



CENTRO FEDERAL DE EDUCAÇÃO TECNOLÓGICA
DE MINAS GERAIS

Departamento de Mecatrônica

Hadler Henrique Tempesta

CONTROLE E SUPERVISÃO DE PROCESSOS
INDUSTRIAIS POR MEIO DO SIMULADOR 3D
FACTORY I/O

Varginha - MG
2022

**CONTROLE E SUPERVISÃO DE PROCESSOS
INDUSTRIAIS POR MEIO DO SIMULADOR 3D
FACTORY I/O**

Hadler Henrique Tempesta

Relatório de conclusão de estágio apresentado como requisito para obtenção do título de técnico em mecatrônica pelo Centro Federal de Educação Tecnológica de Minas Gerais.

Orientação: Prof. Dr. Juliano Coêlho Miranda

Varginha - MG
2022

Lista de Figuras

1	Topologia de rede Modbus RTU.	7
2	Topologia de rede Modbus TCP/IP.	8
3	Formato do pacote Modbus RTU e TCP/IP.	10
4	Topologia de rede Modbus mista com cliente RTU.	12
5	Topologia de rede Modbus mista com cliente TCP/IP.	12
6	Topologia de trabalho do projeto.	14
7	Conexões da topologia de trabalho do projeto.	14
8	Hardware do Gateway Modbus.	15
9	Diagrama de blocos do algoritmo de conversão do Gateway.	17
10	Diagrama de tempo de um ciclo de trabalho do Gateway.	19
11	Bancada de testes para medição da latência do Gateway.	19
12	Tela inicial do Factory I/O.	20
13	Interface de simulação do Factory I/O.	21
14	Adição do driver ao projeto Factory I/O.	21
15	Aplicação genérica do driver Modbus TCP/IP do Factory I/O.	22
16	Planta de simulação industrial do laboratório LASE.	22
17	Itens usados na aplicação I do Factory I/O.	23
18	Comparação dos estágios da aplicação I: laboratório e Factory I/O.	24
19	Tanque de água do Factory I/O.	26
20	Tela inicial do Outseal Studio.	27
21	Configurações do Outseal Studio.	28
22	Oscilador com timer no Outseal Studio.	30
23	Blocos Modbus do Outseal Studio.	31
24	Controlador PID no Outseal Studio.	32
25	Cena do Factory I/O da aplicação I.	35
26	Painel de controle da aplicação I.	36
27	Cena do Factory I/O da aplicação II.	36
28	Resistor errado no shield Ethernet W5100.	40
29	Circuito de entrada do conector RJ45 do shield W5100.	41
30	Resistores de correção de impedância soldados no shield W5100.	41

Lista de Tabelas

1	Tipos de Variáveis do Protocolo Modbus.	9
2	Funções Modbus Comumente Utilizadas.	9
3	Atributos do temporizador do Outseal Studio.	30
4	Medições de latência do Gateway.	35

Sumário

1	Introdução	5
1.1	Factory I/O	5
1.2	Outseal PLC	5
1.3	Sistemas SCADA	6
1.4	Protocolo de Comunicação Modbus	6
1.4.1	Modbus RTU	6
1.4.2	Modbus TCP/IP	7
1.4.3	Tipos de Dados Modbus	8
1.4.4	Pacote de Dados Modbus	9
1.4.5	Gateway Modbus	10
1.5	Objetivo Geral	12
1.5.1	Objetivos Específicos	13
2	Desenvolvimento	14
2.1	Topologia de Trabalho	14
2.2	Gateway Modbus	15
2.2.1	Hardware	15
2.2.2	Algoritmo de Conversão	16
2.2.3	Firmware	18
2.2.4	Testes de Latência	18
2.3	Factory I/O	19
2.3.1	Iniciando os Trabalhos no Factory I/O	20
2.3.2	Drivers de Controle do Factory I/O	21
2.3.3	Aplicação I: Processo de Seleção de Pacotes	22
2.3.4	Aplicação II: Controle de Nível	26
2.4	Outseal Studio	27
2.4.1	Interface Inicial	27
2.4.2	Temporizadores	30
2.4.3	Protocolo Modbus	31
2.4.4	Controlador PID	32
3	Resultados	33
3.1	Gateway Modbus	33
3.1.1	Biblioteca para Arduino IDE	33
3.1.2	Latência do Gateway	35
3.2	Aplicação I: Processo de Seleção de Pacotes.	35
3.3	Aplicação II: Controle de Nível.	36
3.4	Arquivos do Projeto	37
4	Discussões	38
5	Conclusão	39
A	Shield Ethernet W5100 - Erro de Fabricação	40
	Referências	42

1 Introdução

A simulação é uma ferramenta importante no desenvolvimento de sistemas mais eficientes. No ambiente industrial é amplamente utilizada, desde o projeto para novas aquisições de equipamentos e produtos até a adequação da capacidade produtiva. No cenário da simulação, a realidade virtual, com ambiente 3D, pode ser utilizada como uma forma de visualização e interação do usuário com o meio simulado. No ensino, a simulação pode ser utilizada para uma aprendizagem significativa. Os softwares de simulação permitem interatividade e a possibilidade do discente simular situações experimentais e de visualizar processos que muitas vezes são de difícil obtenção pelas instituições de ensino.

1.1 Factory I/O

O Factory I/O é uma ferramenta de simulação 3D para o aprendizado de tecnologias de automação. O software foi desenvolvido pela empresa Real Games, que atua no ramo de softwares de simulação 3D a mais de uma década, e conta com uma ampla rede de apoiadores, assim como pesquisadores e desenvolvedores em universidades e centros de pesquisa [1].

Dentro do ambiente do Factory I/O é possível criar plantas industriais virtuais que podem ser controladas por tecnologias externas, tais como: Soft PLCs, PLCs e plataformas micro controladas. A comunicação do factory I/O com controladores externos pode ser feita de diversas maneiras, para esse projeto, foi escolhido o protocolo Modbus TCP/IP.

Altamente flexível, essa ferramenta possibilita a construção uma fábrica virtual completamente personalizada usando uma seleção de peças. Dentre os componentes disponíveis estão: atuadores elétricos, como: esteiras e posicionadores cartesianos; atuadores eletropneumáticos, como: pistões acionados por solenoide; e sensores comuns no setor industrial, tais como: cortina de luz, capacitivo e indutivo. Além dos componentes discretos para construção da fábrica, existem também módulos prontos, como: o paletizador, e o tanque de água.

Paralelo à possibilidade de criação de uma cena própria, o Factory I/O ainda disponibiliza várias cenas prontas que exemplificam aplicações típicas dos componentes e que podem ser usadas pelo usuário.

1.2 Outseal PLC

A Outseal é a primeira empresa indonésia a atuar no setor de desenvolvimento de tecnologia para automação. A companhia trabalha com o desenvolvimento de CLPs *open source*, assim como da ferramenta para sua programação: o Outseal Studio [2].

Os PLCs da Outseal são baseados no bootloader das placas embarcadas Arduino, de modo que o código escrito no Outseal Studio pode facilmente ser carregado em placas Arduino [2].

Embora ainda não esteja em sua versão final, o Outseal Studio conta com todas as funcionalidades essenciais da linguagem ladder, além de outros recursos, tais como: geração de PWM, controle PID e suporte ao protocolo Modbus RTU.

O Outseal PLC foi escolhido como principal interface de programação para implementação das lógicas de controle ao longo do trabalho. Ressalta-se que, um dos recursos essenciais para a execução do projeto é o suporte ao protocolo de comunicação Modbus. Na versão utilizada no projeto (Outseal Studio 3.6 Beta 5) somente é possível utilizar as funções 01 a 06 do protocolo.

1.3 Sistemas SCADA

O termo SCADA é derivado da expressão *Supervision, Control and Data Acquisition*, ou **Supervisão, Controle e Aquisição de Dados**. Os sistemas SCADA são amplamente utilizados na indústria junto a automatização de processos, realizando um papel essencial na centralização do monitoramento e controle de uma ou mais linhas produtivas.

Com a conexão dos controladores de uma planta a um sistema SCADA, é possível realizar a coleta de todos os dados de saída, entrada e internos de cada um desses dispositivos. Os dados da planta são então centralizados, e podem ser exibidos de forma conveniente em interfaces gráficas.

Além de monitoramento, o sistema SCADA também permite a escrita de valores nos controladores de sua rede, seja essa escrita manual - através da interface gráfica, por exemplo; ou por meio de rotinas pré-construídas, chamadas de *scripts*.

1.4 Protocolo de Comunicação Modbus

O Modbus é um protocolo de comunicação desenvolvido pela empresa Modicon em 1979. Hoje em dia este é o protocolo mais utilizado para redes de comunicação industriais e sistemas SCADA. O protocolo é especializado na centralização da troca de dados de entrada e saída entre dispositivos de controle [3].

O funcionamento do protocolo é baseado em uma rede cliente servidor (*Client-Server*) para monitorar e controlar os dispositivos conectados. Diversos servidores são conectados na rede, cada um com seu ID próprio (valores válidos: 1 a 254). O cliente controla o fluxo de dados no barramento, sendo capaz de requisitar ações de leitura e escrita nos dispositivos servidores, seja em suas portas I/O ou memória interna [3].

Existem duas versões do protocolo Modbus: RTU e TCP/IP

1.4.1 Modbus RTU

A versão RTU é a clássica do protocolo, ela é compatível com canais de comunicação físicos baseados na tecnologia USART (*Universal Serial Assincronous Receiver-Transmitter*).

O padrão de tensão do canal que transmite Modbus RTU é comumente o RS-485, pois este utiliza a tecnologia *differential signaling*, o que possibilita a montagem de uma rede com múltiplos dispositivos. No entanto, também é possível utilizar o padrão RS-232 para redes com apenas dois dispositivos (*single-endend signaling*).

Já a velocidade de transmissão varia entre 1200 e 115200 bps; e a topologia de rede utilizada é a barramento.

Uma rede Modbus RTU genérica é mostrada na Figura 1.

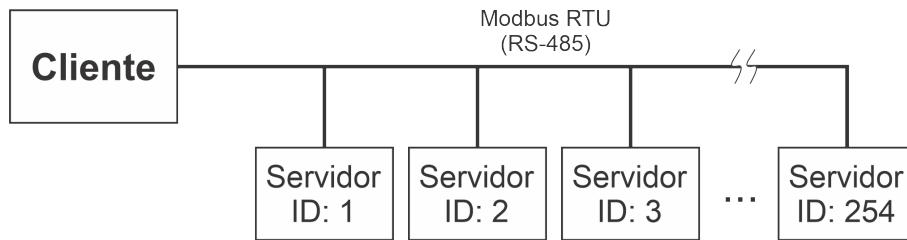


Figura 1: Topologia de rede Modbus RTU.

Quanto à transferência de dados na versão RTU, uma informação enviada pelo cliente diversos Bytes, formando o pacote Modbus RTU. Os Bytes são enviados de forma independente, porém consecutiva; o receptor deve ser então capaz de identificar um pacote completo, isso é, detectar seu início e fim. Para isso, segundo a documentação do Modbus, foi determinado que o tempo máximo entre Bytes transmitidos é de 1.5 vezes a duração de um caractere Modbus, e o tempo mínimo entre o fim de um pacote e início do seguinte é de 3.5 caracteres Modbus. Esse parâmetro é bastante importante e garante a leitura correta dos pacotes durante a implementação do protocolo.

Caractere Modbus: Um caractere Modbus é essencialmente uma unidade de tempo no contexto de transmissão de dados por meio do protocolo na versão RTU. Esse valor se refere ao tempo gasto para transmitir um Byte de dados úteis, esse valor é dependente do baudrate e também das configurações da serial.

Exemplificando: Se o canal serial tem baudrate de 9600bps, dado que o Modbus utiliza 11 bits para transferir um Byte de informação útil, o tempo de transmissão de 1 caractere completo será o tempo de envio de 11 bits a uma velocidade de 9600bps, ou seja, cerca de 1.15ms. Nessa situação, o tempo equivalente a 3.5 caracteres é de cerca de 4.01ms.

A documentação indica, no entanto, que para baudrates acima de 22000 bps, um valor fixo deve ser utilizado para o tempo de 3.5 caracteres, sendo este 1.750ms

1.4.2 Modbus TCP/IP

O TCP/IP é o protocolo mais comum de transporte de dados via internet, provendo uma comunicação confiável entre máquinas [3].

A tecnologia Ethernet tornou-se o padrão comercial para construção de redes de computadores, e portanto sua aparição no ramo industrial foi natural. Com a evolução da tecnologia, a implementação de uma rede Ethernet garante simplicidade, baixos custos e

compatibilidade generalizada com outros dispositivos. O uso da versão TCP/IP do protocolo Modbus permite uma integração generalizada do processo fabril com a intranet da empresa [3].

As características da camada física do protocolo Modbus TCP/IP seguem aquelas definidas na norma IEEE 802.3, que define as características para implementação de uma rede Ethernet. Ressalta-se que, diferentemente do Modbus RTU, o Modbus TCP/IP trabalha, normalmente, com uma topologia de rede estrela, o que permite a criação de uma rede com múltiplos dispositivos.

A troca de informações na versão TCP/IP é semelhante à RTU, existe um dispositivo Cliente, que controla o fluxo de dados na rede, podendo realizar funções de leitura e escrita nos Servidores, que recebem IDs entre 1 e 254. No entanto, esses 254 servidores podem ser alocados de forma arbitrária a endereços IP ao longo da rede Ethernet.

Na Figura 2 é mostrada a topologia de uma rede Modbus TCP/IP genérica.

