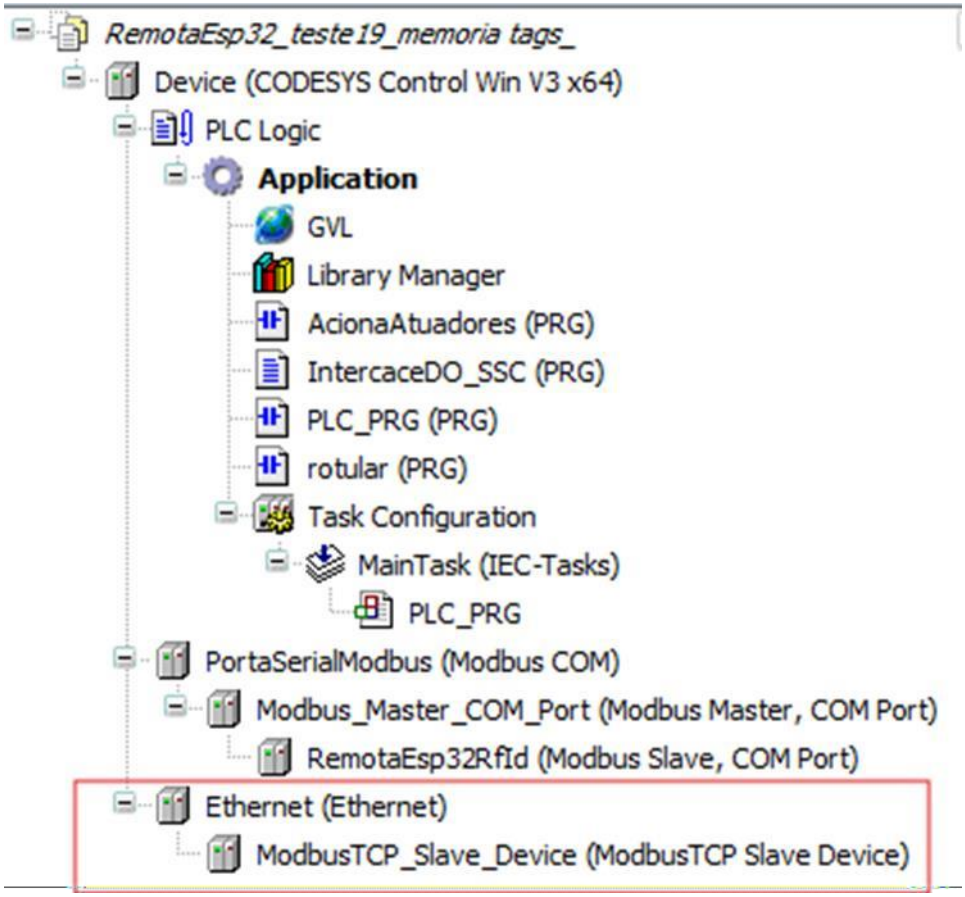
Em todos os programas foram incluídas variáveis globais. As variáveis que não estão nomeadas na aba “RemotaESP32RFID” foram nomeadas na GVL, conforme representado no Apêndice G.

É importante informar que se optou por atribuir a maioria das variáveis como globais para evitar possíveis erros de duplicação e assim, prevenir problemas na execução do programa.

5.2. Comunicação SoftPLC e SCADA

Para a comunicação entre a CPU e o sistema supervisório, foi empregado o protocolo de comunicação Modbus TCP/IP por meio físico Ethernet. Portanto, foi necessário incluir esse novo protocolo na árvore do programa, conforme demonstrado em vermelho na Figura 41.

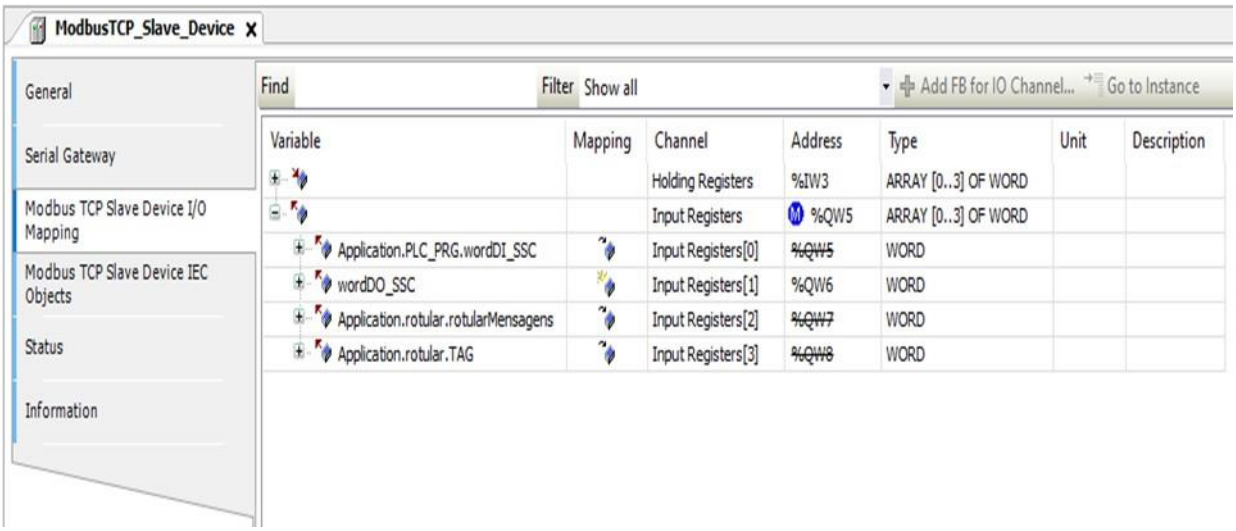
Figura 41: atribuição do protocolo Modbus TCP/IP



Fonte: dos próprios autores, 2024.

Após a inclusão do protocolo na árvore, é necessário atribuir as variáveis que serão enviadas ao supervisório. A seguir, a Figura 42 demonstra que foram incluídas as variáveis "wordDI_SSC", "wordDO_SSC", "rotularMensagens" e "Tag". Elas são responsáveis por enviar todas as entradas e saídas discretas, bem como o valor das variáveis "rotularMensagens" e "Tag", respectivamente.

Figura 42: atribuição das variáveis que serão enviadas ao supervisório

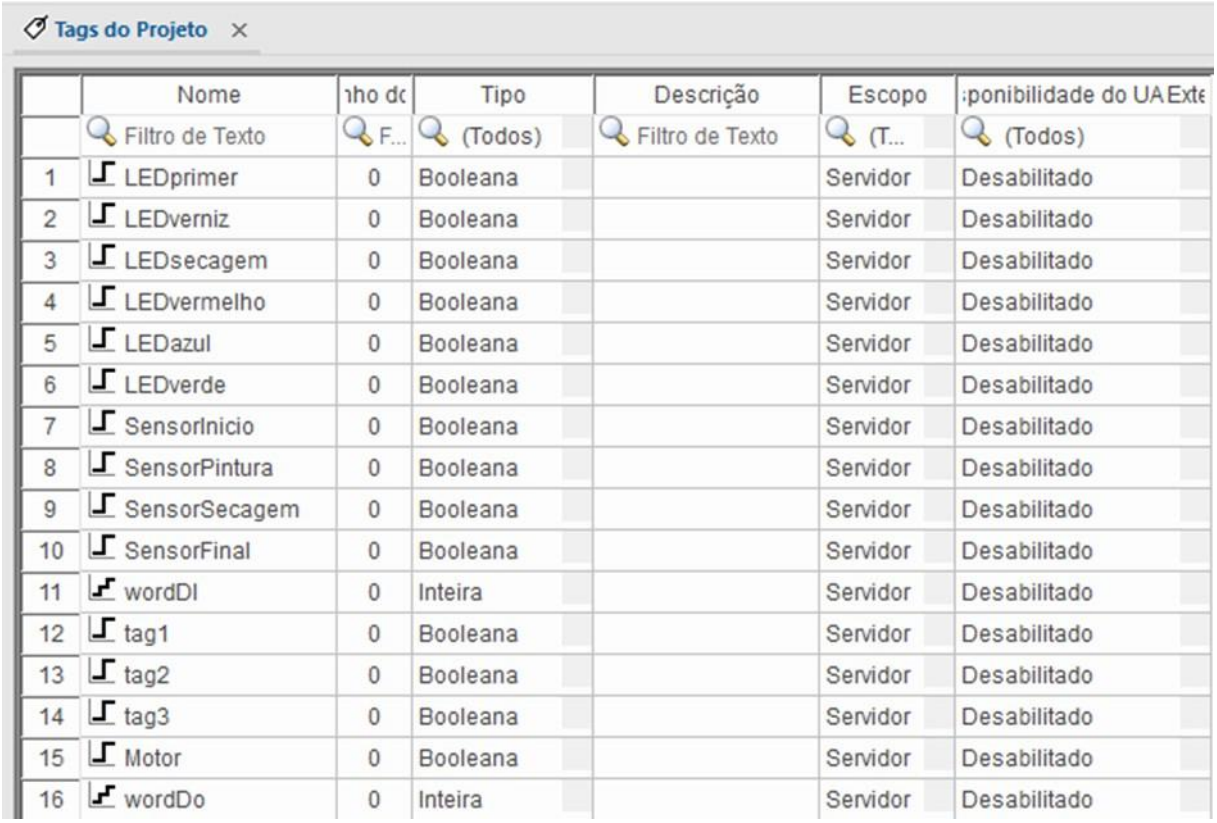


Fonte: dos próprios autores, 2024.

Para a parte referente ao sistema SCADA, foi escolhido o software AVEVA para criar o supervisório. Para realizar a comunicação entre a CPU do SoftPLC e o sistema supervisório, é necessário realizar algumas configurações preliminares. A primeira delas é criar a tabela de variáveis do supervisório na aba "Tags do Projeto".

Conforme representado na Figura 43, foram atribuídos os mesmos nomes das variáveis presentes no CODESYS, facilitando assim a identificação.

Figura 43: atribuição das variáveis do supervisório


Tags do Projeto

	Nome	nho dc	Tipo	Descrição	Escopo	ponibilidade do UA Exte
	Filtro de Texto	F...	(Todos)	Filtro de Texto	(T...	(Todos)
1	LEDprimer	0	Booleana		Servidor	Desabilitado
2	LEDverniz	0	Booleana		Servidor	Desabilitado
3	LEDsecagem	0	Booleana		Servidor	Desabilitado
4	LEDvermelho	0	Booleana		Servidor	Desabilitado
5	LEDazul	0	Booleana		Servidor	Desabilitado
6	LEDverde	0	Booleana		Servidor	Desabilitado
7	SensorInicio	0	Booleana		Servidor	Desabilitado
8	SensorPintura	0	Booleana		Servidor	Desabilitado
9	SensorSecagem	0	Booleana		Servidor	Desabilitado
10	SensorFinal	0	Booleana		Servidor	Desabilitado
11	wordDI	0	Inteira		Servidor	Desabilitado
12	tag1	0	Booleana		Servidor	Desabilitado
13	tag2	0	Booleana		Servidor	Desabilitado
14	tag3	0	Booleana		Servidor	Desabilitado
15	Motor	0	Booleana		Servidor	Desabilitado
16	wordDo	0	Inteira		Servidor	Desabilitado

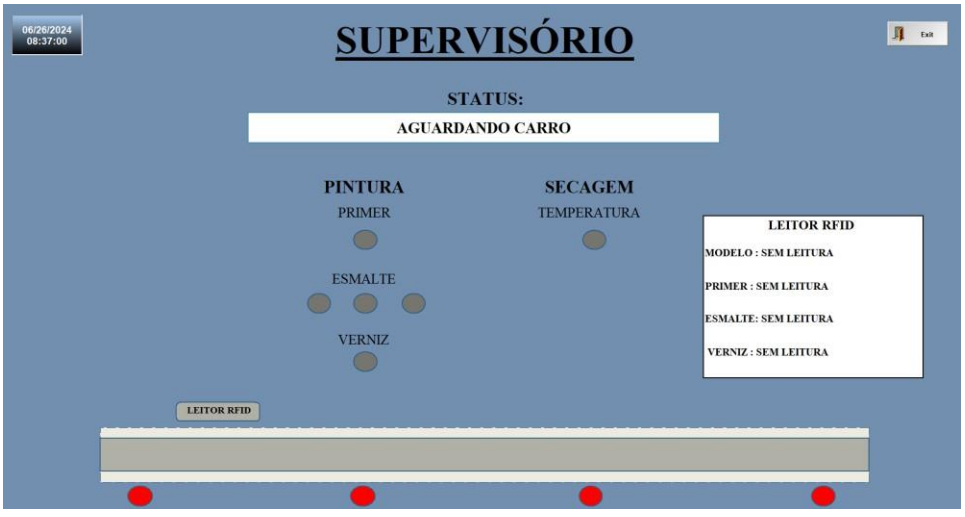
Fonte: dos próprios autores, 2024.

Após incluir as variáveis, foi preferível criar as telas do supervisório para possibilitar a realização de testes durante o restante da configuração. Portanto, como demonstrado nas Figuras 44 e 45, foram criadas duas telas para o monitoramento do processo: uma de "Boas-vindas", que corresponde a tela inicial, e outra "Tela Principal", que apresenta o status em tempo real das entradas e saídas do programa.

Figura 44: tela inicial no supervisório AVEVA

Fonte: dos próprios autores, 2024.

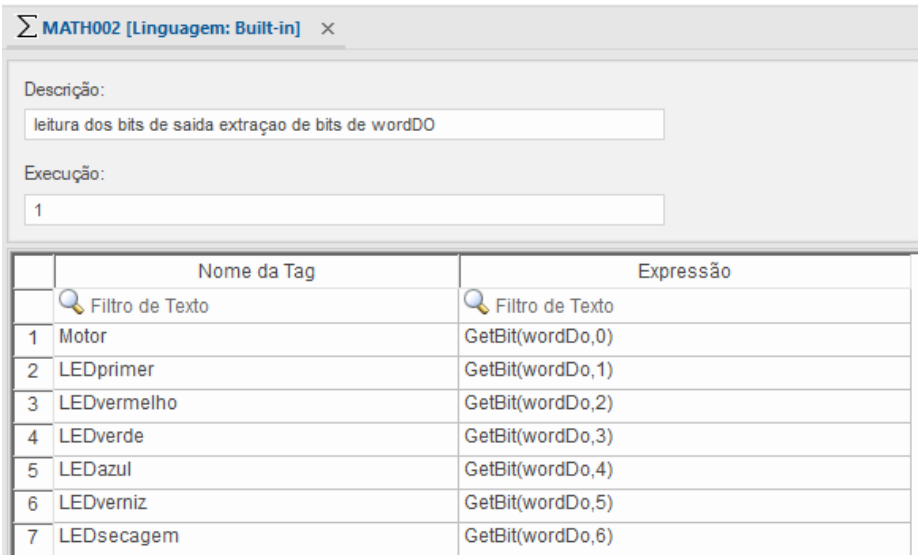
Figura 45: tela principal no supervisório AVEVA



Fonte: dos próprios autores. 2024.