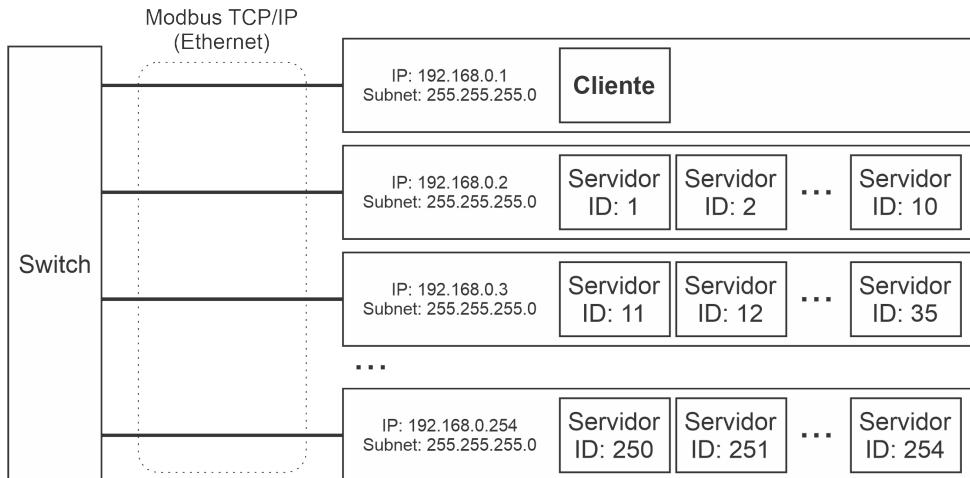


Figura 2: Topologia de rede Modbus TCP/IP.

1.4.3 Tipos de Dados Modbus

O protocolo Modbus utiliza 4 tipos de dados para transferência de informações. Esses são classificados quanto às suas permissões para leitura e escrita, e também pelo tipo de informação que ele carrega. Os nomes atribuídos aos dados Modbus são: **Coil**, **Input Status**, **Holding Register** e **Input Register**.

Os dados Coil e Input Status são do tipo booleano, isso é, são usados para variáveis do tipo True/False ou Ligado/Desligado. Já os dados Holding Register e Input Register são capazes de armazenar valores numéricos discretos.

Quanto às permissões de acesso das variáveis Modbus, elas podem ser classificadas como somente leitura (R), ou leitura e escrita (RW). As variáveis Input Status e Input Register são utilizadas para armazenar dados de entrada, e, portanto, são do tipo somente leitura. Já as variáveis Coil e Holding Register são comumente usadas para saídas, e podem receber ações de leitura e escrita.

Na Tabela 1 é resumido os tipos de variável Modbus, com suas classificações.

Tabela 1: Tipos de Variáveis do Protocolo Modbus.

Nome	Permissão	Tipo de Dado
Coil	RW	Booleano
Input Status	R	Booleano
Holding Register	RW	Numérico
Input Register	R	Numérico

1.4.4 Pacote de Dados Modbus

O campo de informação do pacote de Modbus é compartilhado entre as versões Modbus RTU e TCP/IP. Os dados são divididos entre 3 partes principais:

- **ID (1 Byte):** O campo de ID é o primeiro que aparece no pacote Modbus, ele identifica o ID do servidor que está sendo requisitado. No caso do pacote enviado pelo cliente, o ID indica o endereço para qual o pacote deve ser enviado; quando o pacote é de resposta, ou seja, enviado pelo servidor, esse campo indica a origem do pacote.
- **Função (1 Byte):** O funcionamento do Modbus é baseado em funções, elas são codificadas por um byte, e indicam o tipo de ação que é requisitada pelo cliente. Algumas das funções Modbus mais comumente usadas são mostradas na Tabela 2.

Tabela 2: Funções Modbus Comumente Utilizadas.

Função	Definição	Descrição
01	Read Coil	Leitura de Saída(s) Booleana(s)
02	Read Input Status	Leitura de Entrada(s) Booleana(s)
03	Read Holding Register	Leitura de Saída(s) Numérica(s)
04	Read Input Register	Leitura de Entrada(s) Numérica(s)
05	Write Single Coil	Escrita de Saída Booleana
06	Write Single Register	Escrita de Saída Numérica
15	Write Multiple Coils	Escrita de Múltiplas Saídas Booleanas
16	Write Multiple Registers	Escrita de Múltiplas Saídas Numéricas

Vale ressaltar que os servidores são capazes de informar erros ao cliente por meio deste campo. Para isso, a função requisitada é retornada com o bit mais significativo setado, por exemplo: um erro na execução da função 0x03 seria respondido com este campo no valor 0x83.

- **Parâmetros de Função (4+ Bytes):** O campo de parâmetros tem pelo menos 4 Bytes de tamanho, e o significado de seus dados é dependente da função Modbus.

Exemplificando, para o caso da função 01 (Read Coil), são utilizados dois bytes para indicar o primeiro endereço de um conjunto de coils, e os segundos dois bytes indicam a quantidade de coils a serem lidos a partir desse endereço.

A função 15 (Write Multiple Coils) já utiliza, além dos 4 bytes para indicação do alvo da ação, bytes extras para carregar os valores a serem escritos nos coils.

Particularidades do pacote Modbus RTU: A versão RTU do protocolo Modbus, conta ainda com 2 Bytes extras ao fim do pacote básico:

- **CRC - Cyclic Redundancy Check (2 Bytes):** Mecanismo de detecção de erros pelo método de Checagem de Redundância Cíclica.

Particularidades do pacote Modbus TCP/IP: Na versão TCP/IP existem 6 bytes que precedem os dados do pacote Modbus básico, esses dados são:

- **TI - Transaction Identifier (2 Byte):** Utilizado para identificação dos pacotes enviados, possibilitando o controle de fluxo de dados.
- **PI - Protocol Identifier (2 Bytes):** Utilizado para identificar sub protocolos, possibilita a implementação de funções personalizadas. Valor padrão: 0x0000, (Modbus Convencional).
- **Length (2 Bytes):** Indica o comprimento do restante do pacote em bytes.

Para a transmissão, todo esse conjunto de dados (TI + PI + Length + Modbus) irá formar apenas o campo de dados do pacote TCP/IP, que é precedido pelo cabeçalho do protocolo.

Na Figura 3 é possível comparar os formatos dos pacotes Modbus em suas diferentes versões.

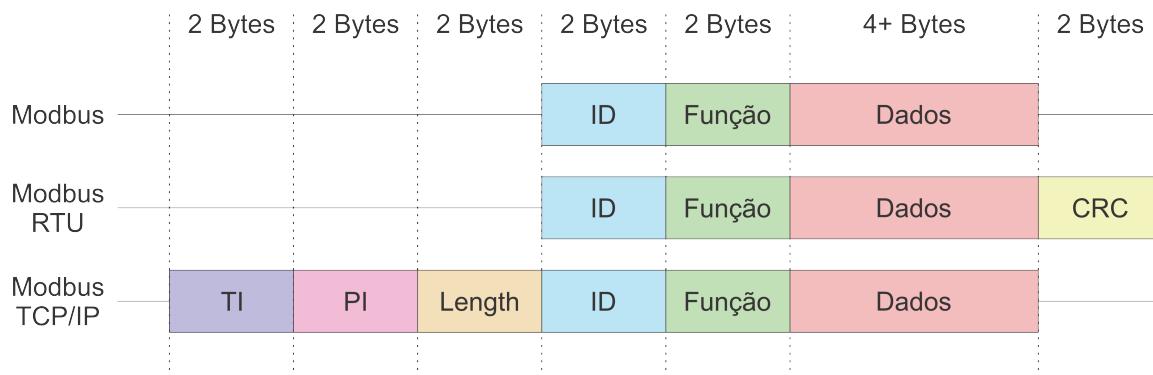


Figura 3: Formato do pacote Modbus RTU e TCP/IP.

1.4.5 Gateway Modbus

Como já citado, o Modbus é o protocolo de comunicação mais utilizado na indústria, dessa forma, grande parte das fabricantes de equipamentos de automação (como, por

exemplo, CLPs) adicionam compatibilidade com o protocolo, normalmente para a versão RTU, cujo canal de comunicação é mais robusto para o chão de fabrica.

No entanto, muitas vezes é desejado uma compatibilidade com a versão TCP/IP por seu potencial de integração. Embora ambos os protocolos sejam Modbus, eles apresentam diferenças consideráveis na estrutura de seus pacotes, e, portanto, não podem ser conectados diretamente.

A solução é a criação de um gateway, que é um dispositivo capaz de estabelecer uma comunicação entre dispositivos que utilizam protocolos de comunicação diferentes. Isso é possível pois o gateway é capaz de compreender os dois protocolos usados. Ao receber dados do protocolo A, o gateway é capaz de interpretá-los, e converter a mensagem para protocolo B, e vice-versa. Enquanto que o gateway resolve o problema em questão, sua maior desvantagem é a latência adicional gerada no canal de comunicação devido à conversão, essa grandeza deve ser conhecida para garantir que não afete o funcionamento da rede

Para o Modbus, é possível criar um gateway Modbus, capaz de converter os pacotes do Modbus RTU para o Modbus TCP/IP, e vice-versa. No entanto, deve-se atentar ao cliente Modbus, que é um dispositivo único. Ele pode estar no lado RTU ou TCP/IP da rede, o que gera dois casos diferentes.

Cliente RTU: Caso o cliente seja conectado no lado RTU da rede, a função do gateway será monitorar o canal RTU. Quando um pacote vindo do cliente é identificado, é feita a conversão para o TCP/IP e o dado é encaminhado para o canal Ethernet.

Uma resposta é então aguardada pelo cliente e, caso o servidor requisitado seja um servidor Modbus TCP/IP, a resposta será coletada pelo gateway, que converte a informação de volta para RTU, e a injeta no canal original.

Na Figura 4 é mostrada um exemplo de rede Modbus com gateway e cliente RTU.

Cliente TCP/IP: Já no caso de o cliente trabalhar com Modbus TCP, o gateway também fica encarregado de fazer o controle do fluxo de dados entre os dispositivos.

Diferentemente do cliente RTU, o cliente Modbus TCP/IP não necessariamente aguarda a resposta de uma requisição para enviar a próxima. Esse controle de fluxo de informações é natural do Modbus TCP/IP e previsto no protocolo, no entanto, o comportamento em questão não é compatível como o Modbus RTU, e causaria certamente colisão de dados entre os servidores RTU e o gateway no canal RS-485.

As colisões do canal RTU podem ser evitadas com um manejo correto dos pacotes pelo gateway, o que aumenta o grau de complexidade do dispositivo.

Na Figura 5 é mostrado um exemplo de rede Modbus mista com cliente TCP/IP.

O gateway Modbus é um dispositivo necessário para o desenvolvimento desse projeto, uma vez que o Factory I/O oferece suporte apenas para o Modbus TCP/IP, enquanto que o Outseal Studio permite uso apenas do Modbus RTU.

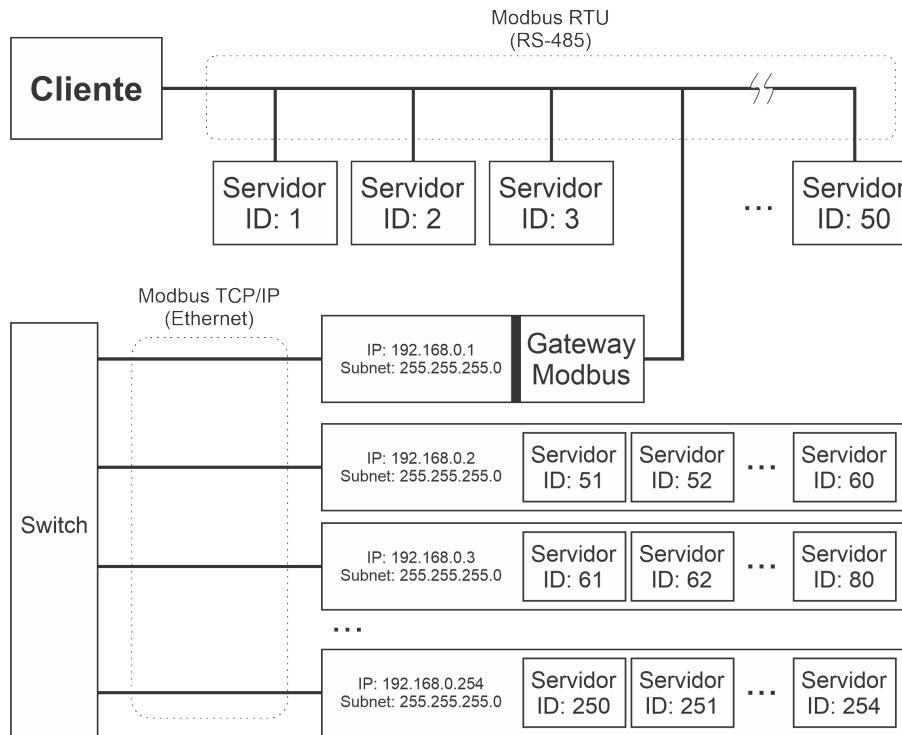


Figura 4: Topologia de rede Modbus mista com cliente RTU.

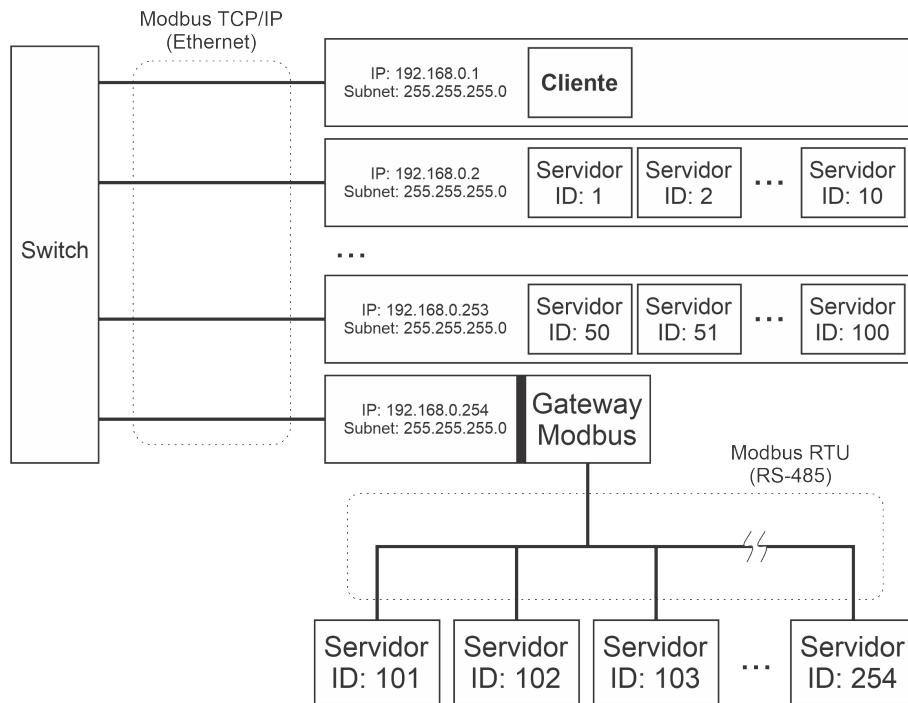


Figura 5: Topologia de rede Modbus mista com cliente TCP/IP.

1.5 Objetivo Geral

O objetivo primário da pesquisa foi estabelecer um conjunto de etapas para o uso inicial da ferramenta Factory I/O no contexto de sistemas SCADA. Os casos de estudo serão associados ao contexto educativo das disciplinas de automação industrial, robótica, instrumentação, controle e desenvolvimento de sistemas supervisórios.

1.5.1 Objetivos Específicos

- Desenvolver cenas de plantas fabris no Factory I/O;
- Criar um algoritmo de controle para as plantas desenvolvidas;
- Codificar os algoritmos de controle em linguagem Ladder no Outseal Studio;
- Elaborar um algoritmo de gateway Modbus;
- Implementar um gateway Modbus na plataforma micro controlada Arduino;
- Estabelecer uma rede de comunicação entre o Outseal PLC e o Factory I/O;
- Documentar as etapas para criação de uma rede de comunicação entre o Factory I/O e o Outseal PLC.

2 Desenvolvimento

A primeira etapa do projeto foi a criação do gateway Modbus, pois sem esse dispositivo não é possível estabelecer uma comunicação entre o Factory I/O e Outseal PLC. Na sequência, foram criadas cenas para controle no Factory I/O, e então implementadas lógicas de controle para os processos criados no Outseal Studio.

2.1 Topologia de Trabalho

Como o Factory I/O é o simulador da planta produtiva, foi natural que seu controlador se tornasse um servidor Modbus, assim o processo pode ser controlado pelo cliente da rede. O Outseal PLC, por sua vez, se tornou o cliente.

Assim a topologia de trabalho foi completamente determinada, sendo ela composta por: um servidor Modbus TCP/IP, um cliente Modbus RTU, e um gateway Modbus.

Embora fosse possível criar o canal Ethernet entre o servidor e o gateway de forma direta, utilizando um cabo Ethernet crossover, para fins de monitoramento e simplicidade, esse canal foi centralizado por meio de um switch.

A topologia é mostrada na Figura 6.

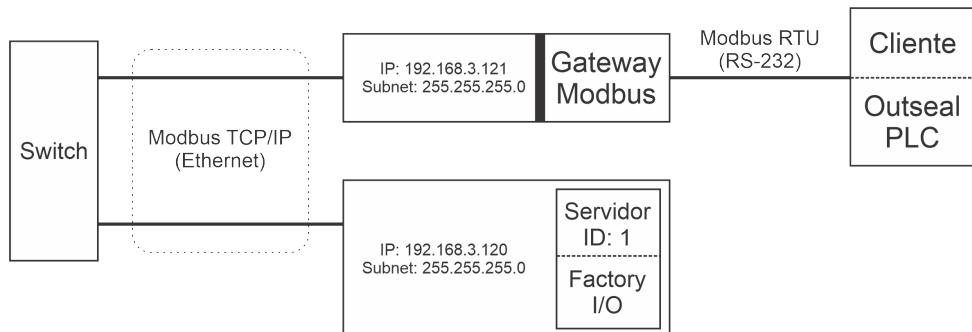


Figura 6: Topologia de trabalho do projeto.

As conexões físicas feitas para criação da topologia da Figura 6 são mostradas com mais detalhes na Figura 7.

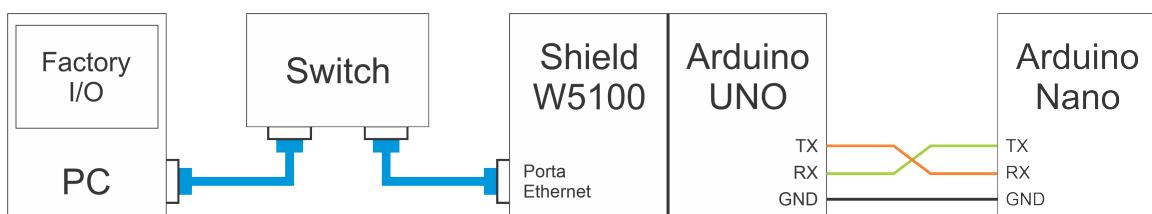


Figura 7: Conexões da topologia de trabalho do projeto.

2.2 Gateway Modbus

Como comentado, o gateway Modbus foi um dispositivo crucial para o projeto, já que o Factory I/O tem suporte somente à versão TCP/IP do Modbus, enquanto que o Outseal Studio só oferece o Modbus RTU.

2.2.1 Hardware

Com intuito de desenvolver um gateway de menor custo, ele foi baseado na plataforma embarcada Arduino. A placa específica escolhida para o desenvolvimento foi a Arduino UNO, equipada com um shield Ethernet W5100. Ambas as placas são mostradas na Figura 8.

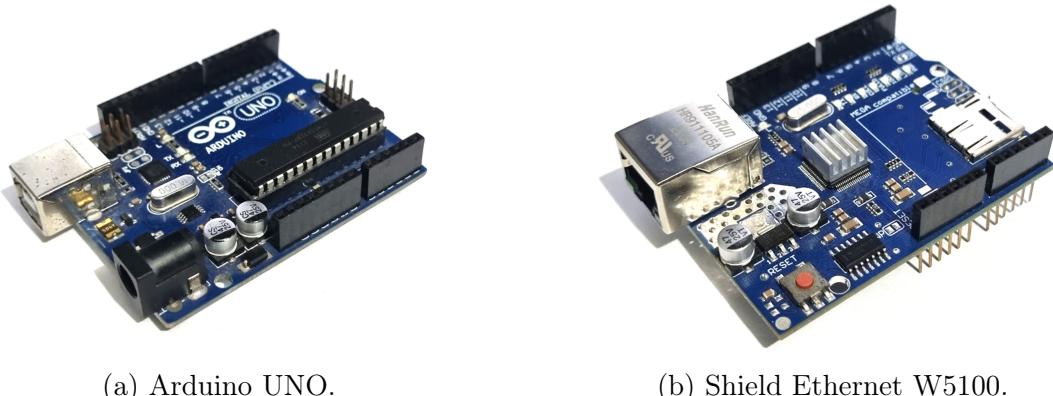


Figura 8: Hardware do Gateway Modbus.

Modbus RTU: O Modbus RTU pode ser utilizado diretamente com o hardware USART nativo da placa Arduino, que usa os padrões de tensão CMOS.

Caso seja necessário a criação de um canal físico mais robusto, que utilize o padrão RS-485, é possível conectar à USART (Pinos: 0-RX, 1-TX) do Arduino um CI MAX485. Esse é um componente comercial capaz de realizar a conversão do padrão de tensão TTL, ou CMOS, para RS-485.

Modbus TCP/IP: Já o Modbus TCP/IP não é suportado pelo Arduino, pois necessita de compatibilidade com a tecnologia Ethernet. Para tornar o Arduino compatível com a Ethernet, foi utilizado um shield Ethernet W5100.

O shield Ethernet W5100 utiliza o protocolo SPI para comunicação com o Arduino, deixando o módulo USART livre para uso com o Modbus RTU. O shield é equipado com os componentes necessários para comunicação em um canal físico Ethernet.

Para implementação de projetos com o shield, é utilizada a biblioteca `Ethernet.h`, que é nativa à Arduino IDE. Essa biblioteca permite facilmente configurar os parâmetros para uso da Ethernet, como: IP, máscara de sub-rede, gateway padrão, entre outros.

Nota: Existem alguns lotes do shield Ethernet W5100 com um erro de fabricação no hardware que compromete seu funcionamento. No Apêndice A é mostrado como identificar e corrigir esse defeito do shield.

2.2.2 Algoritmo de Conversão

O algoritmo de conversão entre as versões Modbus implementado é baseado na semelhança entre os pacotes dos dois protocolos, que contém um campo de informação idêntico, assim como já exibido na Figura 3.

Como no projeto foi utilizado o cliente no lado RTU da rede, o algoritmo foi criado exclusivamente para essa situação.

1^a Etapa: Enquanto em espera (*stand-by*), o gateway verifica se algum dado é recebido na serial (lado RTU da rede). Quando é detectado a chegada de dados, se espera até que o pacote todo chegue, e então é verificado o CRC para concluir se os dados recebidos apresentaram ou não algum erro na transmissão.

Caso um erro seja detectado por meio do CRC, o gateway retornará uma mensagem de erro adequada para o cliente e o processo volta para o estado de *stand-by*. Já se não houver erro, o processo de conversão é iniciado.

2^a Etapa: A conversão do pacote RTU para TCP/IP consiste em remover o campo de CRC, e adicionar o novo cabeçalho precedendo o campo de informações original.

Por conta de o cliente utilizar o Modbus RTU, ele não realiza o controle do fluxo de dados, mas sim envia uma requisição e aguarda sua resposta antes de enviar a próxima mensagem. Dessa forma, o campo de TI(*Transaction Identifier*) deve ser colocado como 0x0000. O campo PI(*Protocol Identifier*) também é colocado como 0x0000, pois é utilizado o Modbus convencional.

Por fim existe o campo *Length*, este recebe o tamanho em Bytes do campo de informações do pacote original.

Esses dados são então agrupados e enviados para a rede Ethernet por meio das funções da biblioteca `Ethernet.h`, que gera o cabeçalho necessário para a transferência desses dados na rede.

3^a Etapa: Após o envio dos dados para o lado TCP da rede, o gateway agora espera pela resposta do servidor. Quando os dados são recebidos, o gateway é capaz de realizar a leitura adequada do pacote graças ao campo *Length* do Modbus TCP/IP, que especifica exatamente quantos Bytes devem ser lidos.

Ressalta-se que a resposta do servidor pode nunca chegar, seja por um problema na rede, ausência do servidor ou qualquer outro empecilho. Para isso, é implementado uma lógica simples de *time out*, isso é, é esperada a resposta por um período determinado de tempo, e caso nenhuma resposta seja recebida nesse intervalo o processo é encerrado.

4ª Etapa: A última etapa do processo é a conversão dos dados recebidos do servidor para Modbus RTU, e envio para o cliente.

A conversão do TCP/IP para RTU consiste na remoção dos 6 primeiros Bytes do pacote (TI, PI e Length), e então a adição dos 2 Bytes de CRC no final do mesmo.

Uma vez que o pacote Modbus RTU é montado, ele é enviado pela porta serial do gateway para o lado RTU da rede.

Na Figura 9 é mostrado um diagrama de blocos do processo de conversão.

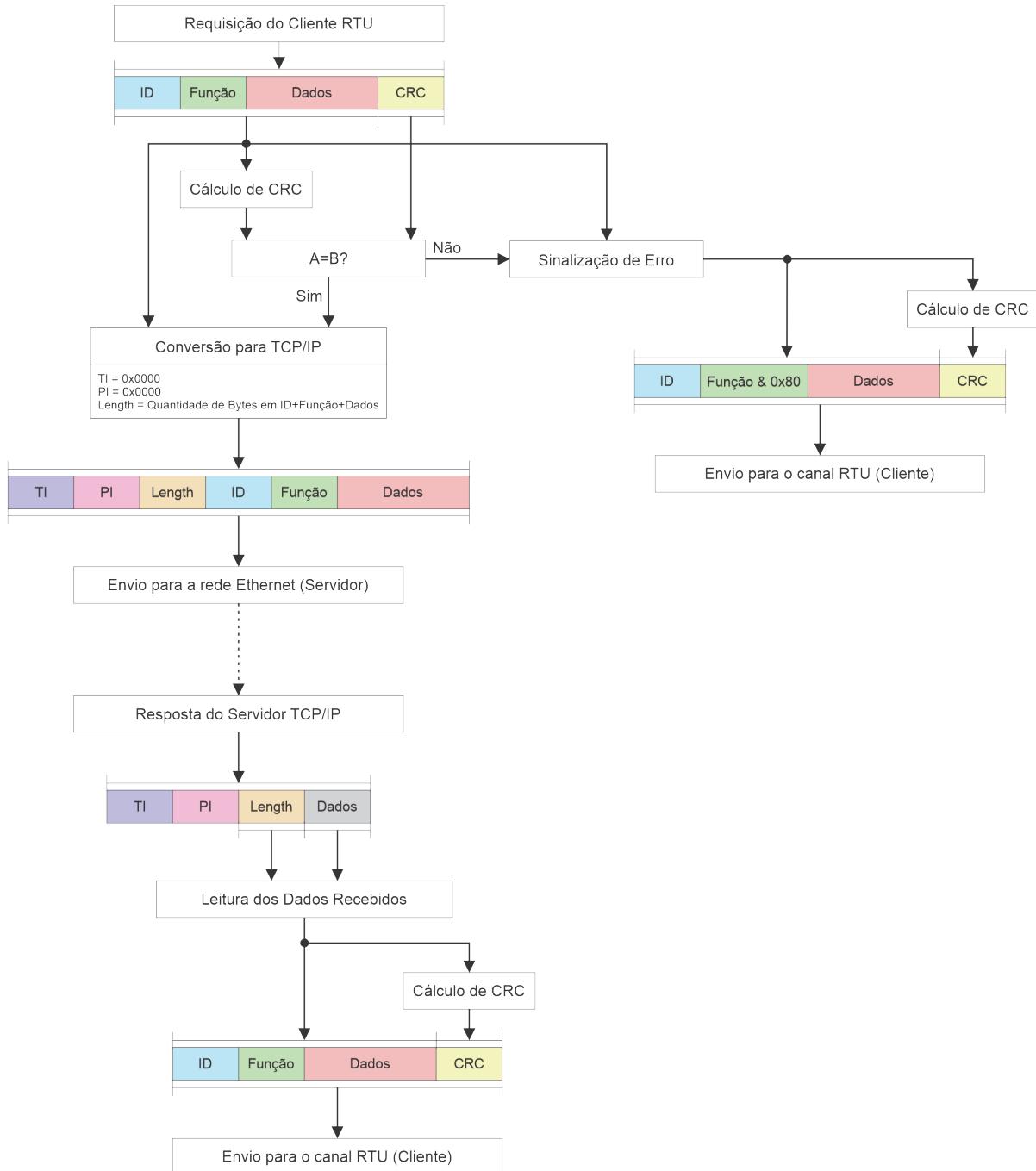


Figura 9: Diagrama de blocos do algoritmo de conversão do Gateway.