Em seguida a essas etapas, iniciam-se os processos necessários para a realização da comunicação. Primeiramente, foram criadas duas tabelas matemáticas para extrair os bits das entradas e saídas discretas presentes nas variáveis "WordDO" e "WordDI" do CODESYS, atribuindo-os às respectivas variáveis do supervisório. Para isso, é necessário designar os endereços, conforme mostrado nas Figuras 46 e 47.

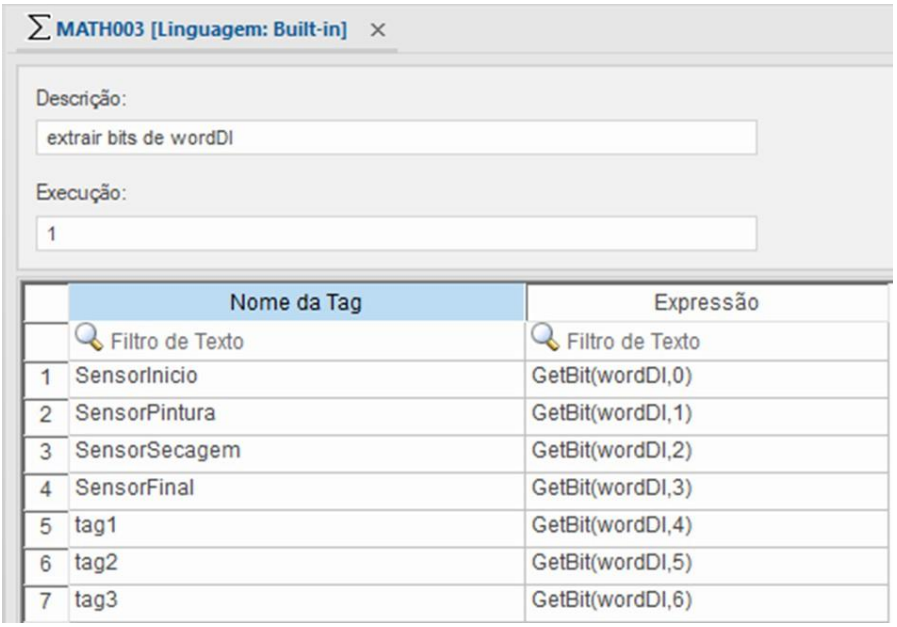
Figura 46: extraindo os bits das saídas de WordDO



	Nome da Tag	Expressão
	Filtro de Texto	Filtro de Texto
1	Motor	GetBit(wordDo,0)
2	LEDprimer	GetBit(wordDo,1)
3	LEDvermelho	GetBit(wordDo,2)
4	LEDverde	GetBit(wordDo,3)
5	LEDazul	GetBit(wordDo,4)
6	LEDverniz	GetBit(wordDo,5)
7	LEDsecagem	GetBit(wordDo,6)

Fonte: dos próprios autores, 2024.

Figura 47: extraindo os bits das entradas de WordDI

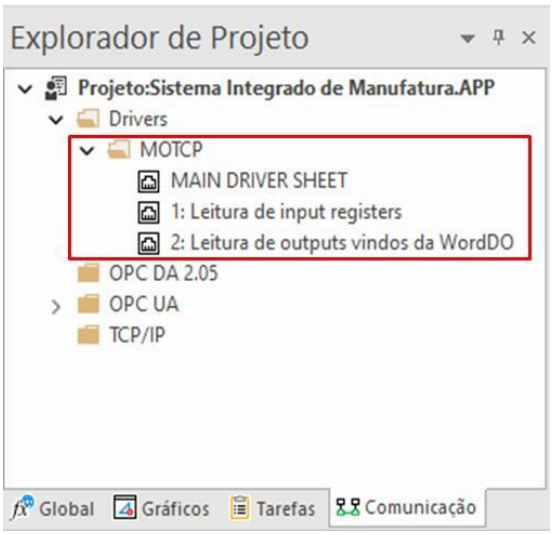


	Nome da Tag	Expressão
	Filtro de Texto	Filtro de Texto
1	SensorInicio	GetBit(wordDI,0)
2	SensorPintura	GetBit(wordDI,1)
3	SensorSecagem	GetBit(wordDI,2)
4	SensorFinal	GetBit(wordDI,3)
5	tag1	GetBit(wordDI,4)
6	tag2	GetBit(wordDI,5)
7	tag3	GetBit(wordDI,6)

Fonte: dos próprios autores, 2024.

Feito isso, o próximo passo para a realização da comunicação foi a inclusão da aba "MOTCP" em drivers. Após a inclusão, foram adicionadas duas abas, nomeadas como "Leitura de Input Registers" e "Leitura de Outputs vindos da WordDO", conforme destacado em vermelho na Figura 48.

Figura 48: comunicação MOTCP

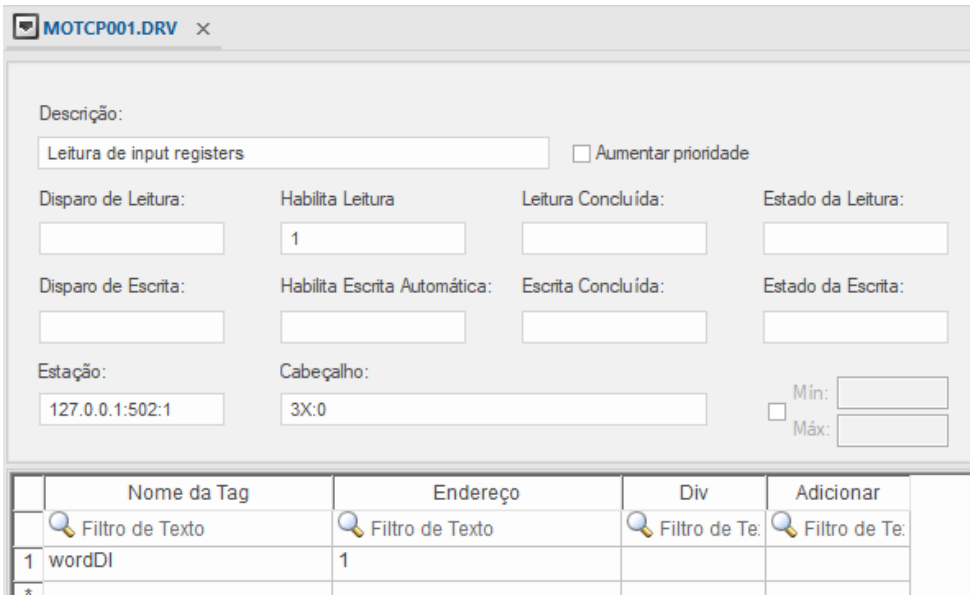


Fonte: dos próprios autores, 2024.

Na aba "Leitura de Input Registers", foi habilitada apenas a leitura, pois, neste projeto, o supervisório precisa apenas ler os estados das entradas. O campo "Estação:" foi configurado com o ID do computador e as portas utilizadas para esse processo e "3X:0" foi atribuído ao campo "Cabeçalho:", que é o número utilizado para a função determinada.

A Figura 49 demonstra esta etapa de leitura das entradas.

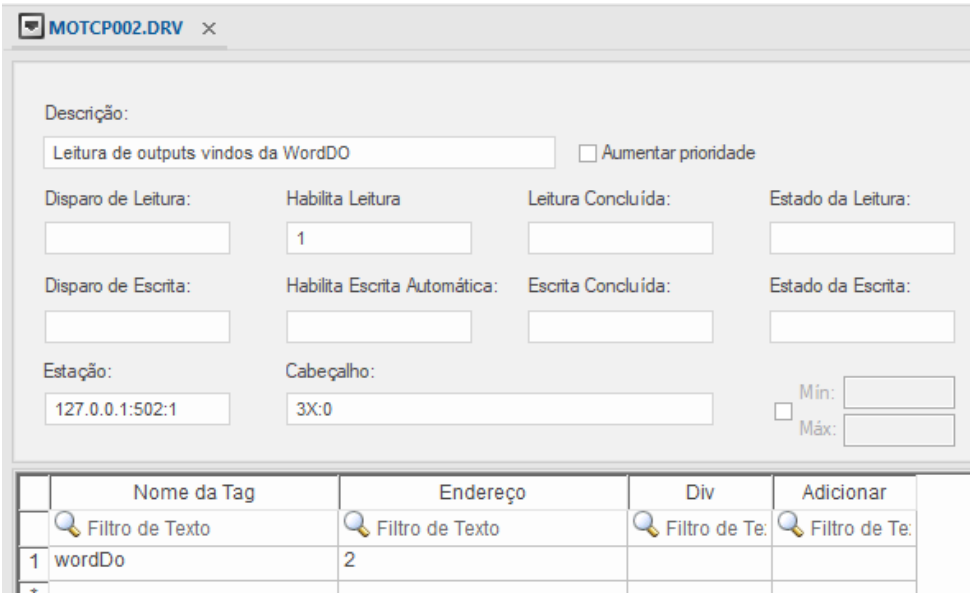
Figura 49: comunicação MOTCP “Leitura de input registers"



Fonte: dos próprios autores, 2024.

Essa configuração também se aplica para a aba "Leitura de Outputs vindos da WordDO", com a alteração apenas no valor da atribuição dos endereços das variáveis, garantindo que não sejam iguais. Como visto na Figura 50.

Figura 50: comunicação MOTCP “Leitura de outputs vindos da WordDO"



Fonte: dos próprios autores, 2024.

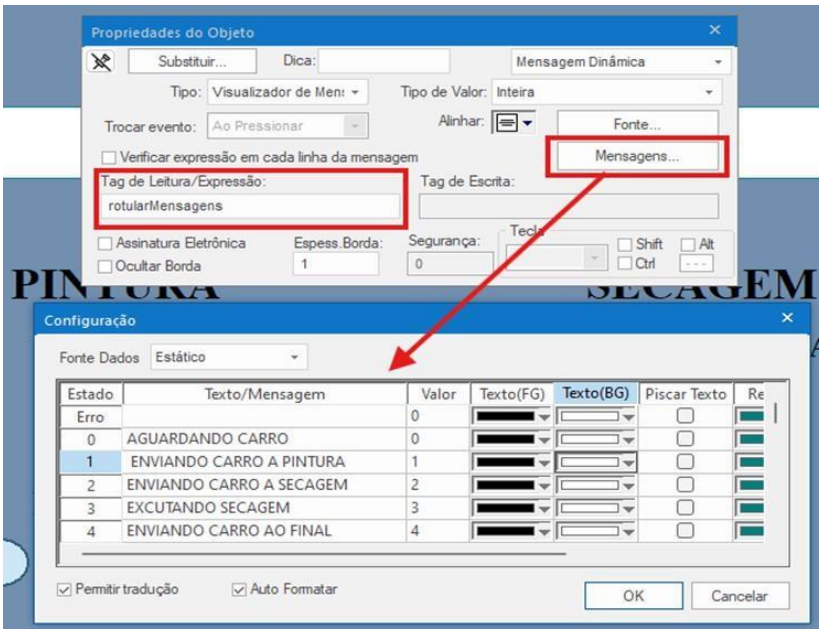
Após concluir esses passos, foi realizada a configuração dos cinco blocos de mensagens dinâmicas presentes na tela principal. A Figura 51 demonstra quais são esses blocos dinâmicos, devidamente numerados em vermelho.

Figura 51: indicação dos blocos de mensagens dinâmicas do supervisório

Fonte: dos próprios autores, 2024.

No Bloco 1, ao abrir a aba de propriedades do objeto, inicialmente foi atribuída a variável "rotularMensagens" ao bloco. Em seguida, na aba de mensagens, foi atribuído um texto específico conforme o valor da variável "rotularMensagens" na POU "rotular" presente no Codesys, conforme representado na Figura 52.

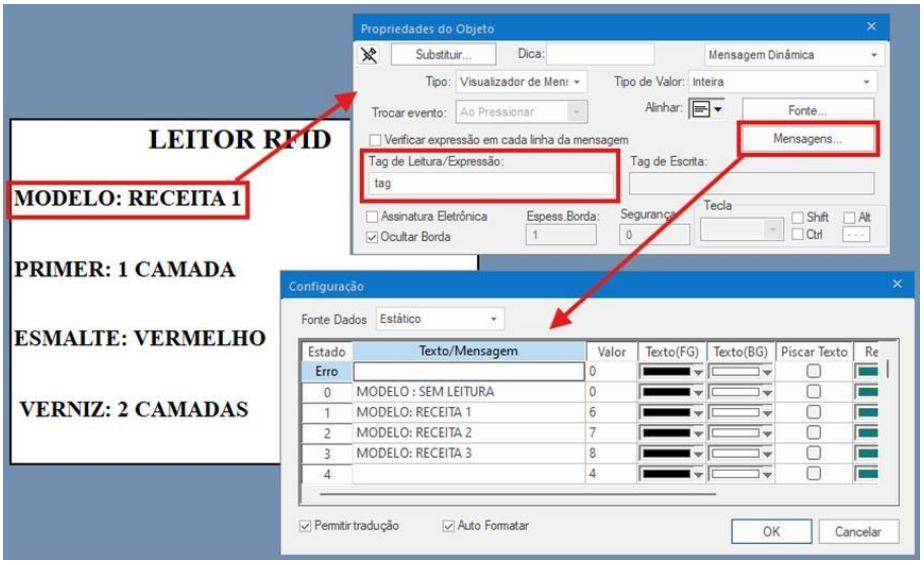
Figura 52: configuração do bloco de mensagem dinâmica 1



Fonte: dos próprios autores, 2024.

No Bloco 2, conforme demonstrado na Figura 53, foi atribuída a variável "tag". Na aba de mensagens, foram criadas diferentes mensagens de texto de acordo com o valor atribuído à variável "tag" na POU "rotular". Da mesma forma que na etapa anterior, para os blocos seguintes (3, 4 e 5), segue-se a mesma programação, alterando apenas a coluna Texto/Mensagem.