2.2.3 Firmware

Uma vez desenvolvido o algoritmo de funcionamento do gateway, foi então possível implementá-lo na plataforma Arduino. Para isso foi usada a Arduino IDE e um conjunto de bibliotecas nativas básicas tais como: `SPI.h`, `Arduino.h` e `Ethernet.h`, que garantiram o acesso às funcionalidades do Arduino e do shield Ethernet, assim como a comunicação entre os dois.

O firmware é baseado em uma classe C++ que recebeu, além do algoritmo de conversão, alguns métodos que permitem ao usuário a configuração adequada do gateway, tais como o baudrate e seu endereço IP. Essa classe foi então convertida em uma biblioteca.

2.2.4 Testes de Latência

Como já comentado, a maior desvantagem da utilização do gateway é sua latência, isso é, o tempo que o mesmo toma para as conversões entre protocolos, que é naturalmente adicionado à latência total da rede.

Para o gateway implementado no projeto, foi adicionada uma funcionalidade para a medição dessa latência. Foram utilizados dois pinos do Arduino para mostrar os diferentes estágios da comunicação: um pino foi programado para ficar em estado lógico alto somente durante a conversão de pacotes de Modbus RTU para Modbus TCP/IP; enquanto que um segundo pino foi configurado para estar em estado lógico alto somente durante a conversão de pacotes Modbus TCP/IP para Modbus RTU.

A partir dessa estrutura é possível medir a largura desses dois pulsos com equipamentos externos como um osciloscópio, a partir das medições conclui-se a magnitude de 4 variáveis importantes:

- t_S : Latência do servidor;
- $t_{RTU,TCP}$: Tempo contado a partir do recebimento do pacote RTU pelo Arduino, contemplando a conversão RTU para TCP/IP e envio dos dados para o Shield Ethernet;
- $t_{TCP,RTU}$: Tempo contado a partir do recebimento do pacote TCP pelo Arduino, contemplando a conversão TCP/IP para RTU e envio dos dados para a USART;
- t_0 : Latência total da rede, incluindo o servidor e o gateway.

Na Figura 10 é mostrado o diagrama de tempo dos estados lógicos dos dois pinos durante um ciclo de troca de dados na rede.

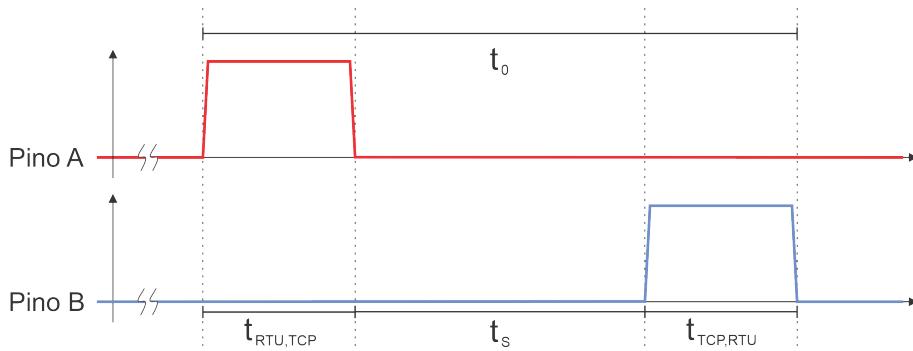


Figura 10: Diagrama de tempo de um ciclo de trabalho do Gateway.

Para a realização dessas medidas na prática, foi montada rede mostrada na Figura 6 em laboratório, e então os pinos sinalizadores de estado (pinos 2 e 3) foram monitorados por meio de um osciloscópio.

Para essa medição inicial, o cliente da rede foi substituído, ao invés do Outseal PLC, foi criado um script em linguagem Python para enviar uma quantidade fixa de pacotes na rede. Enquanto que para o servidor, foi utilizado o simulador de CLP ModRSsim2¹.

Durante os testes foi enviado sempre um pacote fixo: leitura do coil 0x0000 no servidor 1 (pacote: 01 01 00 00 00 01 FD CA). As medidas de tempo foram então parametrizadas em função do baurate do canal RTU, e para cada bateria de medições, foram enviados 1000 pacotes Modbus e medidos os parâmetros de tempo supracitados.

A bancada de trabalho e equipamentos utilizados² para as medições são mostradas na Figura 11.

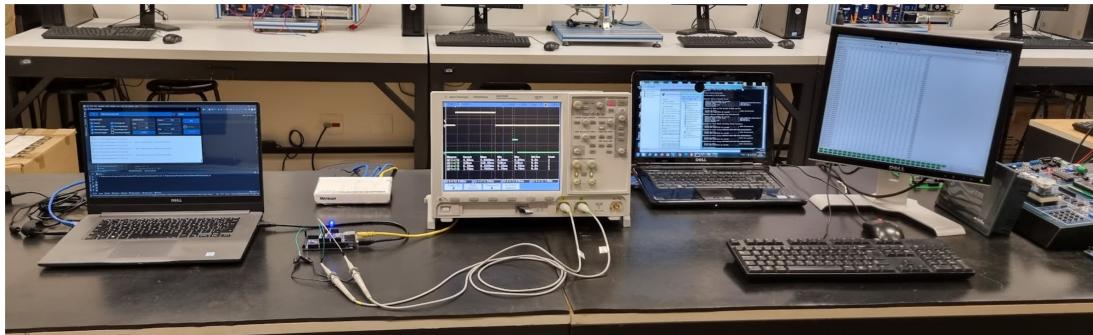


Figura 11: Bancada de testes para medição da latência do Gateway.

2.3 Factory I/O

Uma vez que o gateway Modbus foi desenvolvido, foi possível iniciar a criação de aplicações no Factory I/O.

O objetivo do projeto foi um dos modelos prontos do Factory I/O, e também elaborar uma planta do zero. Para o modelo pronto, foi utilizada a planta da caixa d'água, enquanto

¹Modbus Simulator for RS-232 and TCP/IP. Disponível em: <https://sourceforge.net/projects/modrssim2/>. Acesso em 28 fev. 2023.

²Os testes de latência foram realizados no Laboratório de Automação e Sistemas Embarcados (LASE) do CEFET-MG campus Varginha.

que a planta desenvolvida foi baseada no kit de simulação industrial da empresa DE LORENZO, instalado no laboratório LASE do CEFET-MG campus Varginha.

Na sequência, foram documentadas as etapas para iniciar os trabalhos com o Factory I/O e configurações dos drivers de controle. Por fim, são discutidos os componentes implementados nas duas aplicações oficiais do projeto.

2.3.1 Iniciando os Trabalhos no Factory I/O

Uma vez aberto, a tela inicial do Factory I/O é como a mostrada na Figura 12.

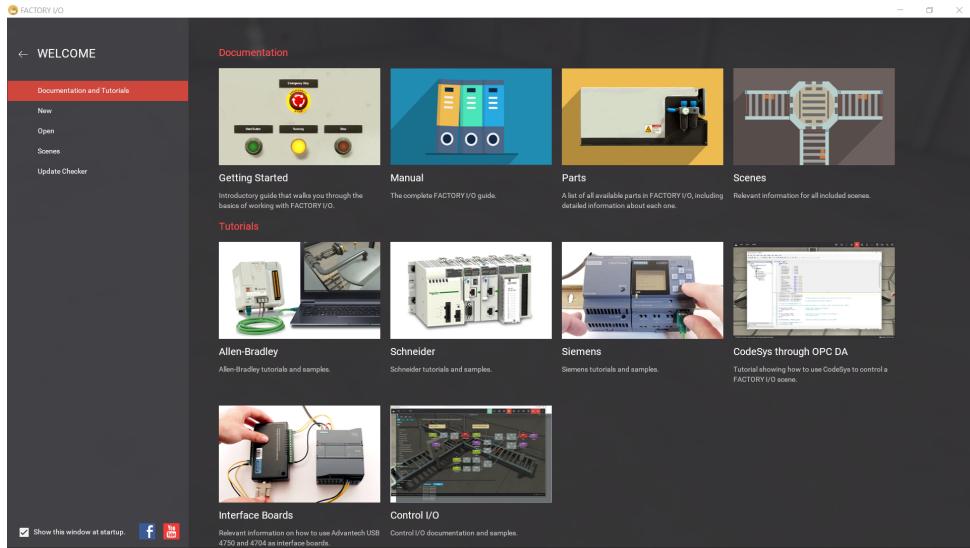


Figura 12: Tela inicial do Factory I/O.

O Factory I/O é um software com uma documentação³ bastante completa, e logo na tela inicial permite que o usuário accesse as páginas de tutoriais, manuais e documentação das peças e cenas disponíveis. A utilização dessa documentação é altamente recomendada.

Com uma quantidade elevada de recursos, a utilização do Factory I/O é simplificada quando feita juntamente com sua documentação, por exemplo: no ambiente 3D de desenvolvimento do software, mostrado na Figura 13, é possível utilizar 3 câmeras diferentes para navegar pelo galpão, sendo elas: *Orbit Camera* (Câmera Orbital), *Fly Camera* (Câmera Voadora) e *First Person Camera* (Câmera em Primeira Pessoa). Essas câmeras podem ser selecionadas no menu superior direito da interface (marcado em vermelho na Figura 13) e cada uma possui características distintas. Na seção *Getting Started -> Navigating* do manual do Factory I/O existem não somente mais detalhes sobre cada uma dessas câmeras, mas também seus controles e particularidades.

Esse padrão de detalhamento na documentação se repete para todos os componentes e ferramentas do simulador e, portanto, recomenda-se a sua leitura e consulta.

³A documentação do Factory I/O é originalmente escrita em inglês, porém, caso seja de interesse do usuário, é possível realizar a tradução das páginas por meio de ferramentas de terceiros, tais como a extensão do Google Tradutor, nativa aos browsers mais comuns como o Google Chrome e o Microsoft Edge.

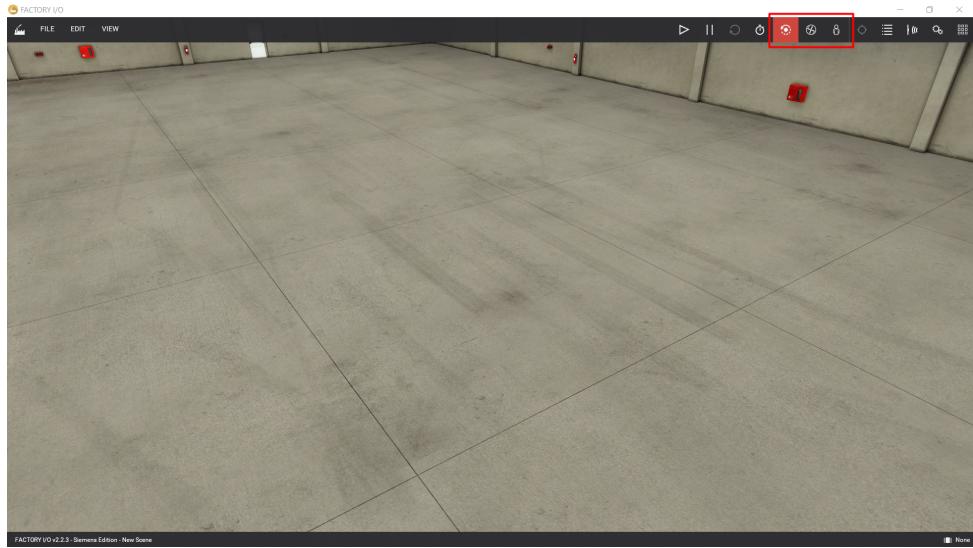


Figura 13: Interface de simulação do Factory I/O.

2.3.2 Drivers de Controle do Factory I/O

Um tópico que merece uma atenção especial dentro do simulador são os drivers de controle. Para realizar o controle dos componentes da cena, isso é, implementar um algoritmo ou lógica de controle, são utilizados drivers.

O Factory I/O oferece diversos drivers de controle, que são compatíveis com diferentes CLPs e protocolos de comunicação, além do software Control I/O, uma extensão do Factory I/O que permite a implementação da lógica de controle em uma linguagem de blocos.

A inserção do driver no projeto pode ser feita de três formas: no menu Drivers, disponível no canto inferior direito da tela; pelo atalho F4; ou pelo menu File -> Drivers, como visto na Figura 14.