Figura 53: configuração do bloco de mensagem dinâmica 2

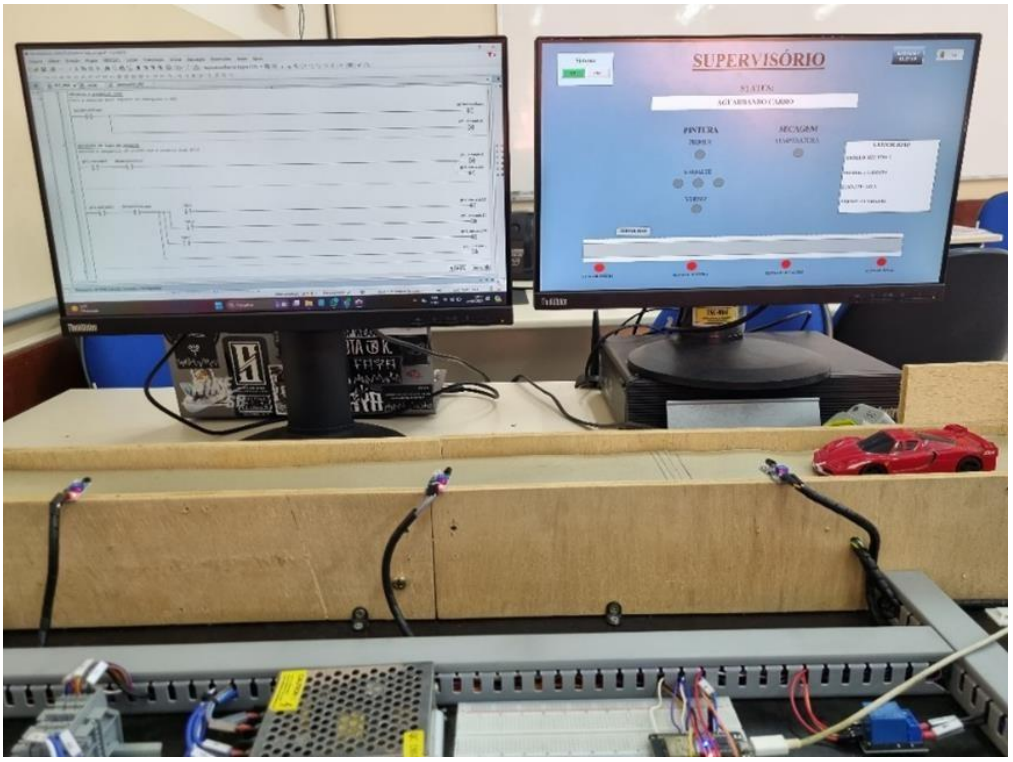


Fonte: dos próprios autores, 2024.

5.3. Finalização do desenvolvimento

Ao concluir todas as etapas descritas neste capítulo, o projeto funcionará em sua totalidade e eficácia, atingindo o objetivo final de construir um protótipo de baixo custo para um sistema integrado de manufatura. A Figura 54 demonstra o protótipo.

Figura 54: projeto finalizado



Fonte: dos próprios autores, 2024.

6. RESULTADOS E DISCUSSÕES

Com a implementação do sistema integrado de manufatura, obteve-se um protótipo funcional. Os testes realizados demonstraram a eficácia do sistema em termos de precisão dos movimentos, tempo de ciclo e definição das receitas. A análise dos resultados mostrou que o sistema é capaz de operar de forma contínua e precisa, com uma taxa de erro mínima.

7. CONSIDERAÇÕES FINAIS

O desenvolvimento deste projeto proporcionou uma experiência prática valiosa na área de automação industrial, permitindo a aplicação de conhecimentos teóricos em um sistema real. O protótipo desenvolvido demonstra a viabilidade de um sistema integrado de manufatura de baixo custo, utilizando tecnologias acessíveis a estudantes. As melhorias futuras incluem a adição de novas etapas ao processo de manufatura, como a possibilidade de incorporar mais receitas predefinidas, um controle de inspeção e a integração de diferentes tecnologias para a otimização e melhoria do processo.

REFERÊNCIAS

ALMEIDA, B. F. et al. **SoftPLC de baixo custo baseado no codesys utilizando remotas de plataformas abertas para fins didáticos**. 2023. Trabalho de conclusão de Curso (Graduação em Tecnólogo em Automação Industrial) Faculdade de Tecnologia de São Bernardo do Campo "Adib Moises Dib", São Bernardo do Campo, 2023.

ALTUS. **CONHEÇA OS DEZ PILARES DA INDUSTRIA 4.0**. 2021. Disponível em: https://www.altus.com.br/post/212/conheca-os-nove-pilares-da-industria-4-0-e-sua-relevancia-para-a-atividade-industrial. Acesso em: 20 jun de 2024.

ALTUS. **CONHECENDO OS PROTOCOLOS MODBUS RTU, PROFIBUS E CANOPEN**. 2021. Disponível em: https://www.altus.com.br/post/413/conhecendo-os-protocolos-modbus-rtu-2c-profibus-e-canopen. Acesso em: 13 jul de 2024.

ALTUS. **Curso de introdução à automação [AULA 01].** 2024. Disponível em: https://www.altus.com.br/post/100/curso-de-introducao-a-automacao--5baula-01-5d. Acesso em: 25 jun de 2024.

ALTUS. **Curso de introdução à automação [AULA 03]**: A norma IEC 61131-3. 2024. Disponível em: https://www.altus.com.br/post/104/curso-de-introducao-a-automacao--5baula-035d#:~:text=A%20Norma%20IEC%2061131%2D3,dentro%20do%20IEC%20TC65%20SC65B. Acesso em: 11 jul de 2024.

BRYAN, L.A. **Programmable controllers**: Theory and implementation. 2 ed. Georgia: An Industrial Text Company Publication, 1997. ISBN 0-944107-32-X.

COELHO, Marcelo S. **Sistemas Supervisórios**. Apostila para disciplina de sistemas supervisório moderno. São Paulo, IFSP, 2010.

ERPFLEX. **Ordem de Produção: o que é e como fazer**. 2021. Disponível em: https://www.erpflex.com.br/ordem-de-producao/. Acesso em: 03 jul de 2024.

ESPRESSIF. **ESP32**. 2024. Disponível em: https://www.espressif.com/en/products/socs/esp32. Acesso em: 12 jul de 2024.

ESPRESSIF. **ESP32 WROOM 32 DATASHEET**. 2023. Disponível em: https://www.espressif.com/sites/default/files/documentation/esp32-wroom-32_datasheet_en.pdf. Acesso: 12 jul de 2024.

GANS. **Pintura industrial: Uma abordagem Abragente Sobre Esse Processo**. 2014. Disponível em: https://www.gans.com.br/pintura-industrial. Acesso em: 03 de jul de 2024.

IOT ANALYTICS. **Soft PLCs: The industrial innovator's dilemma.** 2020. Disponível em: https://iot-analytics.com/soft-plc-industrial-innovators-dilemma/. Acesso em: 11 jul de 2024.

KALATEC AUTOMAÇÃO. **MOTOR DE CORRENTE CONTÍNUA: O QUE É, COMO FUNCIONA E VANTAGENS**. 2024. Disponível em: https://blog.kalatec.com.br/motor-corrente-continua/. Acesso em: 12 jul de 2024.

MODBUS. **Modbus Application Protocol Specification V1.1b3.** 2012. Disponível em: https://www.modbus.org/docs/Modbus_Application_Protocol_V1_1b3.pdf. Acesso em: 13 jul de 2024.

PASQUINI, Nilton Cesar. Revoluções Industriais: uma abordagem conceitual. **Revista Tecnológica da Fatec Americana**, São Paulo, v.8, n.01, p. 29-44, ago. 2020. Disponível em: https://www.fatec.edu.br/revista/index.php/RTecFatecAM/article/view/235. Acesso em: 20 jun de 2024.

PAULA, Marco Antonio Busetti de; SANTO, Eduardo Alves Portela. Uma abordagem metodológica para o desenvolvimento de sistemas automatizados e integrados de manufatura. **Revista SciELO**, São Paulo, maio 2008. Disponível em: https://www.scielo.br/j/prod/a/ZSj5w7p3MKpwPncwKCg5vnb/?format=html. Acesso em: 02 jul de 2024.

PEPPERL+FUCHS. **Módulo de E/S remotas para uma modernização eficiente dos sistemas de controle**. 2023. Disponível em: https://www.pepperl- fuchs.com/brazil/pt/38785.htm. Acesso em: 14 jul de 2024.

PETRUZELLA, Frank D. **Controladores Lógicos Programáveis**. 4 ed. Porto Alegre: AMGH, 2014.

PINHEIRO, Alan Petrônio. **"Prática 8: Comunicação SPI"**. 2018. Universidade Federal de Uberlândia. Faculdade de Engenharia Elétrica. Curso de Engenharia Eletrônica e de Telecomunicações. Patos de Minas. Disponível em: https://www.alan.eng.br/grad/microprocessadores/pratica8_spi.pdf. Acesso em: 26 jun de 2024.

SCHNEIDER Eletric; **POU**. Disponível em:< https://product-help.schneider-electric.com/Machine%20Expert/V1.1/en/SoMProg/SoMProg/Program_Components/Program_Components-3.htm>. Acesso em: junho de 2024.

SERVO. **Sensor infravermelho**. 2023. Disponível em: https://servo.ind.br/sensor-infravermelho. Acesso em: 13 jul de 2024.

SILVA, Gladimir Pinto da. **PLC – Controladores Lógicos Programáveis**. 2011. Disponível em: http://www2.pelotas.ifsul.edu.br/gladimir/Apostila%20de%20PLC_Gladimir.pdf. Acesso em: 10 jul de 2024.

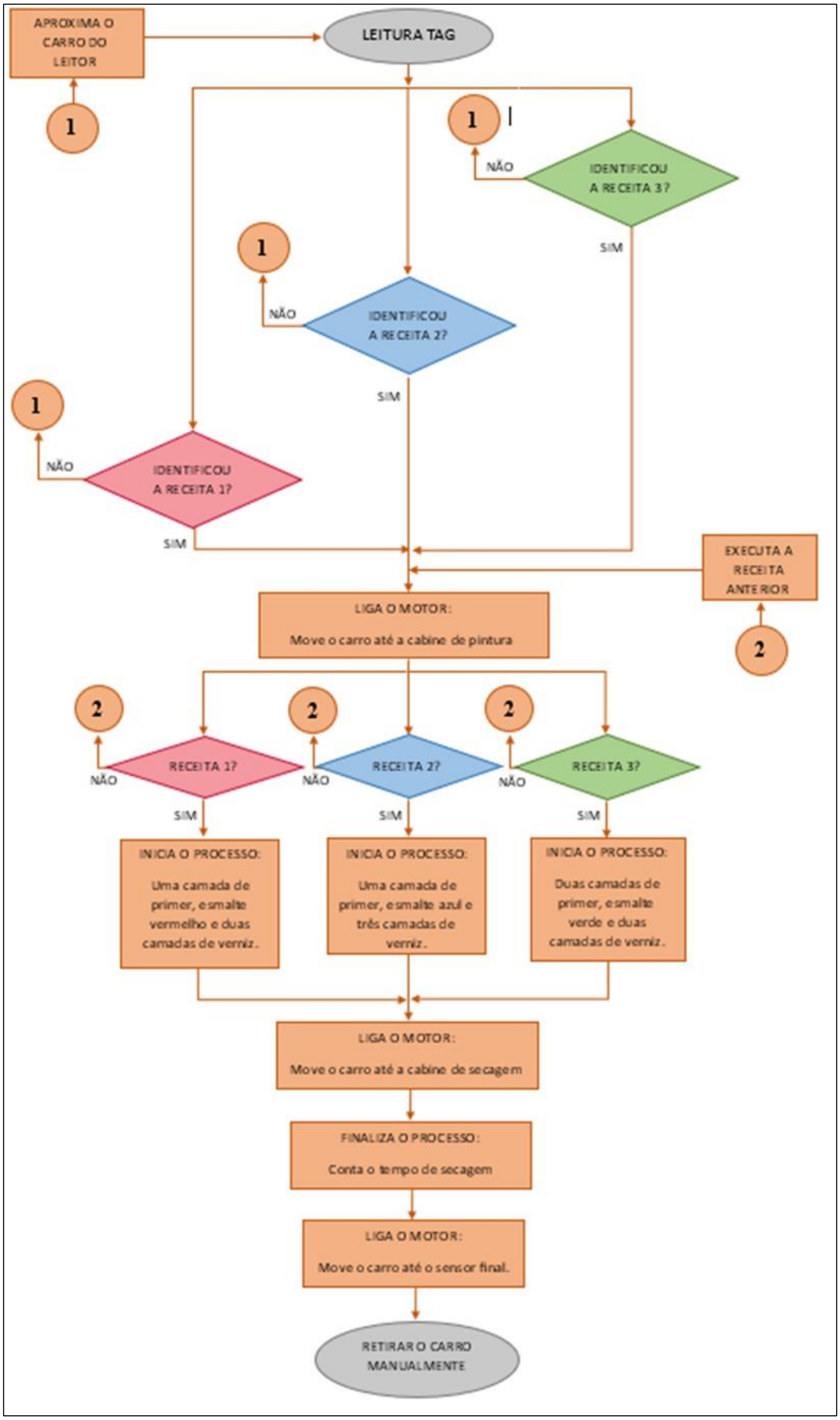
SKA. **Sistema integrado de manufatura da SKA sobe 10% da produtividade da ROMI**. 2023. Disponível em: https://www.ska.com.br/blog/sistema-integrado-de-manufatura-da-ska-sobe-10-da-produtividade-da-romi/#:~:text=Um%20sistema%20integrado%20de%20manufatura,e%20agilidade%20%C3%A0s%20demandas%20fabris.. Acesso em: 25 jun de 2024.

SZEZERBICKI, Arquimedes da Silva; PILATTI, Luiz Alberto; KOVALESKI, João Luiz. HENRY FORD: A VISÃO INOVADORA DEUM HOMEM DO INÍCIO DO SÉCULO XX. **Publication UEPG**, Ponta Grossa, v.12, n.2, p. 105-110, out. 2004. Disponível em: https://revistas.uepg.br/index.php/humanas/article/view/514/516. Acesso em:02 jul de 2024.

TOTVS. **O que é RFID, como funciona, importância, tipos, como usar e mais**. 2022. Disponível em: https://www.totvs.com/blog/gestao-industrial/rfid/. Acesso em: 13 jul de 2024.

WEG. **O que são sensores industriais e por que eles são importantes para a automação industrial?**. 2024. Disponível em: https://www.weg.net/digital/blog/o-que-sao-sensores-industriais/. Acesso em: 13 jul de 2024.

APÊNDICE A – FLUXOGRAMA DO PROGRAMA GERAL



APÊNDICE B – CÓDIGO EM C++

```
// ------------------------------------------------------------------------
--------------//

//BIBLIOTECAS

#include <modbus.h>                          // Comunicação modbus

#include <modbusDevice.h>

#include <modbusRegBank.h>

#include <modbusSlave.h>


#include <SPI.h>                             // Comunicação módulo RFID

#include <MFRC522.h>


// ------------------------------------------------------------------------
--------------//

//CONFIG RFID

#define SS_PIN 5    // Pino do ESP32 conectado ao SS do RC522

#define RST_PIN 2   // Pino do ESP32 conectado ao RST do RC522


MFRC522 mfrc522(SS_PIN, RST_PIN);  // Cria uma instância do MFRC522

// ------------------------------------------------------------------------
--------------//


modbusDevice  regBank;

modbusSlave   slave;


String tagUID = "";  // String para armazenar o UID da tag

String tag1 = "c3f5840d";  // Primeira string de UID a ser comparada

String tag2 = "d362a60d";  // Segunda string de UID a ser comparada

String tag3 = "e3f757a6";  // Terceira string de UID a ser comparada

bool RFID1, RFID2, RFID3;

void readAndCompareTag();
```

```
#define DO5   22        //saida discreta 5
#define DO6   18        //saida discreta 6
#define DO7   04        //saida discreta 7


void setup() {
  regBank.setId(1); //configura o ID do escravo = 1
  pinMode(DI1, INPUT_PULLUP);   //define quais pinos são (entradas ou
saidas)
  pinMode(DI2, INPUT_PULLUP);
  pinMode(DI3, INPUT_PULLUP);
  pinMode(DI4, INPUT_PULLUP);
  pinMode(DO1, OUTPUT);
  pinMode(DO2, OUTPUT);
  pinMode(DO3, OUTPUT);
  pinMode(DO4, OUTPUT);
  pinMode(DO5, OUTPUT);
  pinMode(DO6, OUTPUT);
  pinMode(DO7, OUTPUT);



  SPI.begin();          //inicia a comunicação SPI
  mfrc522.PCD_Init();   //inicia o RC522


  //-------------------------------------------------------------------------
-----------------------------------------------


  //registros de entradas discretas = input status 1X
  regBank.add(10001);// IDENTIFICA SENSOR DE INÍCIO
  regBank.add(10002);// IDENTIFICA SENSOR DE PINTURA
  regBank.add(10003);// IDENTIFICA SENSOR DE SECAGEM
```

```
  slave.setBaud(9600);
}

void readAndCompareTag() {
  // Verifica se há uma nova tag presente
  if (mfrc522.PICC_IsNewCardPresent() && mfrc522.PICC_ReadCardSerial()) {
    // Limpa a string do UID antes de ler um novo
    tagUID = "";

    // Lê o UID da tag e armazena na string
    for (byte i = 0; i < mfrc522.uid.size; i++) {
      // Adiciona os bytes do UID à string (formata com 0 à esquerda se
necessário)
      tagUID += (mfrc522.uid.uidByte[i] < 0x10 ? "0" : "");
      tagUID += String(mfrc522.uid.uidByte[i], HEX);
    }
   }
}
    // Compara o UID com os possíveis UIDs
    if (tagUID == tag1) {
       RFID1=1;
       RFID2=0;
       RFID3=0;

    }
    else if (tagUID == tag2) {
       RFID1=0;
       RFID2=1;
       RFID3=0;
    }
    else if (tagUID == tag3) {
```

```
digitalWrite(DO1, regBank.get(1));
digitalWrite(DO2, regBank.get(2));
digitalWrite(DO3, regBank.get(3));
digitalWrite(DO4, regBank.get(4));
digitalWrite(DO5, regBank.get(5));
digitalWrite(DO6, regBank.get(6));
digitalWrite(DO7, regBank.get(7));

   slave.run();
```

APÊNDICE C – PROGRAMA PRINCIPAL PLC (PLC_PRG)

PLC_PRG

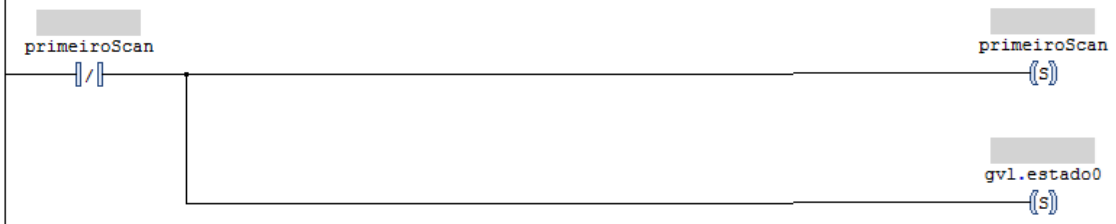
```
1  PROGRAM PLC_PRG
2  VAR
3
4      wordDI_SSC: WORD;// cópia das entradas discretas para o supervisório
5      primeiroScan: BOOL;
6
7  END_VAR
```

100 %

1

detecta o primeiro scan

leva a máquina para repouso ao energizar o PLC



2

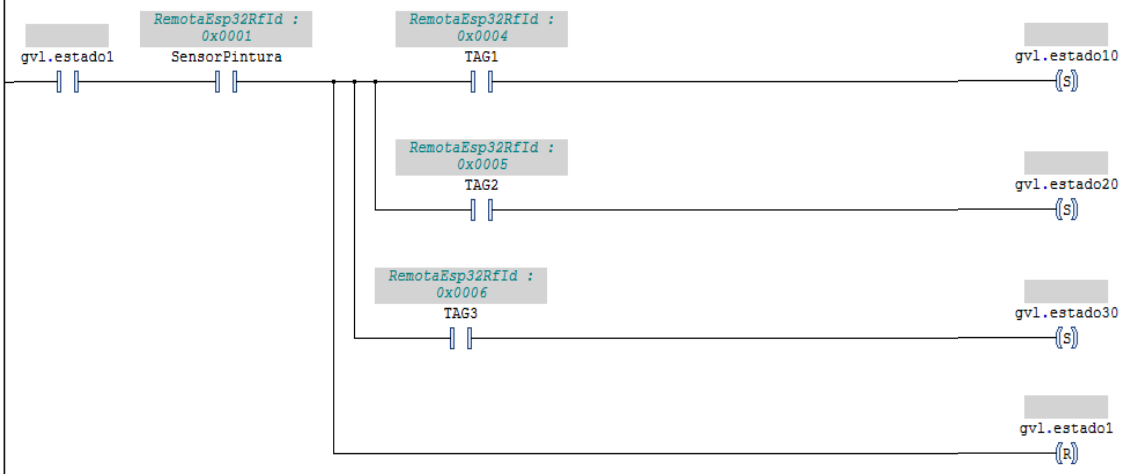
detecção do sensor de inicio

Determina se o sensor de inicio foi ativado

3

detecção do tipo de receita e do sensor de pintura

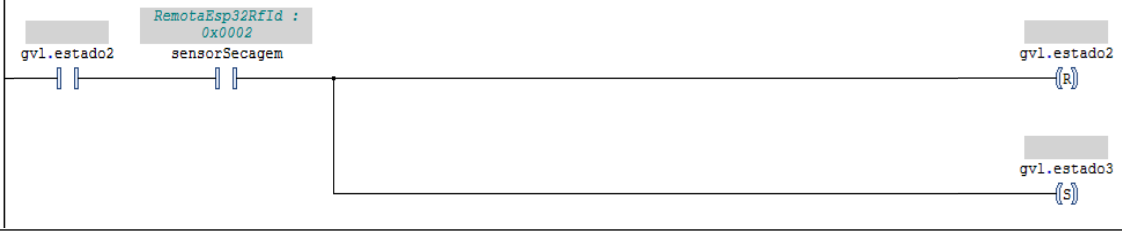
Detecta se o carro esta do sensor de pintura e escolhe e executa a sequencia de acordo com a receita lida RFID