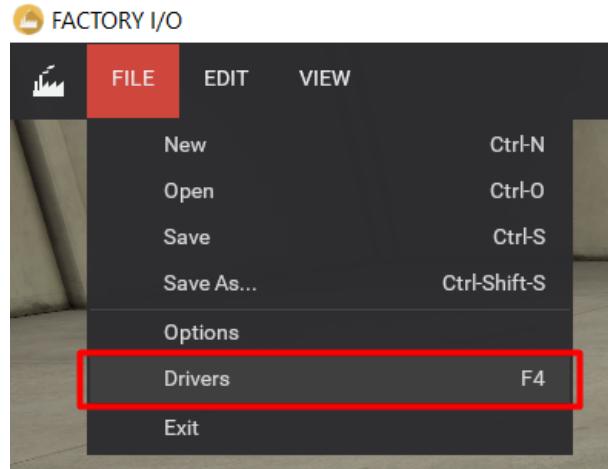


Figura 14: Adição do driver ao projeto Factory I/O.

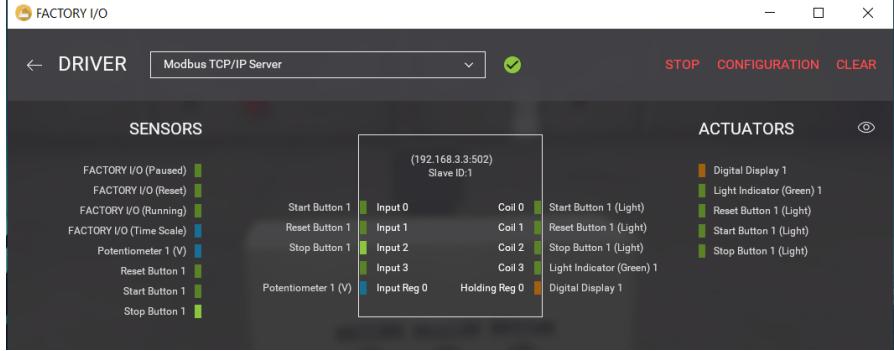
Para esse projeto, o driver de interesse foi o driver "Modbus TCP/IP Server". Com esse driver, o Factory I/O emula um CLP virtual, no qual é possível conectar as diversas

entradas e saídas da cena, esse CLP também naturalmente é compatível com o protocolo Modbus TCP/IP e age como um Servidor, tomando o IP da máquina Host como seu próprio.

Uma vez que o driver foi inserido, é possível observar na tela todas as variáveis do processo. Na Figura 15 é mostrado o driver Modbus TCP/IP Server conectado às saídas e entradas de uma cena simples (Painel com elementos).



(a) Cena simples.



(b) Driver Modbus TCP/IP Server.

Figura 15: Aplicação genérica do driver Modbus TCP/IP do Factory I/O.

As variáveis são divididas entre dois grupos, como se vê na Figura 15b, sendo que as saídas(Atuadores) são colocadas na direita, e entradas(Sensores) na esquerda. A coloração do elemento indica seu tipo: Verde é um dado booleano, Azul é um dado numérico decimal(tensão), e Verde um dado numérico inteiro.

Ainda na Figura 15b é mostrado o campo Configuration, que trás as opções de configuração do driver. Para o caso do Modbus TCP/IP são mostradas configurações como quantidade de saídas e entradas, assim como o adaptador de rede Ethernet.

2.3.3 Aplicação I: Processo de Seleção de Pacotes

Como já citado, uma das aplicações foi feita do zero, e inspirada no módulo de simulação industrial da empresa DE LORENZO, instalado no laboratório LASE do CEFET-MG Campus Varginha. O módulo em questão é mostrado na Figura 16.

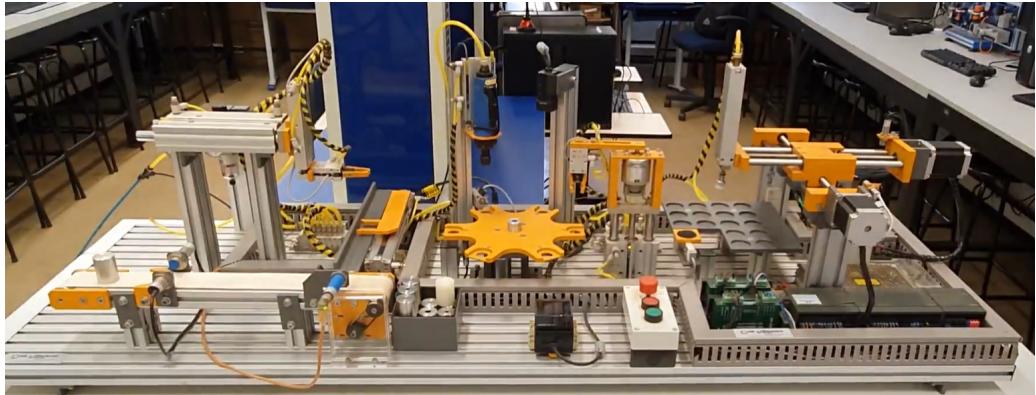


Figura 16: Planta de simulação industrial do laboratório LASE.

Na planta do laboratório LASE, atuadores elétricos e pneumáticos, juntamente com sensores industriais, são usados para selecionar peças segundo seu material, peso e geometria. Especificamente, as peças são feitas de plástico, alumínio ou aço, e tem furos com diferentes diâmetros. Isso permite classificá-las como não-metálicas(plástico), metálica-leve(alumínio) ou metálica-pesada(aço), e então subclassificá-las segundo o diâmetro de seu furo. Para isso, são usados sensores do tipo indutivo, capacitivo, piezoelétrico e uma câmera.

Na aplicação desse projeto, foi feito um processo semelhante, tanto quanto a seu objetivo, como quanto à rota das peças na planta. Como peças, foram utilizados os itens: *Blue Raw Material*, *Green Raw Material*, *Box (S)*, *Box (M)* e *Box (G)*, mostrados respectivamente da esquerda para a direita na Figura 17.



Figura 17: Itens usados na aplicação I do Factory I/O.

O objetivo foi ignorar as peças plásticas, e então classificar as caixas segundo seu tamanho (que, dentro do Factory I/O, está diretamente ligado à sua massa). Para a separação entre os plásticos e as caixas, foi usado o sensor de visão (*Vision Sensor*) e para a classificação de tamanho, a esteira com balança (*Conveyor Scale*).

Para isso, no primeiro estágio, é feita a seleção entre as peças plásticas e caixas por meio do sensor de visão. As peças plásticas são descartadas, enquanto que as caixas são empurradas para outra esteira até um posicionador. A caixa é então colocada pelo posicionador na próxima etapa do processo.

A etapa dois consiste em medir o tamanho da caixa, o que é feito por meio de um sensor cortina de luz. Nota: Esse dado, embora coletado, não contribui para a lógica de seleção, uma vez que o peso e tamanho das caixas são atrelados dentro do Factory I/O. A caixa é então recebida por um posicionador, que a leva até o próximo estágio.

No estágio três, a caixa é pesada pela esteira com balança, e classificada entre: leve, mediana ou pesada. As caixas são então levadas via uma esteira até um posicionador de três eixos, que as coloca nas devidas rampas de saída: as caixas pesadas na primeira rampa, as medianas na segunda e leves na terceira.

Os estágios foram feitos como módulos no Factory I/O, e sua construção foi tão semelhante quanto possível à planta do laboratório LASE. Na Figura 18 são comparados os componentes da planta real com a cena implementada no Factory I/O.

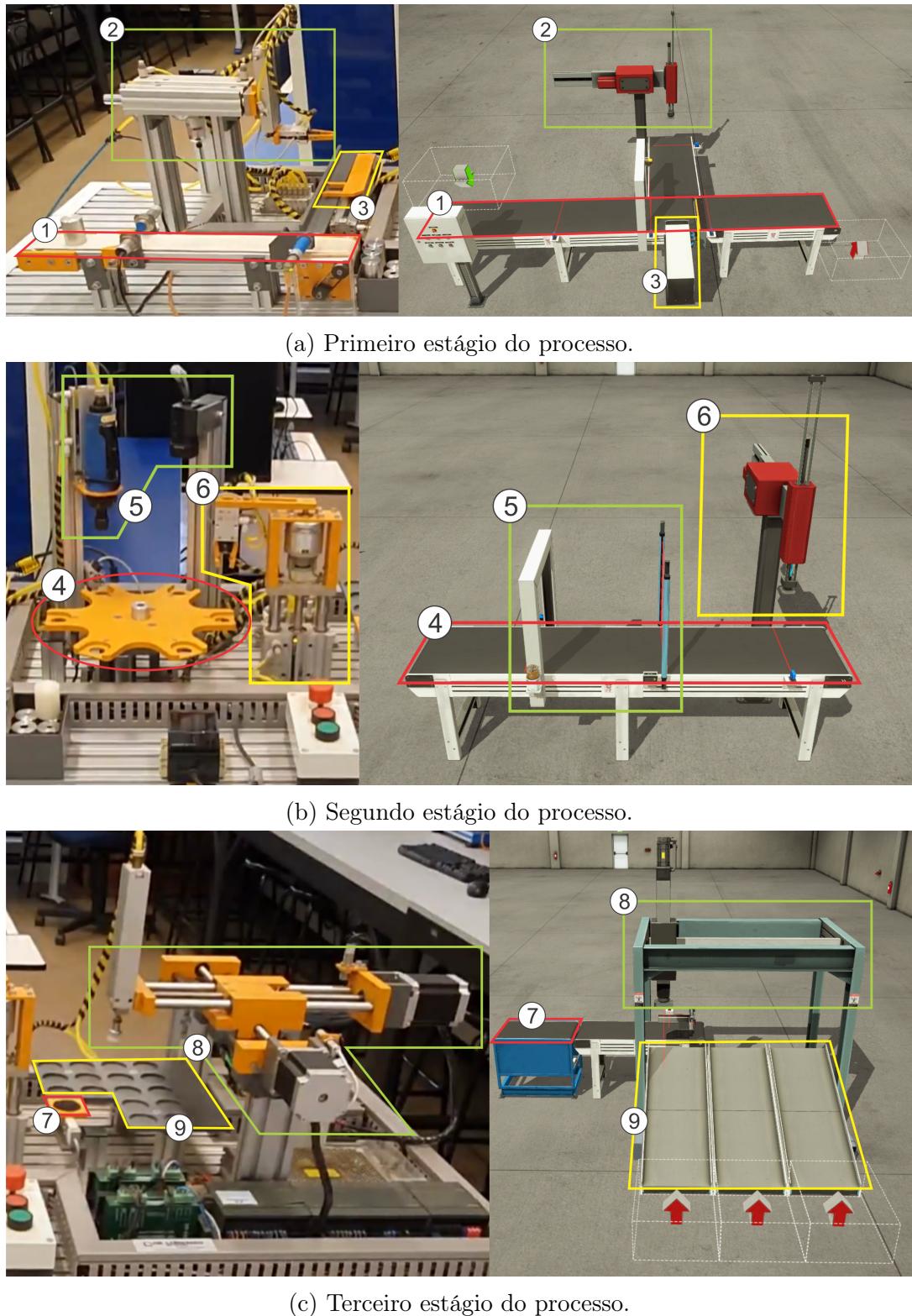


Figura 18: Comparação dos estágios da aplicação I: laboratório e Factory I/O.

Ao longo do processo de criação dos módulos, foram utilizados componentes específicos para algumas tarefas. Abaixo são citadas as motivações de escolha dessas partes:

- **Belt Conveyor (Esteira de correia):** Essas esteiras são classificadas pelo próprio Factory I/O como componentes de carga leve, e foram, portanto, ideais para essa aplicação, na qual foram usadas apenas caixas e itens pequenos.

- ***Two-Axis Pick and Place* (Posicionador de dois eixos)**: Esses posicionadores foram utilizados pois tem um tamanho ideal para transferir as caixas entre os módulos do processo. Além disso, eles tem uma sequência de acionamento bastante semelhante aos dos pistões da planta real, o que reforça o objetivo de tornar as aplicações tão próximas quanto possível. Esses componentes estão presentes nos itens 2 da Figura 18a e 6 da Figura 18b.
- ***Pusher (Impulsor)***: O impulsor foi o componente mais próximo ao garfo presente na planta real (item 3 da Figura 18a), e conseguiu realizar a função do mesmo: mover as peças aceitas para a esteira lateral.
- ***Stop Blade (Barreira)***: A barreira foi usada no processo 3 (Figura 18a) para garantir que as caixas aceitas fossem paradas antes de continuarem no processo.
- ***Pick and Place (Posicionador)***: O posicionador é uma estação com 3 eixos de liberdade para carregar itens. Esse componente, usado no processo 8 da Figura 18c, foi ideal pra o posicionamento das caixas após a classificação. Além disso, essa estação é extremamente semelhante ao posicionador da planta real, contando até mesmo com uma "garra"baseada em uma ventosa.
- ***Left Positioner (Posicionador Esquerdo)***: Esse componente foi utilizado junto ao processo 8 da Figura 18c para posicionar a caixa de forma adequada para que o posicionador de 3 eixos a pudesse pegar.
- ***Vision Sensor (Sensor de visão ou "Câmera")***: Esse sensor foi usado no início do processo (processo 1 da Figura 18a), ele possibilitou a diferenciação entre as peças plásticas e as caixas de papelão, de fato, esse é o único componente com essa capacidade.
- ***Retroreflexive Sensor (Sensor retro reflexivo)***: O sensor óptico retro reflexivo é capaz de detectar objetos independentemente de seu material e foram usados amplamente ao longo do processo. Ao longo das esteiras, vários desses sensores foram colocados, com eles é possível identificar a posição dos itens ao longo do processo.
- ***Conveyor Scale (Balança com esteira)***: A balança foi um dos componentes mais importantes do processo, mostrada no processo 7 da Figura 18c, é com ela que é feita a classificação das caixas.
- ***Emitter (Emissor)***: O emissor é um pseudo-componente do processo, é ele que faz a emissão das peças que serão tratadas. Em uma aplicação real, o emissor seria uma linha de alimentação (robotizada ou não) precedendo o processo estudado.
- ***Remover (Removedor)***: O removedor é outro pseudo-componente, sua função é excluir as peças uma vez que passaram pelo processo.