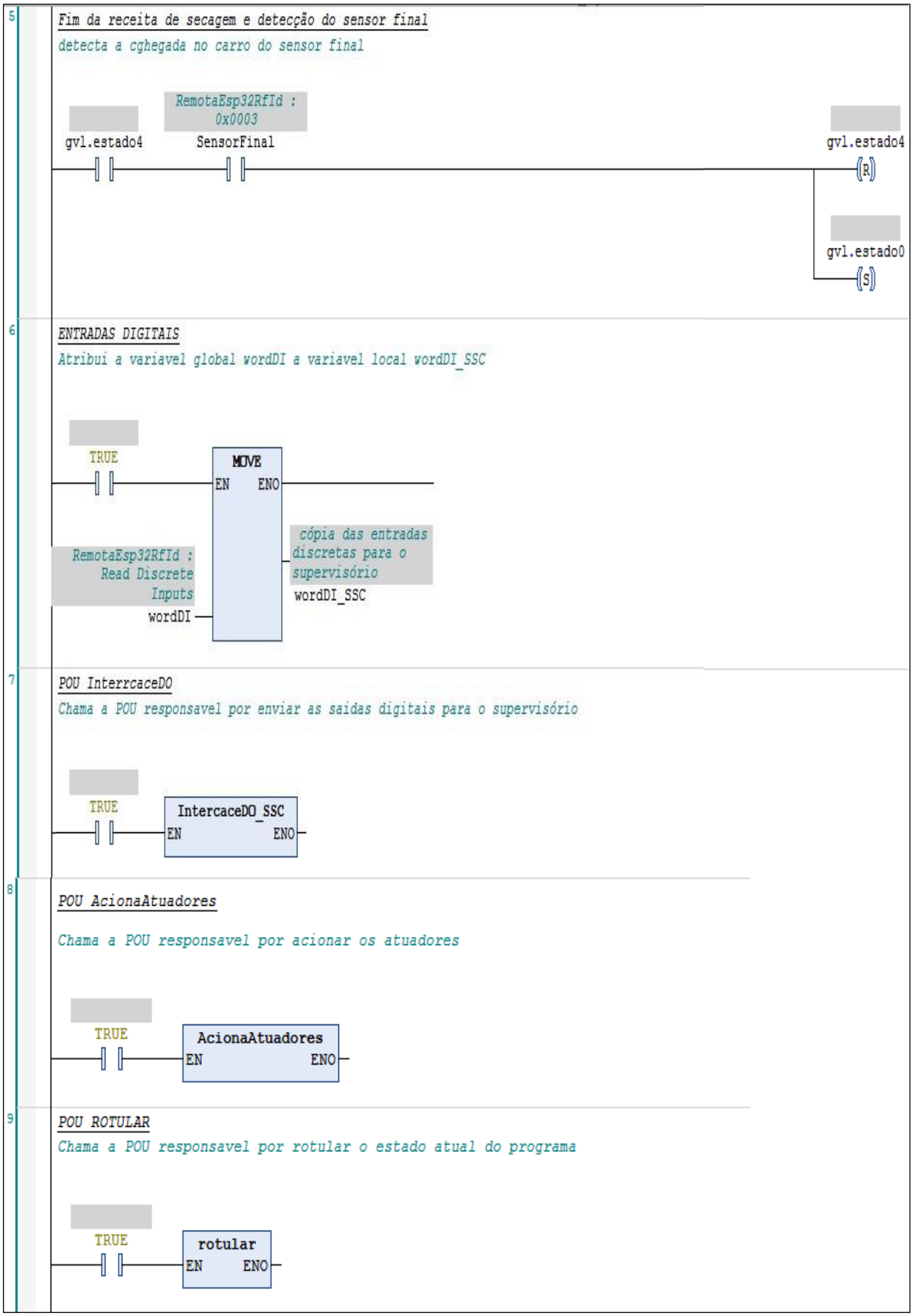


4

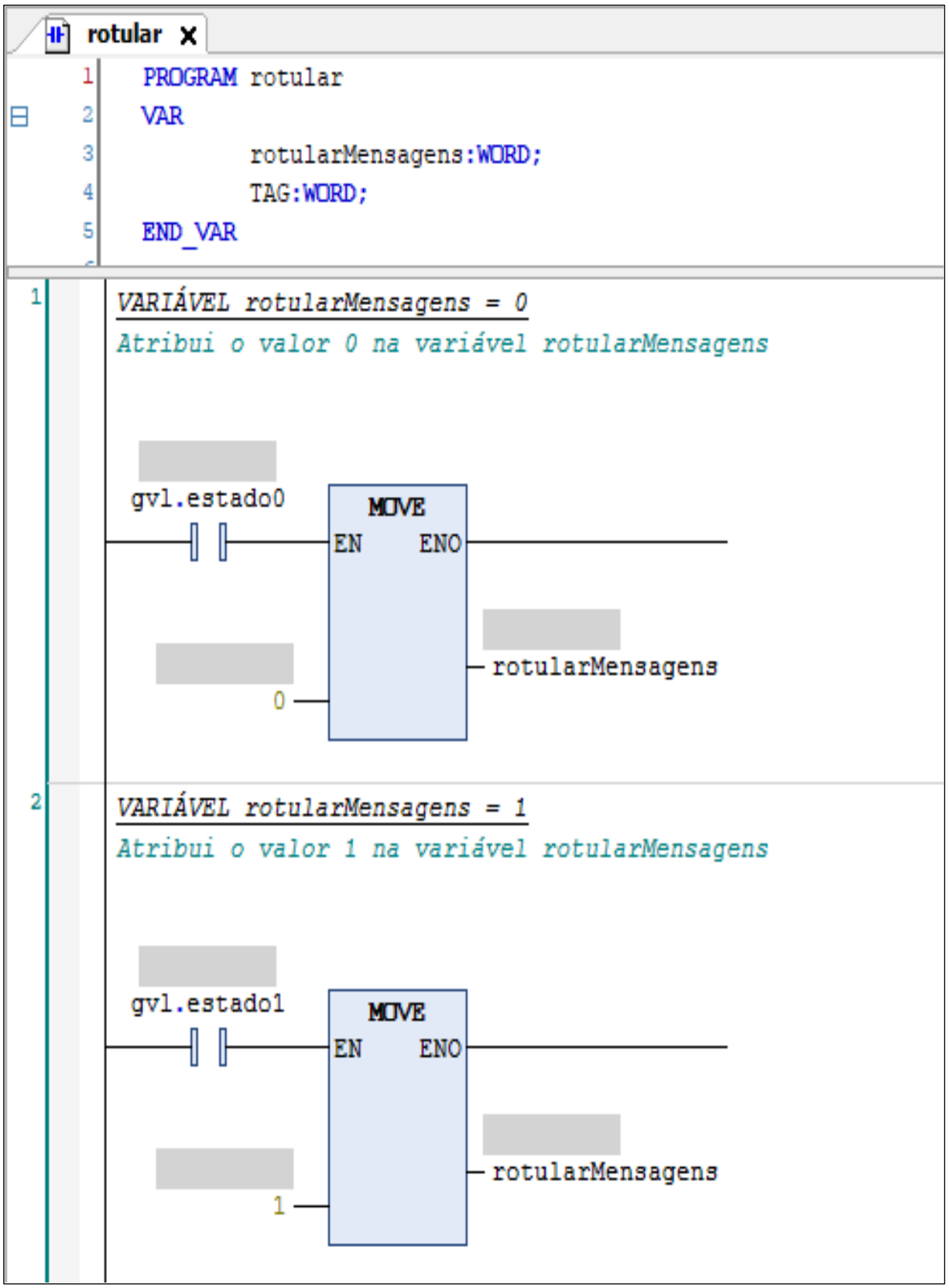
Detecção do sensor de secagem

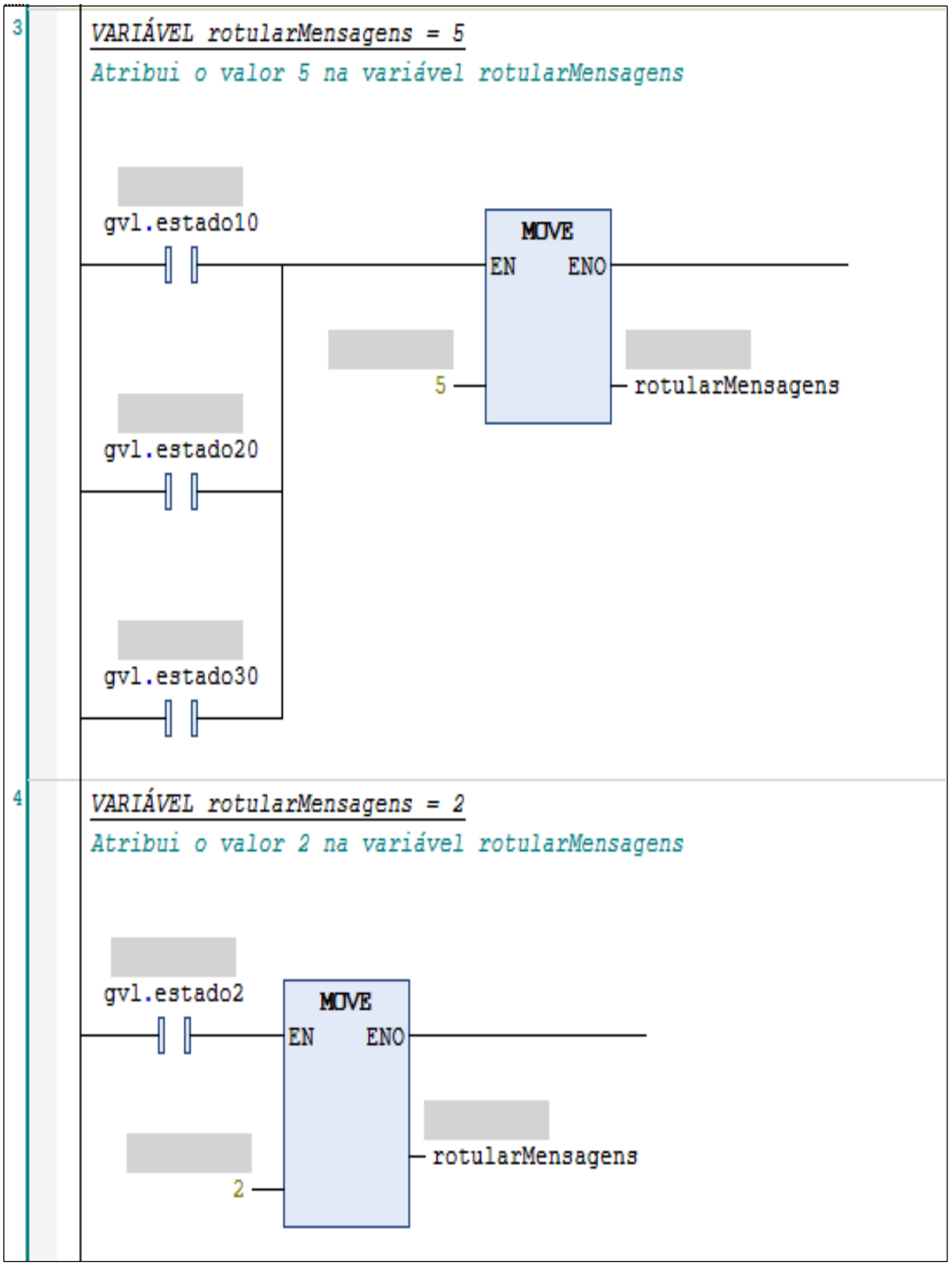
Detecta quando o carro chega do sensor de secagem

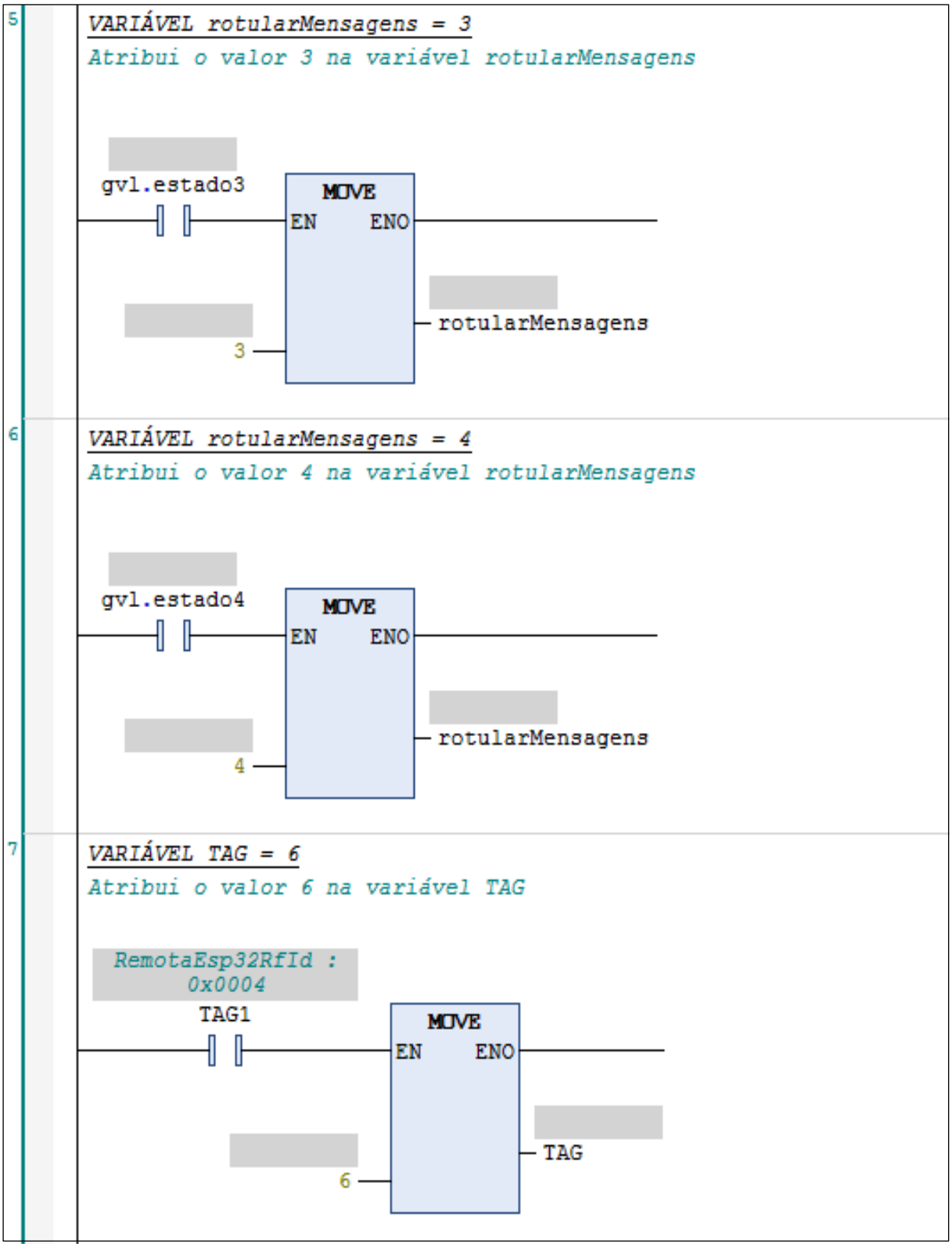



5
Fim da receita de secagem e detecção do sensor final
detecta a cghegada no carro do sensor final
gvl.estado4
RemotaEsp32RfId : 0x0003
SensorFinal
gvl.estado4
R
gvl.estado0
S
6
ENTRADAS DIGITAIS
Atribui a variavel global wordDI a variavel local wordDI_SSC
TRUE
MOVE
EN
ENO
RemotaEsp32RfId : Read Discrete Inputs
wordDI
cópia das entradas discretas para o supervisório
wordDI_SSC
7
POU InterrcaceDO
Chama a POU responsavel por enviar as saidas digitais para o supervisório
TRUE
IntercaceDO_SSC
EN
ENO
8
POU AcionaAtuadores
Chama a POU responsavel por acionar os atuadores
TRUE
AcionaAtuadores
EN
ENO
9
POU ROTULAR
Chama a POU responsavel por rotular o estado atual do programa
TRUE
rotular
EN
ENO

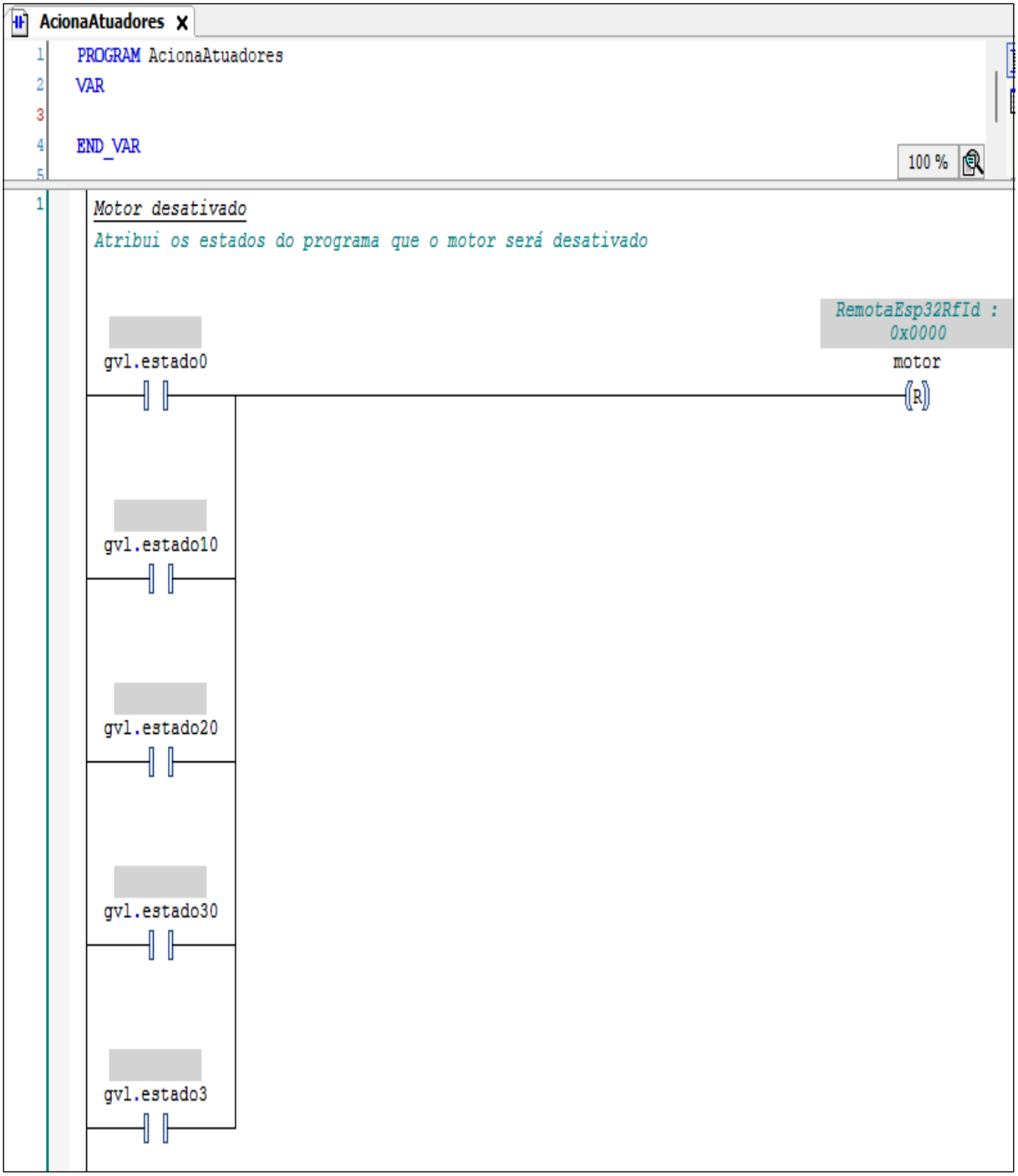
APÊNDICE D – POU “ROTULAR”