Novamente, a leitura da documentação do Factory I/O, que pode ser acessada a partir da tela inicial do software, é indispensável para a compreensão e uso dos componentes supracitados.

2.3.4 Aplicação II: Controle de Nível

O segundo processo deste trabalho foi baseado na estação disponível no Factory I/O: *Water Tank*. Essa estação é baseada em um tanque de água, com uma válvula de entrada e uma de saída, ambas com regulagem numérica de abertura; o tanque também é equipado com um sensor de nível da água, e um sensor de fluxo de saída.

A estação em questão é mostrada na Figura 19.

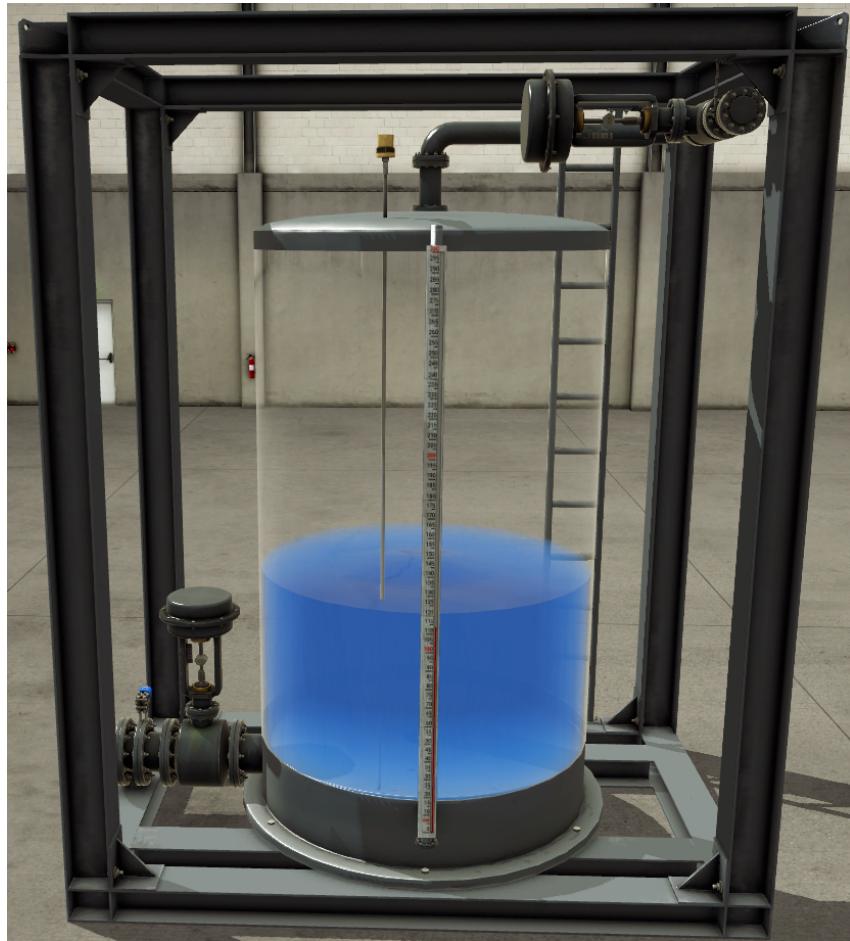


Figura 19: Tanque de água do Factory I/O.

O objetivo nessa planta foi implementar um controle PID para o controle do nível de água.

2.3.4.1 Levantamento da Função de Transferência da Planta A primeira etapa dessa aplicação foi o levantamento da função de transferência da caixa d'água, mais especificamente, da válvula de enchimento em relação ao nível de água.

Para a caixa d'água em questão, dado que a válvula de saída está fechada, o volume de água $V(t)$ é dado pelo volume atual(V_0) somado ao volume adicionado pela válvula de entrada com fluxo $F(t)$:

$$V(t) = V_0 + \int_0^t F(t) dt$$

A mesma expressão escrita de forma diferencial, e então levada ao domínio de Laplace, permite alcançar a função de transferência desejada:

$$\frac{\partial V(t)}{\partial t} = F(t) \quad \xrightarrow{\mathcal{L}} \quad sV(s) = F(s) \quad \rightarrow \quad \frac{F(s)}{V(s)} = s$$

2.3.4.2 Controlador PID Ao analisar a função de transferência obtida por meio da ferramenta RLTool do software MATLAB, concluiu-se que o controlador PD é o mais adequado para o controle de nível da planta. Uma vez que obtida a resposta desejada na ferramenta, foram extraídas as constantes do controlador: $K_p \approx 100$ e $K_d \approx 10$.

No entanto, o controlador obtido é referente somente à válvula de entrada, não considerando a de escape. Naturalmente, para manter o equilíbrio de nível, a válvula de entrada deve abrir tanto quanto a válvula de saída. Por meio de testes na simulação, verificou-se que uma pequena ação do controlador I ($K_I \approx 0.1$) é o suficiente para estabilizar o processo.

2.4 Outseal Studio

Uma vez que definidas as aplicações do projeto, assim como seus algoritmos de controle e sequências de acionamento, foi então utilizado o Outseal PLC para implementação dessas lógicas em linguagem Ladder.

Abaixo é mostrada a sequência de passos necessária para utilizar o Outseal Studio, incluindo a apresentação do software e suas instruções principais.

2.4.1 Interface Inicial

O Outseal Studio usado no projeto foi a versão V3.6 Beta 5, cuja tela inicial é mostrada na Figura 20.

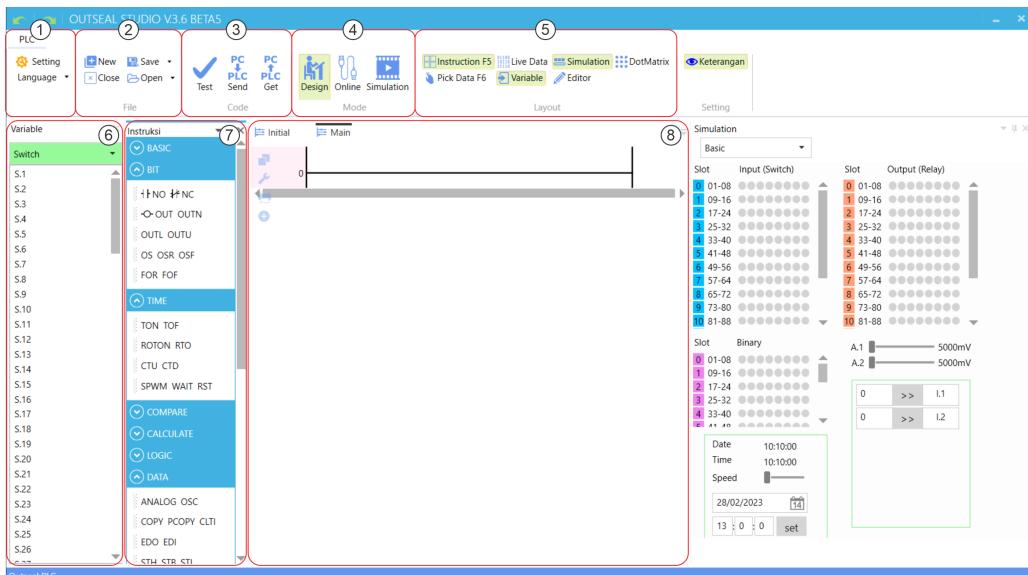


Figura 20: Tela inicial do Outseal Studio.

Na Figura 20 são mostrados 8 campos bastante importantes do software, que são discutidos abaixo:

(1) Configurações: No campo 1 é possível realizar as configurações relacionadas ao PLC, assim como alterar o idioma do programa, que possui as versões em indonésio e inglês.

Quando clicado no item **Settings** a interface de configurações é aberta, essa é mostrada na Figura 21. Nesse menu é possível renomear o projeto, selecionar o CLP⁴ (Hardware) e a porta que este está conectado. No menu **Filter** da seção **Settings** também é possível definir tempos de *debounce* para as entradas digitais.

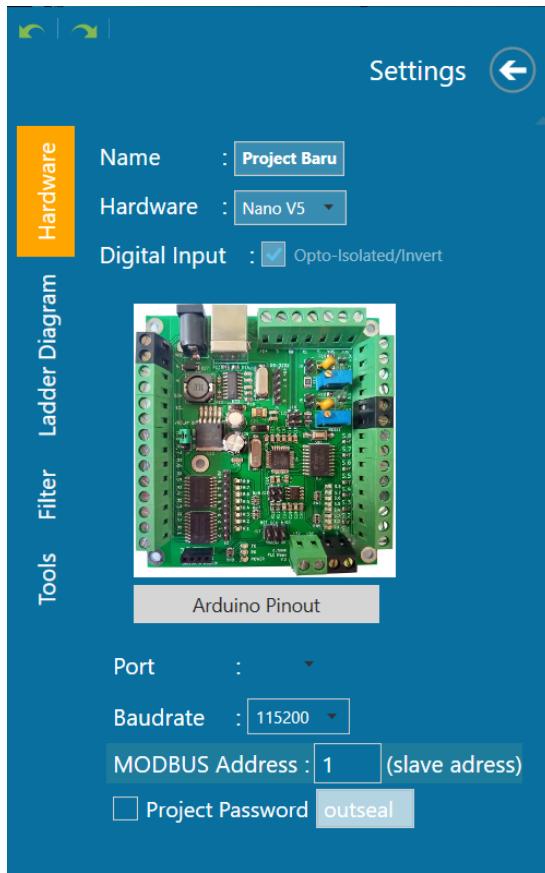


Figura 21: Configurações do Outseal Studio.

(2) Arquivo: Nessa região é possível salvar, abrir, fechar ou criar um projeto.

(3) Código: No terceiro submenu, é possível realizar a verificação de sintaxe do código com o botão *Check*, e caso o código esteja correto, o envio ao CLP pode ser feito por meio do botão *Send*. O Outseal também permite extrair o código do CLP por meio do botão *Get*.

⁴Para uso do Arduino UNO/Nano (ou qualquer placa Arduino baseada no microcontrolador ATMEGA 328P) basta selecionar o Hardware Nano V5.

(4) Modo: O Outseal PLC tem 3 modos de trabalho:

- **Design:** Esse é o modo de desenvolvimento de implementação do código Ladder, permitindo a inserção de instruções e *networks*.
- **Online:** No modo online, o Outseal Studio estabelece uma comunicação direta com o CLP, e mostra o estado das entradas e saídas no diagrama Ladder. Esse modo é bastante útil para checagem e debug da lógica implementada.
- **Simulation:** No modo de simulação, é possível forçar o estado das variáveis para checagem da lógica implementada, de forma muito semelhante ao modo Online, a diferença é que esse teste é feito sem conexão ao CLP.

(5) Layout: No submenu Layout são selecionadas as caixas de ferramentas que ficarão visíveis durante o desenvolvimento do código. Ao longo do desenvolvimento desse projeto, as caixas de instruções (*Instruction*) e de variáveis (*Variable*) se mostraram as mais úteis.

(6) Variáveis: O menu de variáveis armazena uma lista das variáveis booleanas do controlador. As variáveis são separadas entre três tipos: Switch, que são atreladas às entradas físicas do controlador; Relay, que são atreladas às saídas físicas do controlador; e Binary, que podem ser usadas como variáveis internas.

Em qualquer uma das variáveis deste menu, um clique duplo permite a adição de um comentário sobre a mesma; para aplicações maiores, a possibilidade de identificar cada variável com um pequeno texto é uma função extremamente útil, e recomenda-se fortemente que esse recurso seja usado.

Nesse menu ainda é indicado automaticamente quais variáveis fazem parte do processo, isso pode ser visto com a presença de um sombreado amarelo nas variáveis que estão em uso.

(7) Instruções: A caixa de instruções é de muita importância pois traz todos os recursos do Outseal Studio para criação do diagrama Ladder.

Na seção BASIC é possível adicionar degraus (RUNG) e ramos (BRANCH) Ladder, que são a estrutura básica da linguagem.

Na seção BIT são exibidos os recursos para trabalho com variáveis booleanas, dentre os diferentes tipo de contato estão: NO, normalmente Fechado; NC, normalmente aberto; Para os coils existem as opções: OUT, saída tradicional ladder; e OUTN, saída negada. Ainda no campo BIT existem algumas funções especiais, tais como a OS (One shot), que injeta um pulso na saída para cada borda de subida na entrada.

Nos demais campos são vistos outros blocos típicos da linguagem Ladder, tais como: contadores CTU e CTD, temporizadores TON e TOFF, comparadores e copiadores de variáveis numéricas, somadores, subtratores, entre outros.

(8) Interface de desenvolvimento: Na área central da interface, é possível realizar a implementação do código Ladder. Por meio dos itens no canto esquerdo superior da interface, é possível alterar seu tamanho (símbolo da chave fixa), e criar novas networks (símbolo de soma).

2.4.2 Temporizadores

Os temporizadores no Outseal, embora tenham um comportamento idêntico aos temporizadores tradicionais Ladder, tem um modo de uso um pouco diferente. Os temporizadores no programa são tratados de certa forma como classes, que possuem certos atributos.

Para a definição de um temporizador, a sintaxe que o identifica é do tipo T.k, onde k é um número inteiro no intervalo [1, 100]. A partir daí, os atributos desse temporizador podem ser acessados pelo formato T.k.a, onde 'a' é o atributo desejado.

Na Tabela 3 são mostrados os diferentes atributos de um temporizador do Outseal Studio.

Tabela 3: Atributos do temporizador do Outseal Studio.

Atributo	Definição	Tipo de Dado	Descrição
EN	Enable	Booleano	Indica se o timer está habilitado
TT	Timing	Booleano	Indica se o timer está em contagem
DN	Done	Booleano	Indica se a contagem já terminou
PRESET	Preset	Numérico	Valor final pré-definido para contagem
ACC	Accumulator	Numérico	Valor atual da contagem de tempo

Outro valor importante da configuração do temporizador é a base de tempo, que pode ser definida como 1000ms ou 10ms, o tempo de acionamento do temporizador é o valor do PRESET vezes a base de tempo.

Na Figura 22 é mostrada uma implementação de um oscilador quadrado com duty-cycle de 30% e período de 10s utilizando o TON e seus atributos.

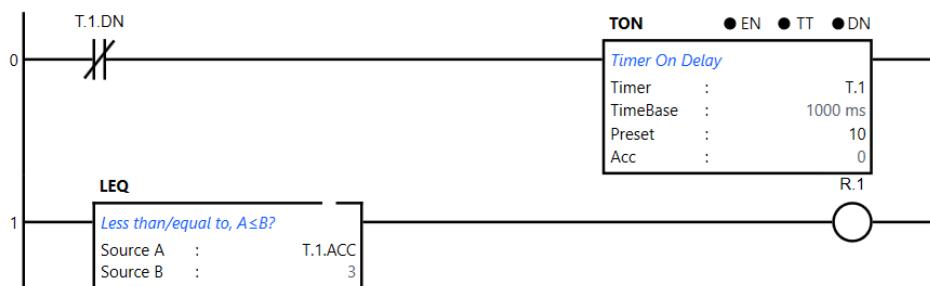


Figura 22: Oscilador com timer no Outseal Studio.

2.4.3 Protocolo Modbus

O uso do protocolo Modbus no Outseal é feito por meio de blocos específicos para cada função do protocolo, esses blocos podem ser encontrados na seção MODBUS da caixa de instruções. Nota-se que neste projeto, o Outseal PLC foi utilizado como cliente da rede e, portanto, as descrições abaixo são relativas a esse tipo de configuração.

Cada bloco Modbus deve ser programado individualmente, recebendo o endereço do servidor a ser acessado, baudrate do canal serial, timeout e os parâmetros da função. Os blocos são identificados pelo formato MF k , onde k é um número inteiro equivalente àquela função (conferir Tabela 2).

Na Figura 27 são mostrados 4 dos blocos mais utilizados: MF2, MF4, MF5 e MF6.

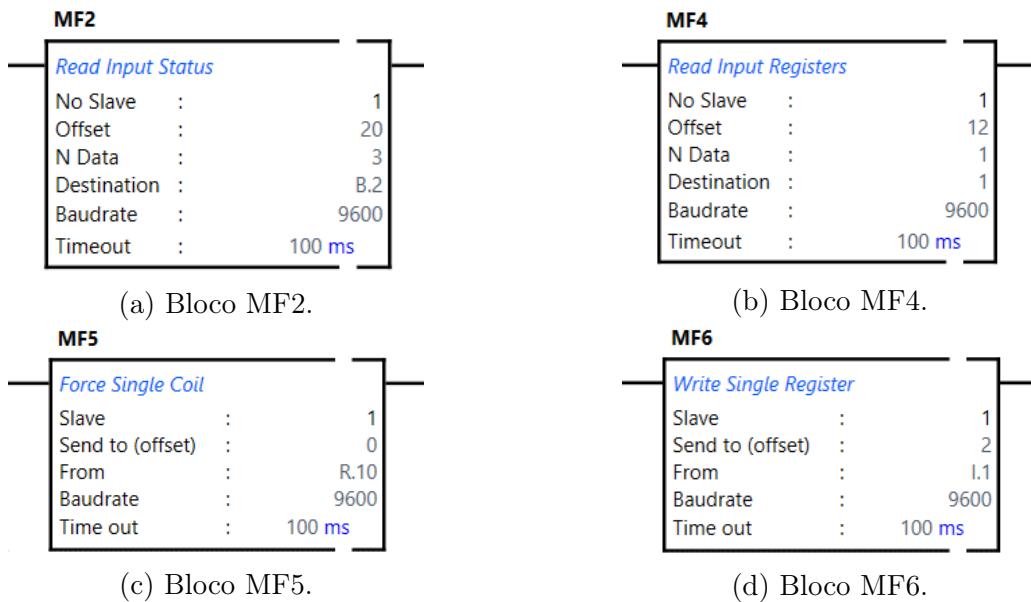


Figura 23: Blocos Modbus do Outseal Studio.

O bloco MF2 (Figura 23a), além do offset e número de dados, precisa da variável a partir da qual os dados lidos serão alocados, que necessariamente precisa ser do tipo Binary, portanto B.k, onde k é um inteiro no intervalo [1, 256]. Se o número de dados for, por exemplo, 3 e a variável dado for B.2, então os valores recebidos serão alocados nas variáveis B.2, B.3 e B.4.

O bloco MF4 (Figura 23b), tem um funcionamento extremamente semelhante ao MF2, mas recebe variáveis numéricas, nomeadamente do tipo I.k, onde k é um inteiro no intervalo [1, 100].

Já as funções de escrita únicas, MF5 e MF6 (respectivamente Figura 23c e Figura 23d), utilizam, além do endereço e offset, a variável cujo valor será enviado via protocolo. No exemplo da Figura 23c o valor do relé R.10 é enviado para o offset 0 do servidor 1. Já no exemplo da Figura 23d, o valor da variável inteira I.1 é enviado ao offset 2 do servidor 1.

Nota-se que o tipo de variável correta deve ser inserido em cada função. Em caso de incerteza do tipo aceito, é possível visualizar as variáveis permitidas com um clique direito sobre o campo em questão.

2.4.4 Controlador PID

O controlador PID nativo do Outseal é composto por dois blocos Ladder: **Set PID Parameters** e **PID**.

O bloco **Set PID Parameters** é usado para realizar as configurações iniciais do controlador, como: set-point; coeficientes k_p , k_i e k_d ; limite do erro integral; limites da resposta de saída e tempo de atualização. Nota-se que esse bloco só precisa ser chamado uma única vez, o que pode ser feito com uso do contato **OS**, já discutido anteriormente.

Esse bloco também requer uma variável inicial, a partir da qual a memória será usada pelo controlador. Por exemplo, caso seja dada a variável I.1 como posição de memória inicial, os endereços de memória de I.1 até I.10 serão alocados para uso do controlador PID, que requer 10 variáveis do tipo inteiro (I).

A entrada de dados é feita na própria variável definida no campo **Memory**; se o valor passado for I.k, o dado de entrada do controlador deve ser copiado para I.k, seja por atribuição direta (bloco MF3 ou MF4), ou por meio de um bloco do tipo **COPY**. Já a saída do controlador é sempre alocada à variável I.(k+9), que pode ser usada diretamente como parâmetro para outros cálculos ou atribuições.

Uma prática comum no PID é a mudança do set-point durante a execução do algoritmo, isso pode ser feito alterando o registrador equivalente ao set-point⁵; para isso, caso a variável de memória inicial seja I.k, deve-se alterar a variável I.(k+1), que é alocada ao set-point, por meio de um bloco do tipo **COPY**.

O segundo bloco do controlador, chamado **PID**, é responsável por executar o algoritmo PID, e deve ser chamado a cada iteração do programa. O campo memória desse bloco deve receber a mesma variável que foi passada ao bloco **Set PID Parameters**; isso pode ser feito com um clique direito sobre o campo e seleção da variável correta.

Na Figura 24 é mostrado um controlador PID devidamente configurado no Outseal Studio.

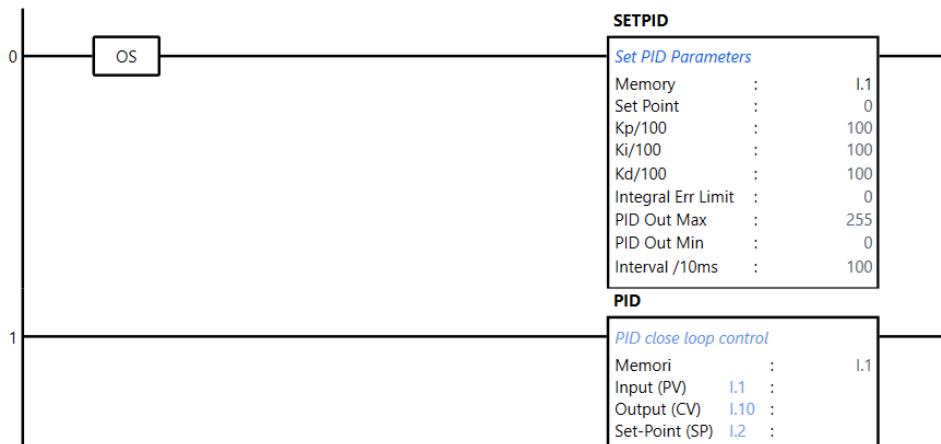


Figura 24: Controlador PID no Outseal Studio.

⁵Embora o Outseal permita que operações sejam feitas em endereços reservados, isso não deve ser feito sem as devidas considerações, pois poderá desconfigurar o controlador PID.

3 Resultados

3.1 Gateway Modbus

A implementação do Gateway Modbus gerou uma biblioteca para uso com a Arduino IDE, e também foram feitos testes de latência no dispositivo.

3.1.1 Biblioteca para Arduino IDE

A biblioteca é baseada em uma classe C++, e possui 3 métodos acessíveis ao usuário.

Objeto Gateway: Classe principal da biblioteca, com a qual é possível realizar a configuração do gateway na rede Ethernet. Para inicializar a classe, é possível inserir dois parâmetros.

- **_ip:** **Tipo:** *IPAddress; **Descrição:** Endereço IP do gateway na rede Ethernet;
- **_subnet:** **Tipo:** *IPAddress; **Descrição:** Máscara de subrede da rede Ethernet.

Caso o endereço IP ou a máscara de sub-rede não sejam especificados, o gateway é acionado em modo de IP dinâmico. Caso os valores sejam especificados, eles serão usados pelo gateway, que trabalhará com IP fixo.

No Código 1 é mostrada a inicialização de um objeto Gateway com IP fixo.

```

1 IPAddress ip(192, 168, 0, 1);
2 IPAddress subnet(255, 255, 255, 0);
3
4 Gateway mb(&ip, &subnet);
```

Código 1: Inicialização do objeto Gateway com IP fixo.

Método configTCP: o método configTCP permite realizar a configuração dos parâmetros para o protocolo Modbus TCP/IP. Esse método conta com 4 parâmetros, sendo 1 obrigatório e 3 opcionais.

Os parâmetros do método são:

- **_serverIP:** **Tipo:** IPAddress; **Descrição:** Endereço IP do servidor Modbus a ser acessado pelo gateway;
- **_serverPort:** **Tipo:** uint32_t; **Valor padrão:** 502; **Descrição:** Porta UDP do servidor Modbus a ser acessado;
- **timeout:** **Tipo:** uint16_t; **Valor padrão:** 500; **Descrição:** Tempo de espera por resposta em milissegundos (ms);
- **tryToConnect:** **Tipo:** uint8_t; **Valor padrão:** KEEP_TRYING; **Descrição:** Quantidade de tentativas de conexão ao servidor.

No Código 2 é mostrado um exemplo de configuração do método onde são passados todos os parâmetros.

```
1 mb.configTCP(IPAddress(192, 168, 0, 10), 502, 500, KEEP_TRYING);
```

Código 2: Configuração do protocolo TCP/IP.

Método configRTU: O método `configRTU` permite especificar os parâmetros do protocolo Modbus RTU. Esse método possui 3 parâmetros, 2 obrigatórios e 1 opcional. Os parâmetros são:

- `_serial`: **Tipo:** *HardwareSerial; **Descrição:** USART do Arduino a ser usada como interface para o canal Modbus RTU;
- `baudrate`: **Tipo:** uint32_t; **Descrição:** Baudrate do canal Modbus RTU;
- `serialConfig`: **Tipo:** uint8_t; **Valor padrão:** SERIAL_8N1; **Descrição:** Parâmetros do canal serial - bits por dados, paridade e bit de paragem.

No Código 3 é mostrado um exemplo de configuração do gateway onde foi usada a USART 0 como interface para o Modbus RTU.

```
1 mb.configRTU(&Serial, 38400, SERIAL_8N1);
```

Código 3: Configuração do protocolo RTU.

Método run : O método `run` é que realiza a varredura pelos pacotes Modbus e, quando os encontra, faz sua validação e conversão entre protocolos. Uma vez que o gateway foi devidamente inicializado por meio dos métodos `configTCP` e `configRTU`, o método `run` deve ser chamado em um loop infinito para que o gateway funcione corretamente.

No Código 4 é mostrada uma implementação da biblioteca `Gateway` na Arduino IDE.

```
1 #include <Gateway.h>
2
3 IPAddress ip(192, 168, 0, 1);
4 IPAddress subnet(255, 255, 255, 0);
5
6 Gateway mb(&ip, &subnet);
7
8 void setup() {
9     mb.configTCP(IPAddress(192, 168, 0, 10), 502, 500, KEEP_TRYING)
10    mb.configRTU(&Serial, 38400, SERIAL_8N1);
11 }
12
13 void loop() {
14     mb.run();
15 }
```

Código 4: Implementação da biblioteca `Gateway.h`.

3.1.2 Latência do Gateway

Os resultados das medições de latência, assim como o aumento na latência do canal com a inserção do Gateway, são mostrados na Tabela 4.

Tabela 4: Medições de latência do Gateway.

Baudrate [bps]	t_0 [ms]	t_S [ms]	$t_{RTU,TCP}$ [ms]	$t_{TCP,RTU}$ [μs]	Aumento de Latência [%]
9600	$18,7 \pm 2,8$	$6,6 \pm 2,8$	$11,581 \pm 0,014$	520 ± 11	182,4
19200	$13,0 \pm 2,9$	$6,4 \pm 2,9$	$6,088 \pm 0,005$	522 ± 13	103,3
38400	$10,9 \pm 3,0$	$6,1 \pm 2,9$	$4,255 \pm 0,499$	520 ± 8	78,5
57600	$11,1 \pm 3,1$	$9,5 \pm 3,1$	$4,100 \pm 0,051$	520 ± 19	17,4
115200	$10,9 \pm 3,1$	$6,3 \pm 3,1$	$4,090 \pm 0,002$	522 ± 4	73,0

*Referências das variáveis são exibidas na Subsubseção 2.2.4.