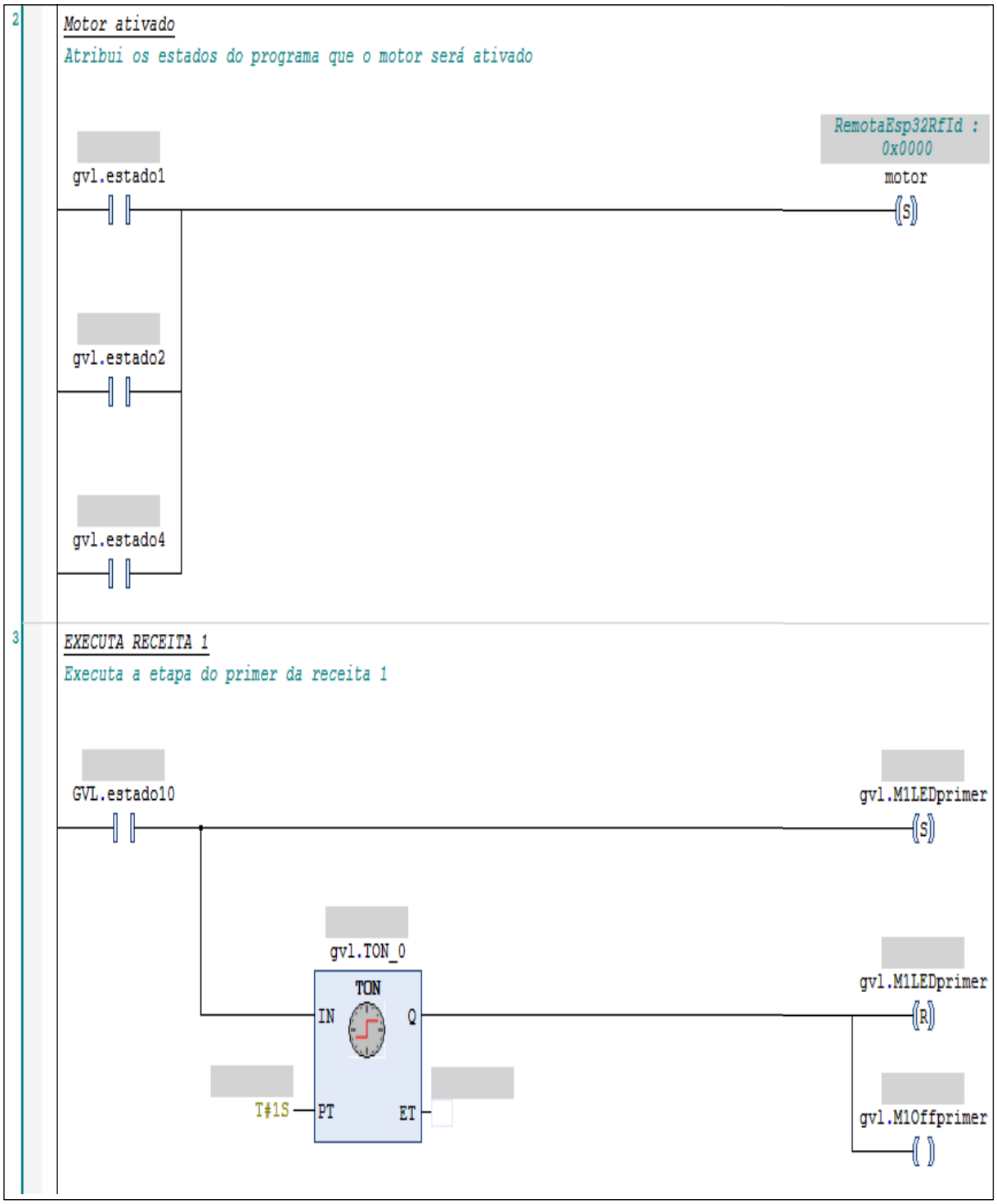


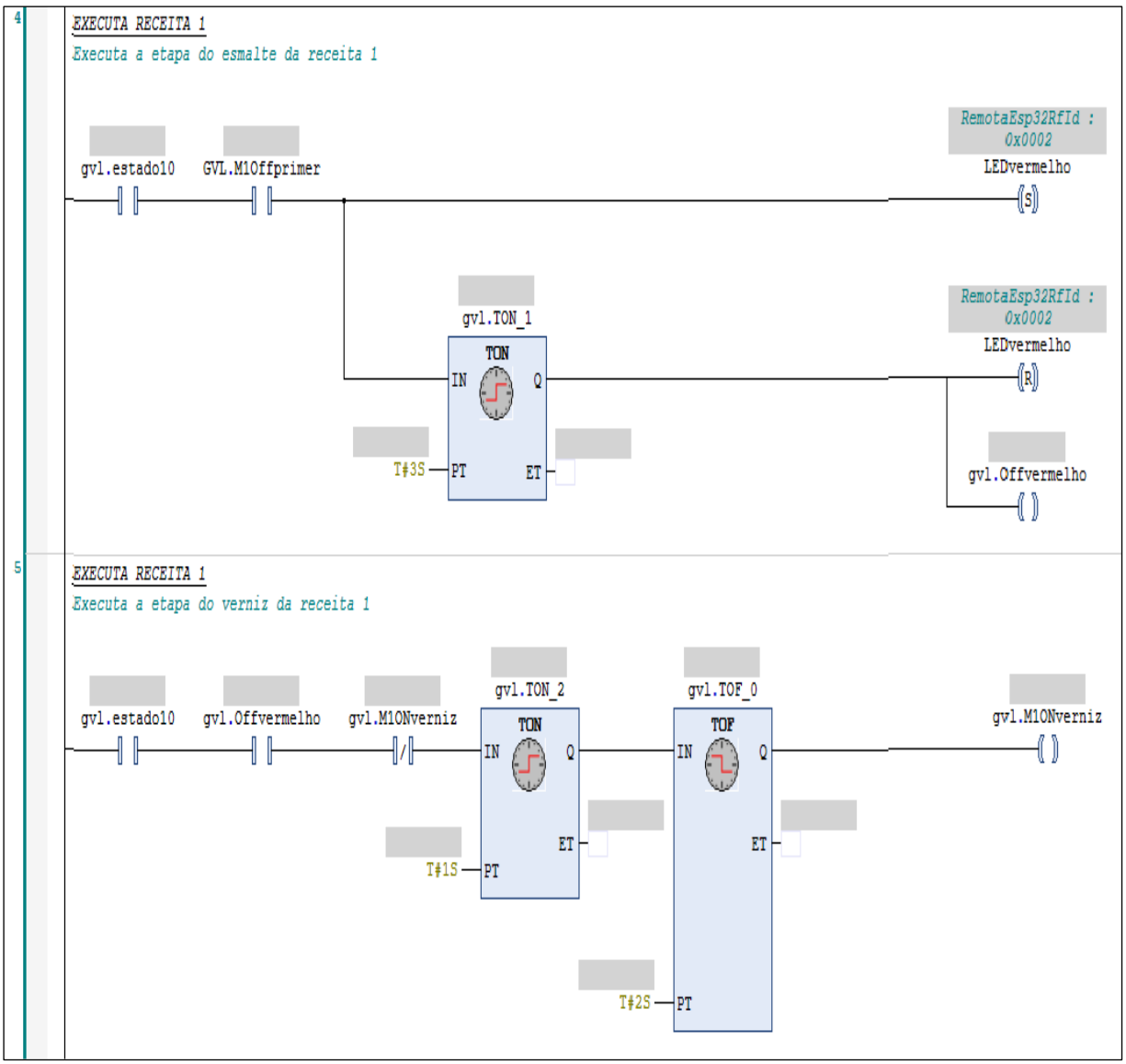

3
VARIÁVEL rotularMensagens = 5
Atribui o valor 5 na variável rotularMensagens
gvl.estado10
MOVE
EN
ENO
5
rotularMensagens
gvl.estado20
gvl.estado30
4
VARIÁVEL rotularMensagens = 2
Atribui o valor 2 na variável rotularMensagens
gvl.estado2
MOVE
EN
ENO
rotularMensagens
2

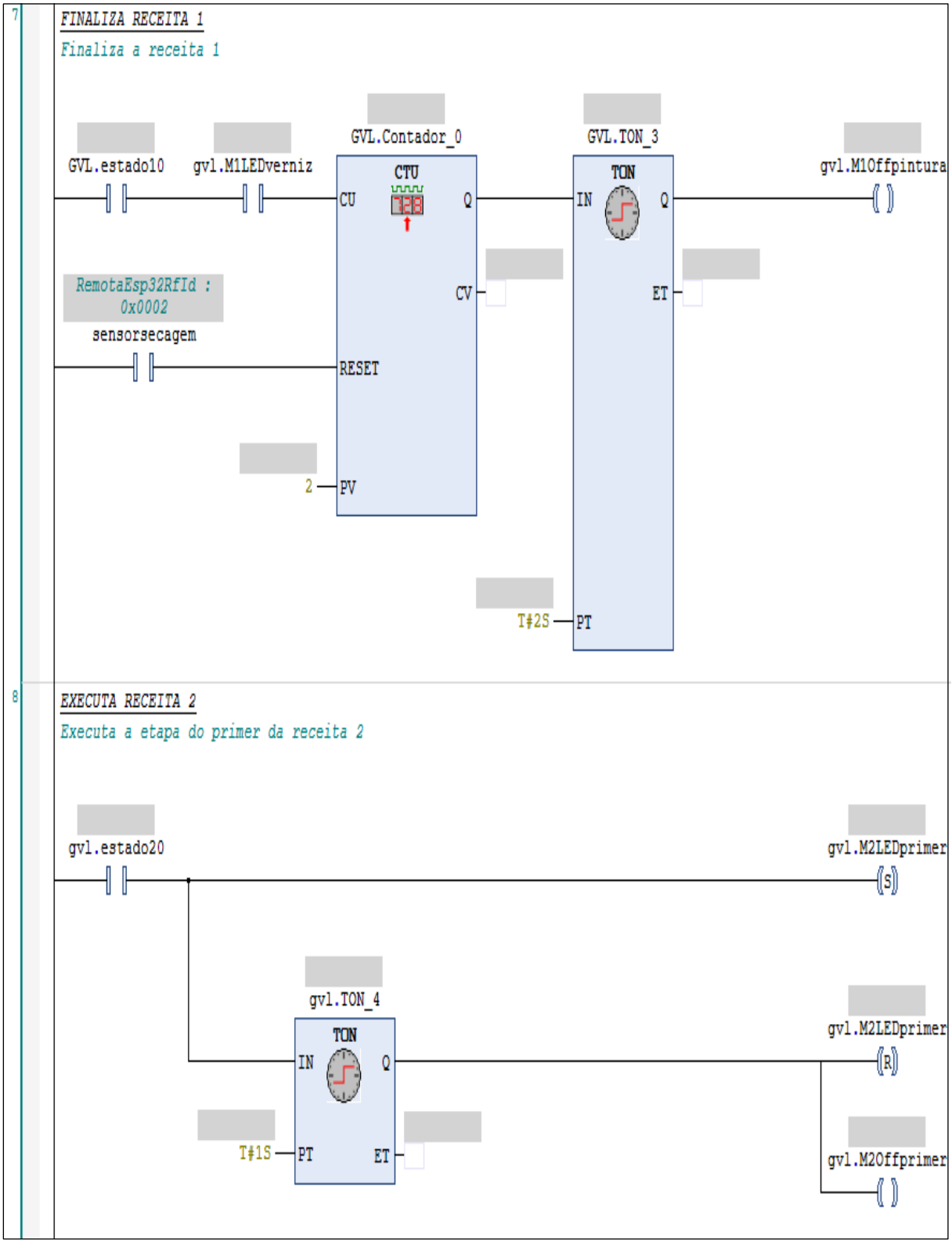

5
VARIÁVEL rotularMensagens = 3
Atribui o valor 3 na variável rotularMensagens
gvl.estado3
MOVE
EN
ENO
3
rotularMensagens
6
VARIÁVEL rotularMensagens = 4
Atribui o valor 4 na variável rotularMensagens
gvl.estado4
MOVE
EN
ENO
4
rotularMensagens
7
VARIÁVEL TAG = 6
Atribui o valor 6 na variável TAG
RemotaEsp32RfId : 0x0004
TAG1
MOVE
EN
ENO
6
TAG

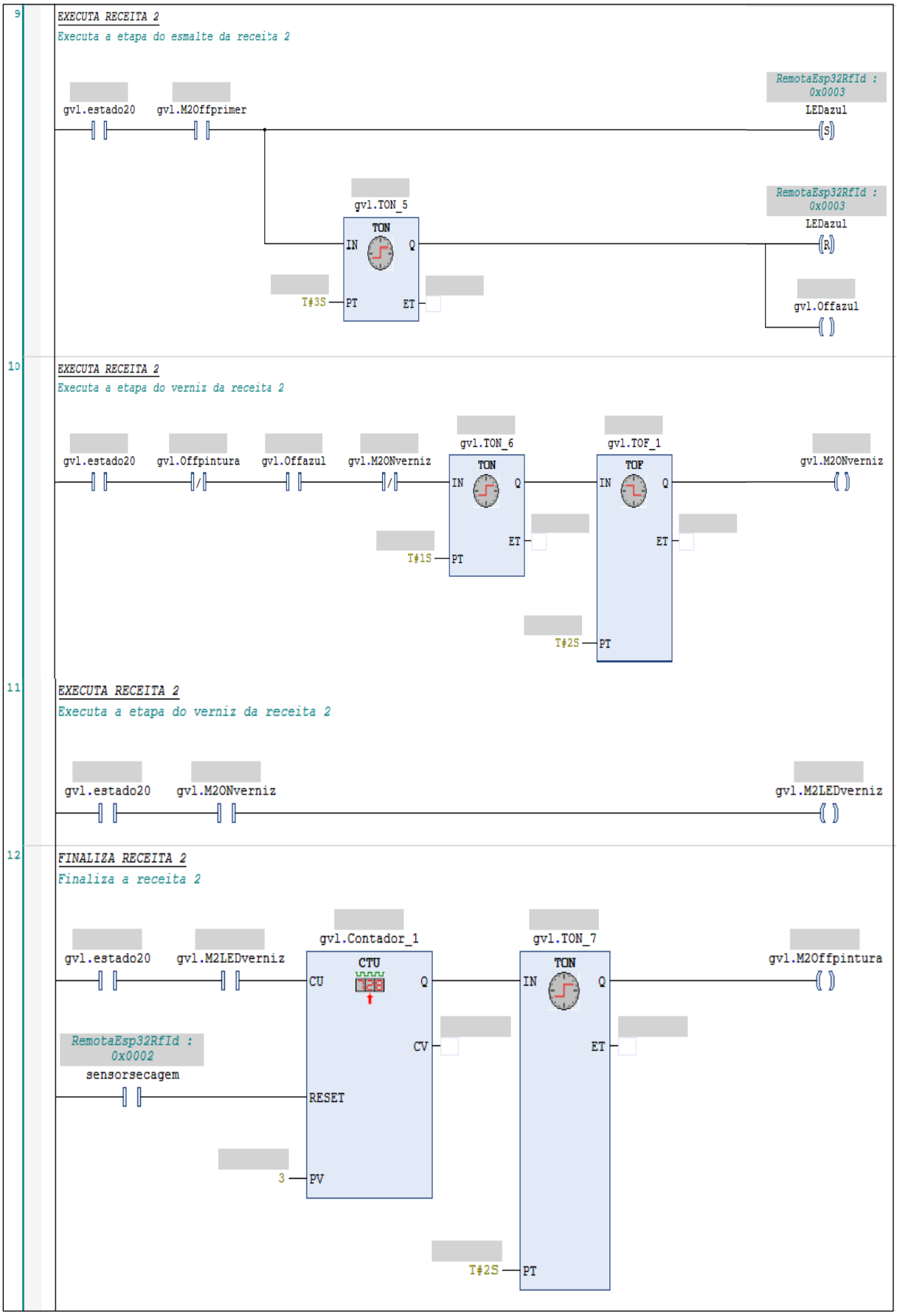
APÊNDICE E – POU “ACIONAATUADORES”