3.2 Aplicação I: Processo de Seleção de Pacotes.

A cena desenvolvida no Factory I/O para a aplicação I é mostrada na Figura 25.

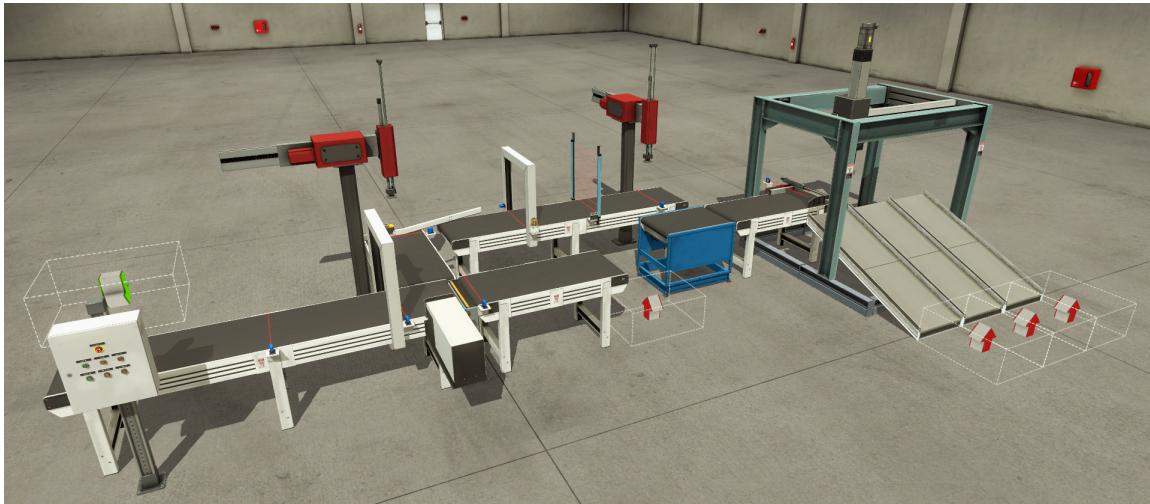


Figura 25: Cena do Factory I/O da aplicação I.

Além da planta produtiva, a cena também recebeu um painel de controle, com botões de Start(Verde), Reset(Amarelo) e Stop(Vermelho) e luzes que indicam essas três funções. O painel também conta com um botão de emergência, que, quando pressionado, faz uma paragem total da planta, sendo necessário reiniciá-la (pressionar o botão Reset) para retomar o processo. O painel montado é exibido na Figura 26.



Figura 26: Painel de controle da aplicação I.

3.3 Aplicação II: Controle de Nível.

A cena desenvolvida no Factory I/O para a aplicação II é a própria estação "Tank" do Factory I/O, mostrada na Figura 27a, mas também foi adicionado um painel de controle para a planta, mostrado na Figura 27b.



(a) Cena.



(b) Painel de Controle.

Figura 27: Cena do Factory I/O da aplicação II.

O painel da aplicação II conta com uma chave seletora para ativar/desativar o controlador de nível. Três displays numéricos também foram colocados, suas funções são: mostrar o nível atual de água; mostrar o nível desejado de água; e mostrar a abertura da válvula de escape. A partir do painel, é possível controlar a abertura da válvula de escape pelo potenciômetro da direita; e também selecionar o nível de água desejado a partir do potenciômetro da esquerda.

3.4 Arquivos do Projeto

Por conta das aplicações serem arquivos de simulação e diagramas Ladder, sua apresentação via documento escrito é limitada. Por esse motivo, os arquivos gerados no projeto, incluindo: cenas do Factory I/O, arquivos Ladder do Outseal Studio, e a biblioteca Arduino para uso do Gateway Modbus; foram adicionados a um repositório no GitHub. Por meio do repositório é possível acessar todos os arquivos, e fazer seu download para realização de testes e simulações.

O repositório foi hospedado no endereço: <https://github.com/HadlerHT/Projeto-Simulador-FactoryIO>.

Em especial para aplicação I, foi feito um breve vídeo do processo em funcionamento, que foi então hospedado na plataforma YouTube. O vídeo pode ser acessado no link: https://youtu.be/909P_Rwogds.

4 Discussões

Gateway Modbus O desenvolvimento de um Gateway Modbus, embora não tenha sido previsto na proposta inicial do projeto, tornou-se uma parcela significativa da execução do mesmo. Como visto ao longo do trabalho, esse foi um componente indispensável para estabelecimento de um canal de comunicação entre as partes estudadas.

No entanto, como exibido na seção anterior, a adição desse componente no canal de comunicação acabou por aumentar significadamente a latência, isso é, o tempo de resposta do canal. Em relação ao tempo que seria gasto sem o Gateway na rede, o valor do tempo de resposta foi aumentado, no melhor caso, cerca de 17.4%, enquanto que o pior caso teve um aumento de cerca de 182.4%. Para as aplicações desse projeto, embora o aumento tenha sido considerável em alguns casos, não causou efeitos negativos na rede, pois a magnitude da latência total permaneceu na ordem de 10ms.

Essa etapa do projeto acabou por despertar algumas perguntas sobre o Gateway Modbus, tais como: como otimizar o algoritmo de conversão? E qual o hardware mais adequado para sua implementação? Essas perguntas abriram a possibilidade de um futura pesquisa focada nesse tópico.

Aplicações As aplicações mostradas no tópico anterior satisfizeram as propostas iniciais do projeto, ao criar ambientes preparados pra a introdução ao ensino de supervisão e controle de processos industriais.

A aplicação I, que foi baseada em um módulo real disponível aos estudantes do curso técnico em mecatrônica do CEFET-MG Varginha, permitirá a criação de um raciocínio lógico pra implementação do algorítimo de controle por meio de simulação. Esse conhecimento prévio dos princípios de funcionamento da planta por meio do simulador, podem auxiliar nas aulas em laboratório com a planta real.

A aplicação II, baseada no controle de nível com controlador PID, permite que os alunos do curso técnico observem esse controlador num sistema dinâmico. Com a variação dos parâmetros do controlador, e observação dos resultados na simulação, é gerado um ambiente onde o(a) aluno(a) cria uma intuição dos efeitos dos módulos P, I e D no sistema, se familiarizando com o controlador.

5 Conclusão

O Factory I/O se mostrou uma ferramenta excelente para criação de cenas industriais. Com uma quantidade grande de partes para montagem dos processos e uma documentação extremamente bem escrita e detalhada, é possível introduzi-lo a um(a) aluno(a) de forma rápida e com clareza. As partes presentes no simulador se aproximam bastante de componentes reais usados no setor industrial e, com seu uso no simulador, é possível desenvolver a capacidade lógica necessária para programação desses dispositivos.

O Outseal Studio, embora ainda em fase de desenvolvimento, é um software com muito potencial, permitindo implementação de processos bastante complexos em linguagem Ladder, isso graças às suas várias funcionalidades já implementadas. No contexto do ensino de controle, o Outseal se destacou por permitir o uso da linguagem Ladder em controladores Arduino, o que permite a exposição de alunos à programação de CLPs com um custo relativamente reduzido.

A Shield Ethernet W5100 - Erro de Fabricação

Alguns lotes do shield Ethernet W5100 foram fabricados com um erro no hardware, trata-se do resistor que determina a impedância de saída da porta Ethernet, que foi usado com o valor errado.

Segundo o esquemático do fabricante, o resistor em questão deveria ter uma resistência de 49.9Ω , no entanto, nesses lotes defeituosos, foram soldados resistores de 510Ω . A impedância de saída incorreta causa inúmeros problemas durante a comunicação, tornando o equipamento inutilizável.

Na Figura 28 é mostrado o resistor de 510Ω em um shield defeituoso.

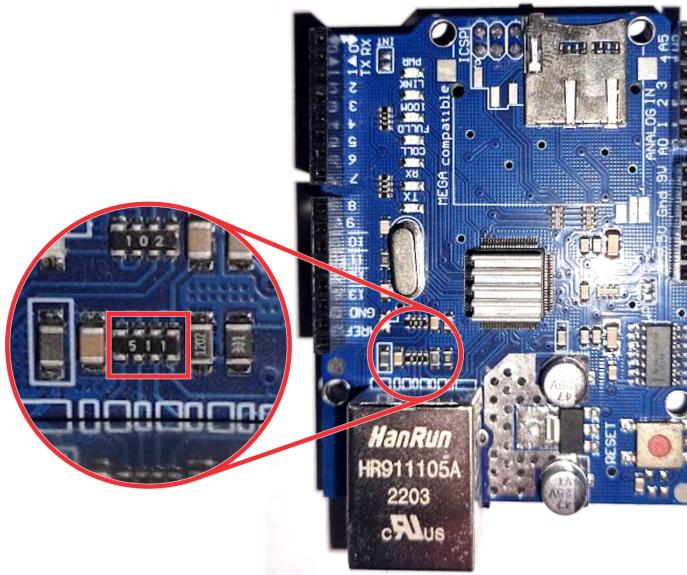


Figura 28: Resistor errado no shield Ethernet W5100.

Existe no entanto uma solução simples para o problema. A impedância característica correta para o canal Ethernet é de $100 \pm 10\% \Omega$, enquanto que o valor nominal no shield defeituoso é de 1020Ω . Portanto, é possível atingir o valor correto a partir da colocação de um resistor externo em paralelo ao circuito já existente, reduzindo o valor de impedância para mais próximo do ideal.

Na Figura 29a é mostrado o circuito de entrada do canal de recebimento de dados do RJ45 do shield defeituoso, pode-se observar que a impedância DC nominal é de $Z_{DC} = 1020\Omega$. Colocando uma resistência de valor R em paralelo com a porta de entrada, conclui-se que a nova de impedância de entrada será o valor 1020Ω em paralelo com R.

Como se deseja um $Z_{DC} = 100\Omega$, calcula-se o valor de R.

$$100\Omega = \frac{1020\Omega \cdot R}{1020\Omega + R} \quad \rightarrow \quad R \approx 110.9\Omega$$

Como R é cerca de 110Ω , foi utilizado o valor comercial mais próximo: 100Ω . O circuito corrigido é mostrado na Figura 29b. Ressalta-se que, na figura, é mostrado apenas o canal de recebimento de dados do RJ45, e que o mesmo processo deve ser feito para o canal de transmissão.

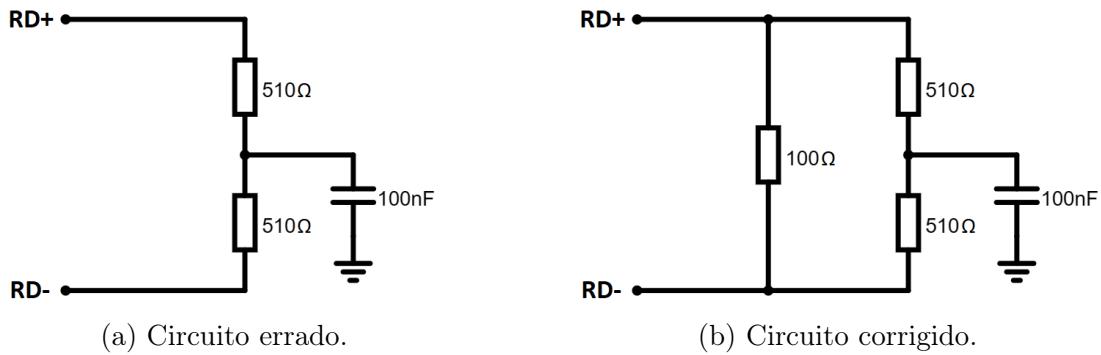


Figura 29: Circuito de entrada do conector RJ45 do shield W5100.

A soldagem desse resistor no shield é um processo simples: utilizando resistores discretos through-hole, é possível fazer a soldagem desses na parte inferior do shield, diretamente nos pinos do conector RJ45.

A posição dos resistores no shield é mostrada na Figura 30

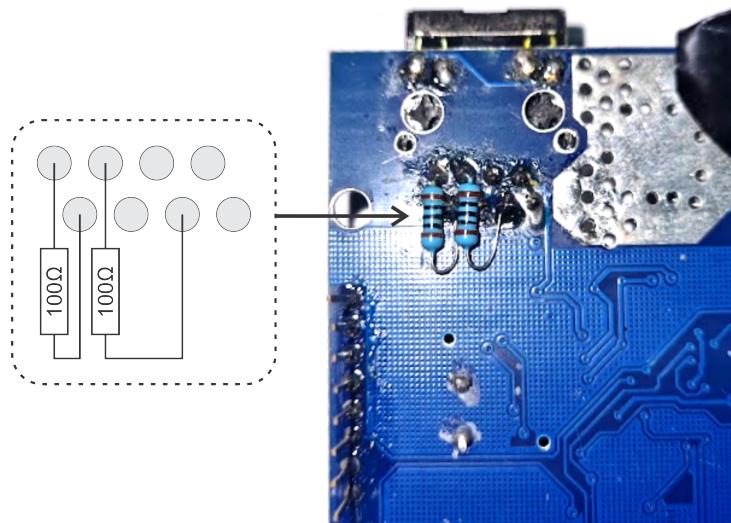


Figura 30: Resistores de correção de impedância soldados no shield W5100.

Com essa correção, é possível utilizar o shield normalmente.

Referências

- [1] **REAL GAMES**: *Changing the way people learn.* Disponível em: <https://realgames.co/company/>. Acesso em: 08 de fev. de 2023.
- [2] **OUTSEAL**. Disponível em: <https://outseal.com/site/index.html>. Acesso em: 08 de fev. de 2023.
- [3] **MODBUS**: *About the protocol.* Disponível em: <https://modbus.org/faq.php>. 2023. Acesso em: 08 de fev. de 2023.