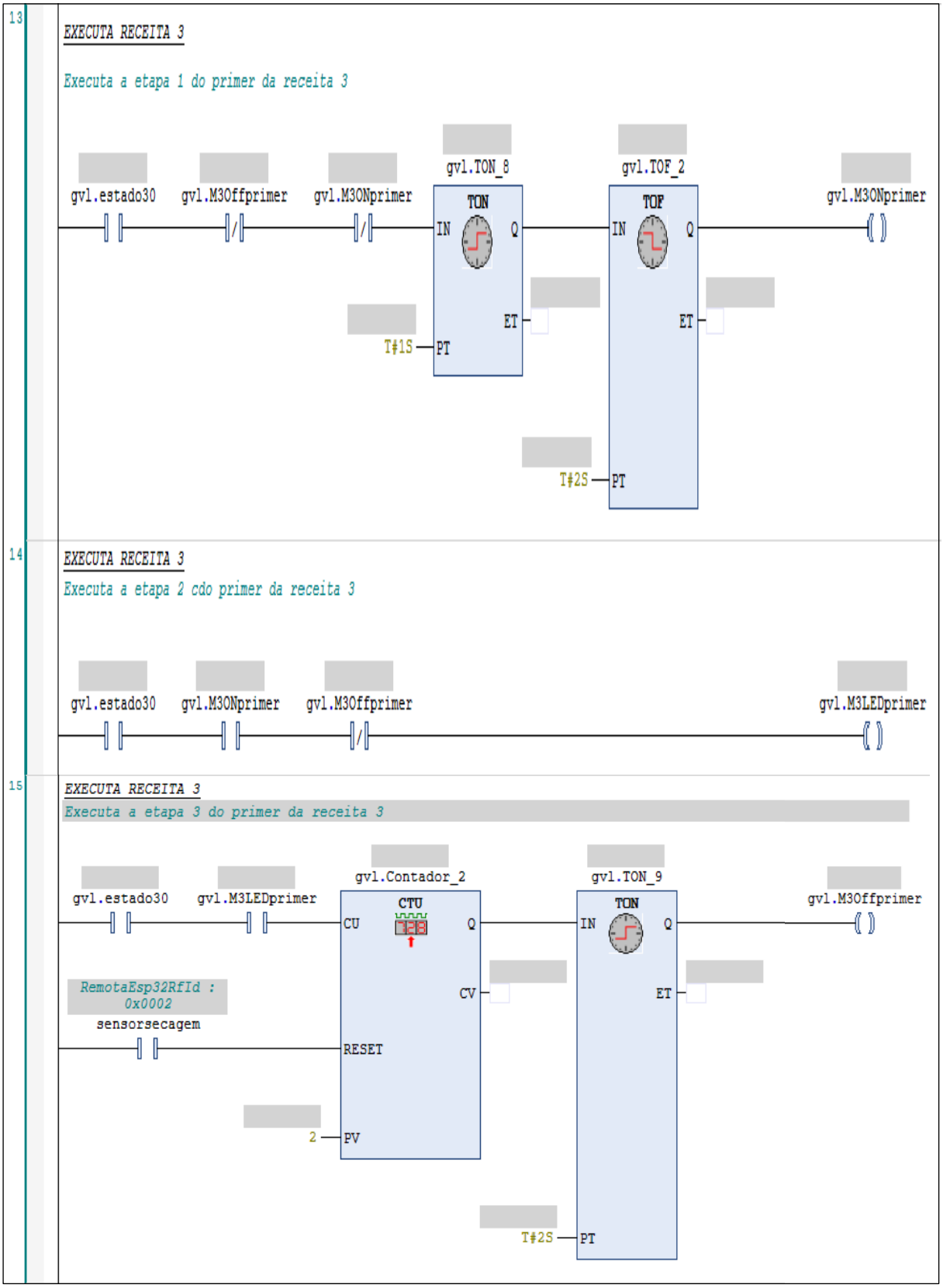


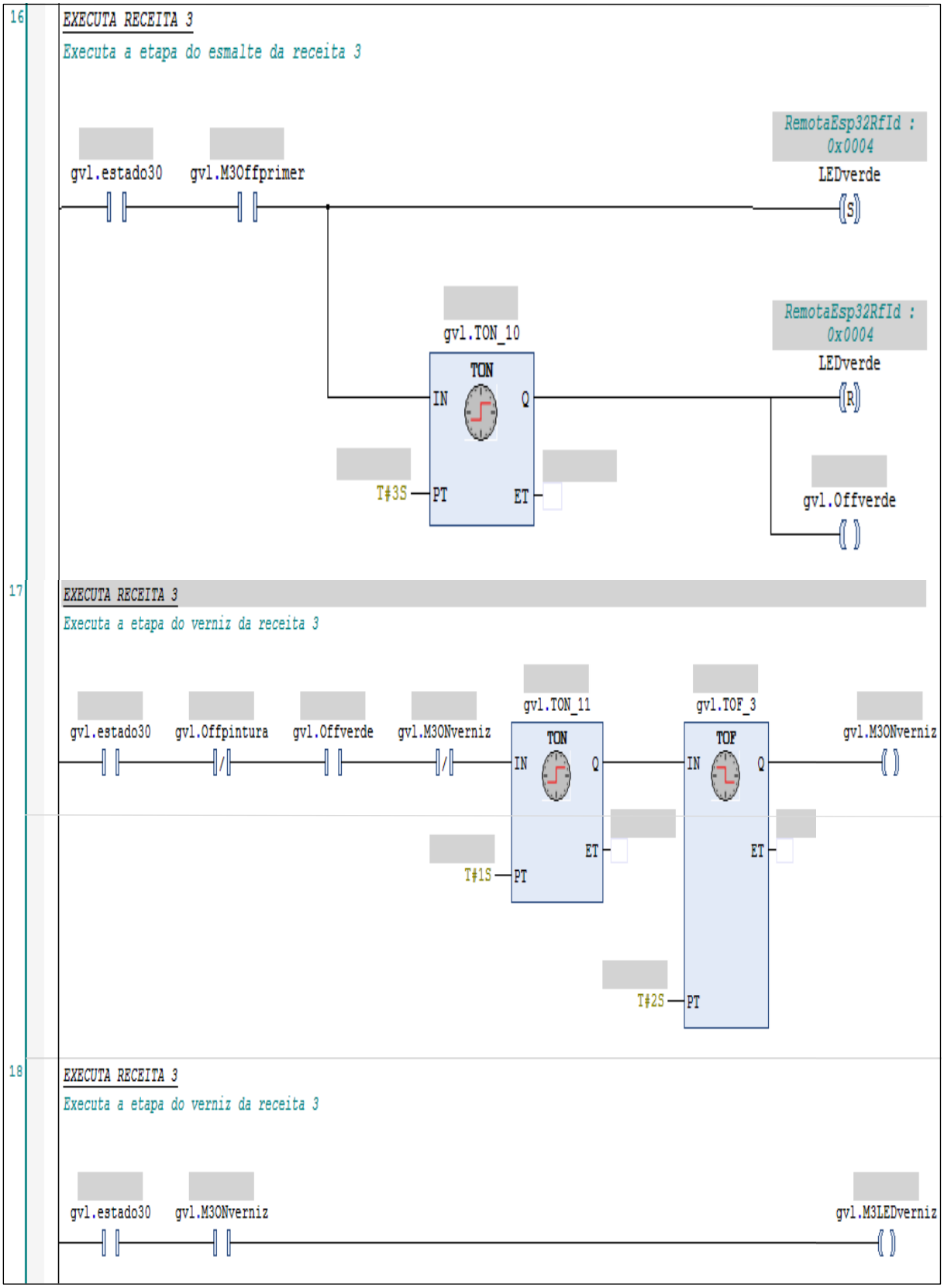

2
Motor ativado
Atribui os estados do programa que o motor será ativado
RemotaEsp32RfId : 0x0000
gvl.estado1
motor
S
gvl.estado2
gvl.estado4
3
EXECUTA RECEITA 1
Executa a etapa do primer da receita 1
GVL.estado10
gvl.M1LEDprimer
S
gvl.TON_0
TON
IN
Q
T#1S
PT
ET
gvl.M1LEDprimer
R
gvl.M1Offprimer

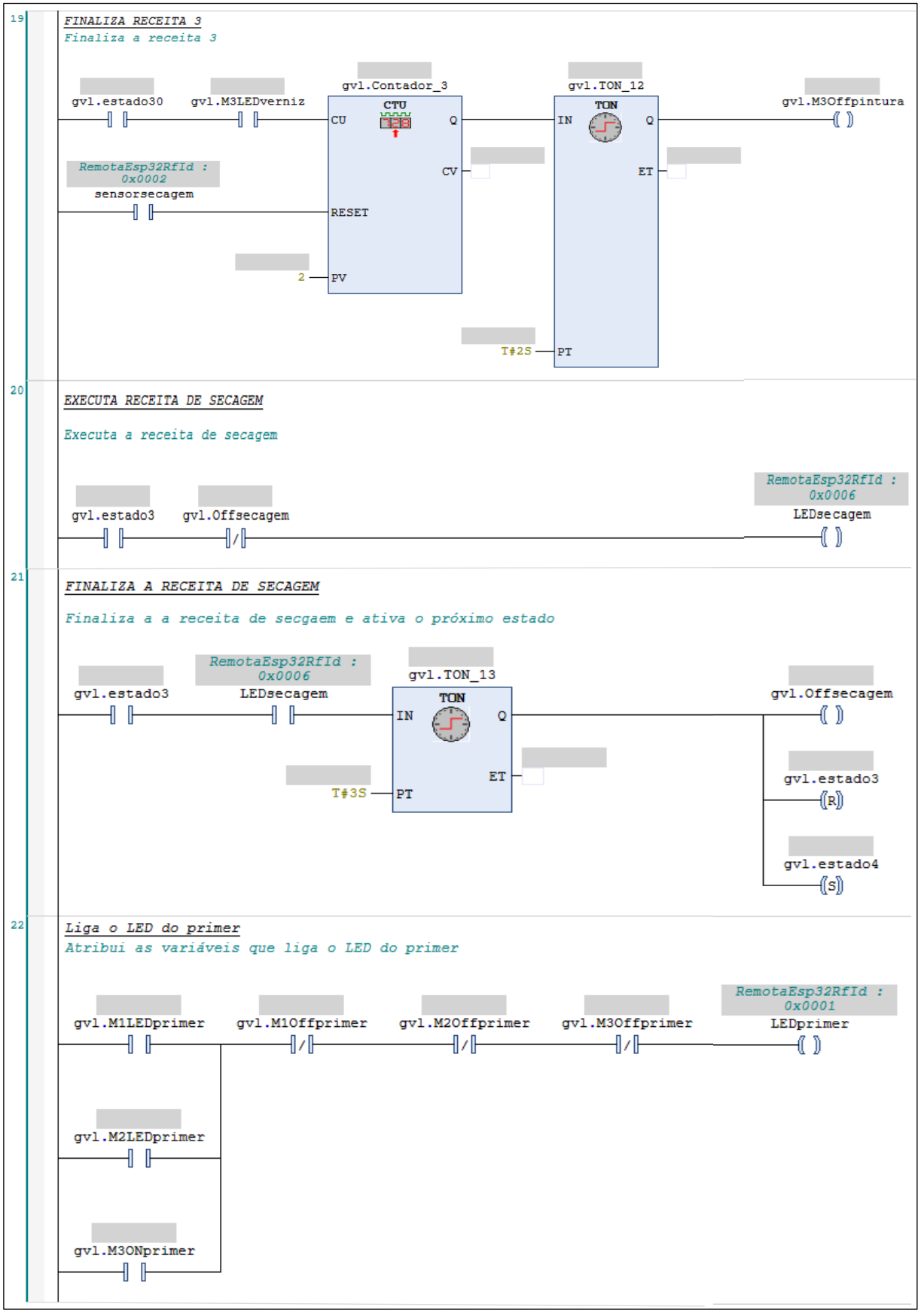

4
EXECUTA RECEITA 1
Executa a etapa do esmalte da receita 1
gvl.estado10
GVL.M1Offprimer
RemotaEsp32RfId : 0x0002
LEDvermelho
S
gvl.TON_1
TON
IN
Q
T#3S
PT
ET
RemotaEsp32RfId : 0x0002
LEDvermelho
R
gvl.Offvermelho
5
EXECUTA RECEITA 1
Executa a etapa do verniz da receita 1
gvl.estado10
gvl.Offvermelho
gvl.M1ONverniz
gvl.TON_2
TON
IN
Q
T#1S
PT
ET
gvl.TOF_0
TOF
IN
Q
T#2S
PT
ET
gvl.M1ONverniz

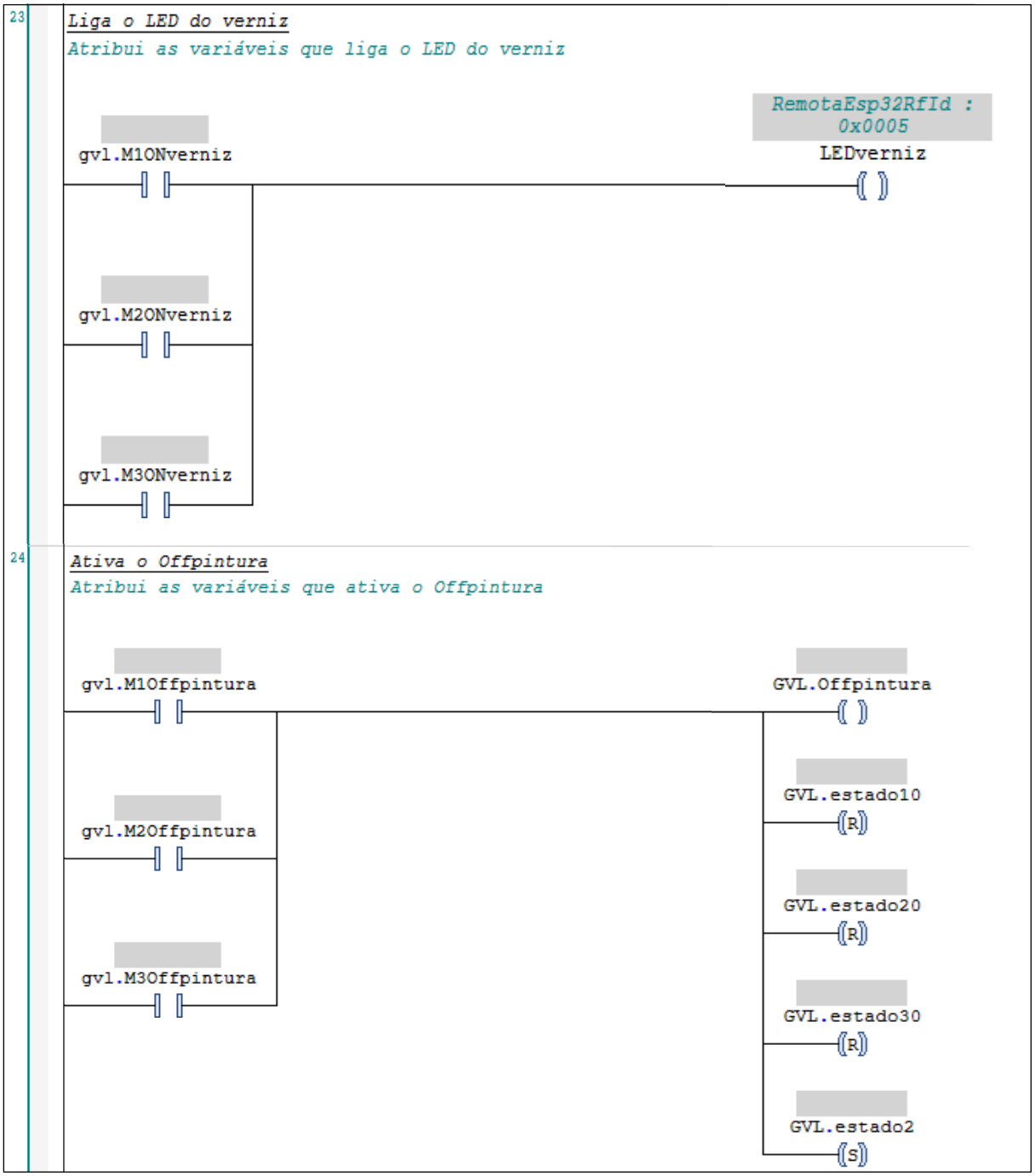

7
FINALIZA RECEITA 1
Finaliza a receita 1
GVL.estado10
gvl.M1LEDverniz
GVL.Contador_0
CTU
CU
Q
CV
RESET
PV
2
GVL.TON_3
TON
IN
Q
ET
PT
T#2S
gvl.M1Offpintura
RemotaEsp32RfId : 0x0002
sensorsecagem
8
EXECUTA RECEITA 2
Executa a etapa do primer da receita 2
gvl.estado20
gvl.M2LEDprimer
S
gvl.TON_4
TON
IN
Q
PT
ET
T#1S
gvl.M2LEDprimer
R
gvl.M2Offprimer


9
EXECUTA RECEITA 2
Executa a etapa do esmalte da receita 2
gvl.estado20
gvl.M2Offprimer
RemotaEsp32RfId : 0x0003
LEDazul
S
gvl.TON_5
TON
IN
Q
T#3S
PT
ET
RemotaEsp32RfId : 0x0003
LEDazul
R
gvl.Offazul
10
EXECUTA RECEITA 2
Executa a etapa do verniz da receita 2
gvl.estado20
gvl.Offpintura
gvl.Offazul
gvl.M2ONverniz
gvl.TON_6
gvl.TOF_1
TON
TOF
T#1S
T#2S
gvl.M2ONverniz
11
EXECUTA RECEITA 2
Executa a etapa do verniz da receita 2
gvl.estado20
gvl.M2ONverniz
gvl.M2LEDverniz
12
FINALIZA RECEITA 2
Finaliza a receita 2
gvl.estado20
gvl.M2LEDverniz
gvl.Contador_1
CTU
CU
Q
CV
RESET
PV
3
gvl.TON_7
TON
T#2S
gvl.M2Offpintura
RemotaEsp32RfId : 0x0002
sensorsecagem

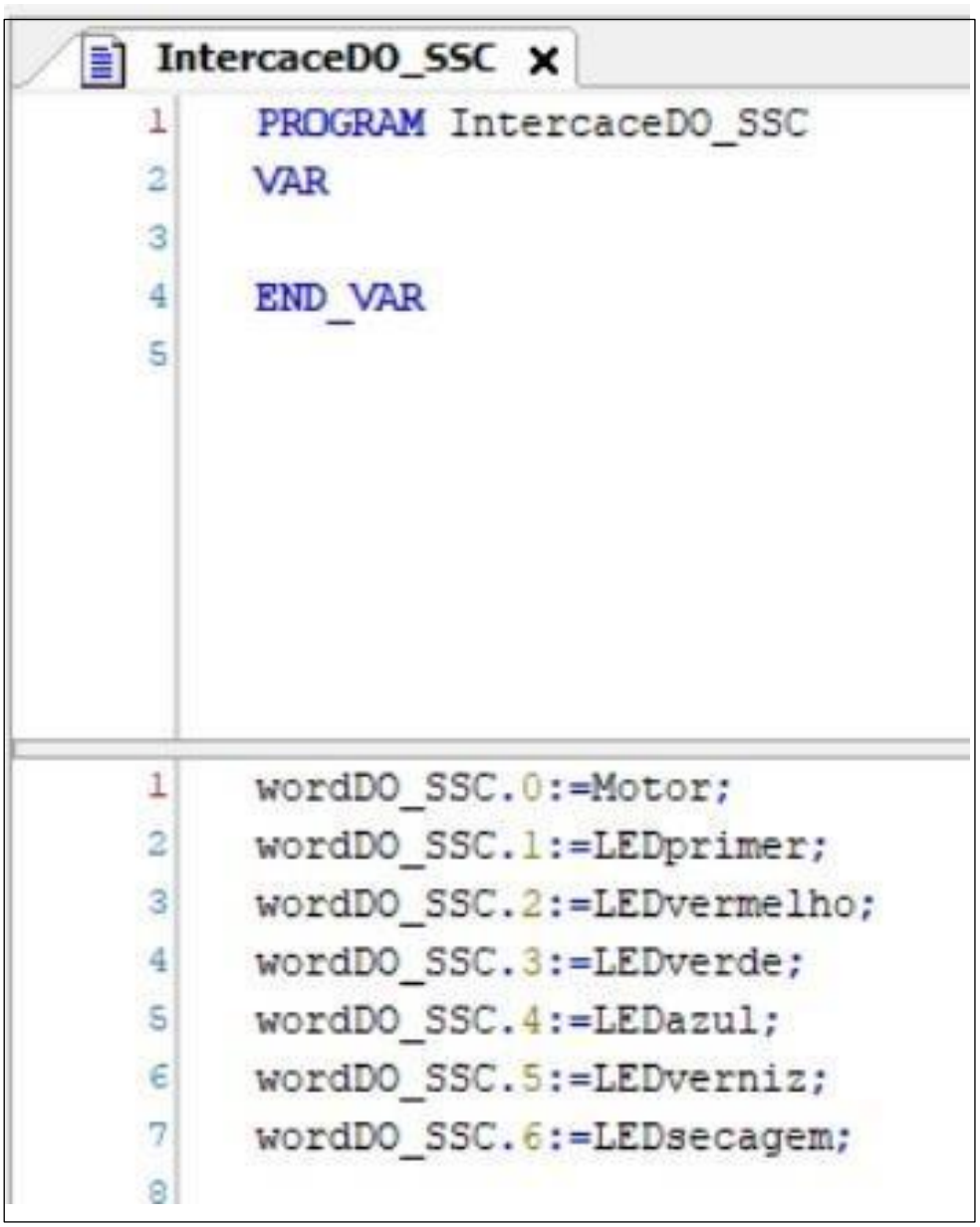

13
EXECUTA RECEITA 3
Executa a etapa 1 do primer da receita 3
gvl.estado30
gvl.M3Offprimer
gvl.M3ONprimer
gvl.TON_8
TON
IN
Q
ET
T#1S
PT
gvl.TOF_2
TOF
IN
Q
ET
T#2S
PT
gvl.M3ONprimer
14
EXECUTA RECEITA 3
Executa a etapa 2 cdo primer da receita 3
gvl.estado30
gvl.M3ONprimer
gvl.M3Offprimer
gvl.M3LEDprimer
15
EXECUTA RECEITA 3
Executa a etapa 3 do primer da receita 3
gvl.estado30
gvl.M3LEDprimer
gvl.Contador_2
CTU
CU
Q
CV
RESET
2
PV
RemotaEsp32RfId : 0x0002
sensorsecagem
gvl.TON_9
TON
IN
Q
ET
T#2S
PT
gvl.M3Offprimer


16
EXECUTA RECEITA 3
Executa a etapa do esmalte da receita 3
gvl.estado30
gvl.M3Offprimer
RemotaEsp32RfId : 0x0004
LEDverde
S
gvl.TON_10
TON
IN
Q
T#3S
PT
ET
RemotaEsp32RfId : 0x0004
LEDverde
R
gvl.Offverde
17
EXECUTA RECEITA 3
Executa a etapa do verniz da receita 3
gvl.estado30
gvl.Offpintura
gvl.Offverde
gvl.M3ONverniz
gvl.TON_11
TON
gvl.TOF_3
TOF
gvl.M3ONverniz
T#1S
T#2S
18
EXECUTA RECEITA 3
Executa a etapa do verniz da receita 3
gvl.estado30
gvl.M3ONverniz
gvl.M3LEDverniz


19
FINALIZA RECEITA 3
Finaliza a receita 3
gvl.estado30
gvl.M3LEDverniz
gvl.Contador_3
CTU
CU
Q
CV
RESET
PV
2
RemotaEsp32RfId : 0x0002
sensorsecagem
gvl.TON_12
TON
IN
Q
ET
PT
T#2S
gvl.M3Offpintura
20
EXECUTA RECEITA DE SECAGEM
Executa a receita de secagem
gvl.estado3
gvl.Offsecagem
RemotaEsp32RfId : 0x0006
LEDsecagem
21
FINALIZA A RECEITA DE SECAGEM
Finaliza a a receita de secgaem e ativa o próximo estado
gvl.estado3
RemotaEsp32RfId : 0x0006
LEDsecagem
gvl.TON_13
TON
IN
Q
ET
PT
T#3S
gvl.Offsecagem
gvl.estado3
R
gvl.estado4
S
22
Liga o LED do primer
Atribui as variáveis que liga o LED do primer
gvl.M1LEDprimer
gvl.M1Offprimer
gvl.M2Offprimer
gvl.M3Offprimer
RemotaEsp32RfId : 0x0001
LEDprimer
gvl.M2LEDprimer
gvl.M3ONprimer


23
Liga o LED do verniz
Atribui as variáveis que liga o LED do verniz
RemotaEsp32RfId : 0x0005
gvl.M1ONverniz
LEDverniz
gvl.M2ONverniz
gvl.M3ONverniz
24
Ativa o Offpintura
Atribui as variáveis que ativa o Offpintura
gvl.M1Offpintura
GVL.Offpintura
GVL.estado10
R
gvl.M2Offpintura
GVL.estado20
R
gvl.M3Offpintura
GVL.estado30
R
GVL.estado2
S

APÊNDICE F – POU “INTERCACEDO_SSC”



APÊNDICE G – VARIÁVEIS GLOBAIS

GVL

```
{attribute 'qualified_only'}
VAR_GLOBAL
    estado0:BOOL;
    estado1:BOOL;
    estado2:BOOL;
    estado3:BOOL;
    estado4:BOOL;
    estado5: BOOL;
    estado6: BOOL;
    estado7: BOOL;
    estado10:BOOL;
    estado20:BOOL;
    estado30:BOOL;

    Offvermelho: BOOL;
    Offazul: BOOL;
    Offverde: BOOL;
    Offpintura:BOOL;
    Offsecagem:BOOL;
    M1LEDprimer:BOOL;
    M2LEDprimer:BOOL;
    M3LEDprimer:BOOL;
    M1Offprimer:BOOL;
    M2Offprimer:BOOL;
    M3Offprimer:BOOL;
    M3ONprimer:BOOL;
    M1LEDverniz:BOOL;
    M2LEDverniz:BOOL;
    M3LEDverniz:BOOL;
    M1ONverniz:BOOL;
    M2ONverniz:BOOL;
    M3ONverniz:BOOL;
    M1Offpintura:BOOL;
    M2Offpintura:BOOL;
    M3Offpintura:BOOL;
    TON_0:TON;
    TON_1:TON;
    TON_2: TON;
    TON_3: TON;
    TON_4: TON;
    TON_5: TON;
    TON_6: TON;
    TON_7: TON;
    TON_8: TON;
    TON_9: TON;
    TON_10: TON;
    TON_11: TON;
    TON_12: TON;
    TON_13: TON;
    TOF_0:TOF;
    TOF_1:TOF;
    TOF_2:TOF;
```

```
53          TOF_3:TOF;
54          Contador_0: CTU;
55          Contador_1: CTU;
56          Contador_2: CTU;
57          Contador_3: CTU;
58          MTAG1: BOOL;
59          MTAG2: BOOL;
60          MTAG3: BOOL;
61
62          rotularMensagens:WORD;
63
64      END_VAR
```