



CURSO ENGENHARIA DE CONTROLE AUTOMAÇÃO

Raquel Ururahy Lopes

Graduando

Engenharia Controle e Automação

ururahy@ucl.br

Saulo B. Gelinski

Graduando

Engenharia de Controle e Automação

saulobgelinski@ucl.br

MANUAL DE PARAMETRIZAÇÃO

SERRA-ES

2023

SUMÁRIO

1. Modbus Configuração	2
1.1. Configuração Modbus RTU	2
1.2. Tempo de resposta mestre RTU	3
1.3. Tempo de resposta escravo RTU	3
1.4. Configuração de canais I/Os	4
1.5. Configuração Modbus/TCP	5
1.5.1. Configuração da rede	5
1.5.2. Configuração de parâmetros	5
1.5.3. Configuração de endereços das I/Os no Arduino.....	6
1.5.4. Configuração de comunicação Ethernet no Elipse	7
1.6. Parametrização Numérica	9
1.7. Monitoramento de variáveis.....	9
1.8. Erros de Comunicação	13
2. Esquema de Ligação.....	13

MANUAL DE PARAMETRIZAÇÃO

1. Modbus Configuração

O Modbus RTU é um protocolo de comunicação serial amplamente utilizado em sistemas SCADA (Supervisory Control and Data Acquisition) e CLP (Controladores Lógicos Programáveis). Por outro lado, o Modbus TCP é uma versão mais moderna e amplamente utilizada em redes Ethernet.

Para a configuração e programação foi utilizado o software Codesys V3 na versão 3.5.19.40.

Na figura 1, a estrutura de rede Modbus denota a comunicação RTU e TCP, onde o Codesys é o mestre e os escravos são o Elipse e o Arduino para cada modo de comunicação. O Modbus RTU é usado para comunicar o Codesys com o Arduino, e no Modbus TCP via Ethernet, o Codesys comunica com o Elipse.

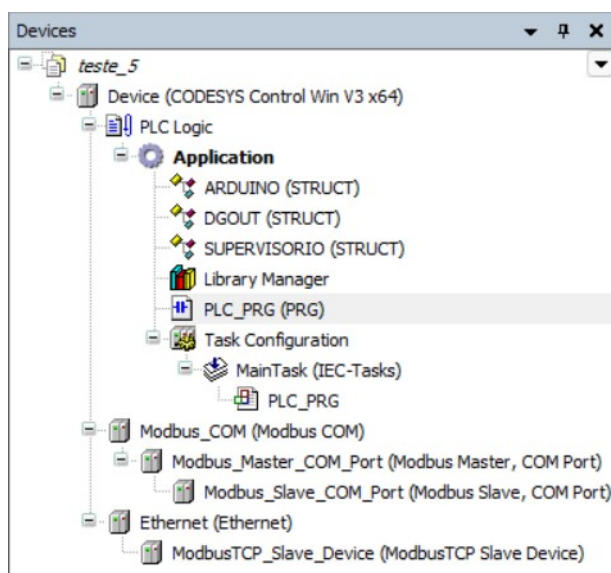


Figura 1 – Estrutura da rede Modbus

1.1. Configuração Modbus RTU

Para configurar a porta serial do programa, basta ir na estrutura de rede representada na figura 1 e clicar 2 vezes na subcategoria “Device”, em “Modbus COM”. No instante seguinte abrirá uma janela, clique em General, em seguida aparecerá a configuração mostrada na figura 2.

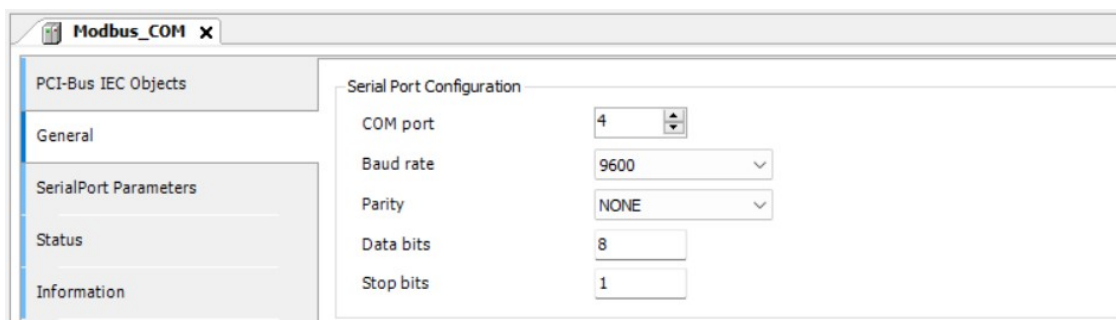


Figura 2 – Configuração porta serial

1.2. Tempo de resposta mestre RTU

Nesse campo é feita a seleção do protocolo RTU e o tempo limite de resposta do mestre em relação aos escravos, e após este período, uma nova requisição do mestre para os escravos é feita.

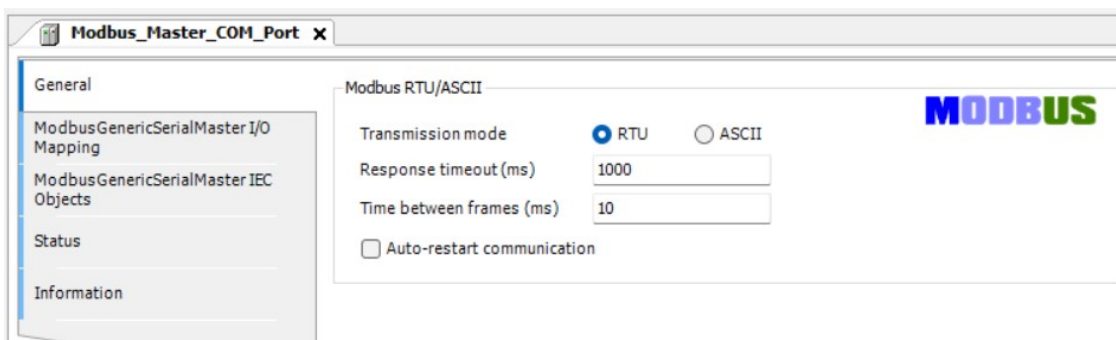


Figura 3 – Tempo de resposta mestre RTU

1.3. Tempo de resposta escravo RTU

Nesse campo é feita a seleção do endereço do escravo e também do tempo limite de resposta do escravo à solicitação do mestre, esse tempo deve ser suficiente para o escravo receber a solicitação e processar o pedido para depois enviar a resposta.

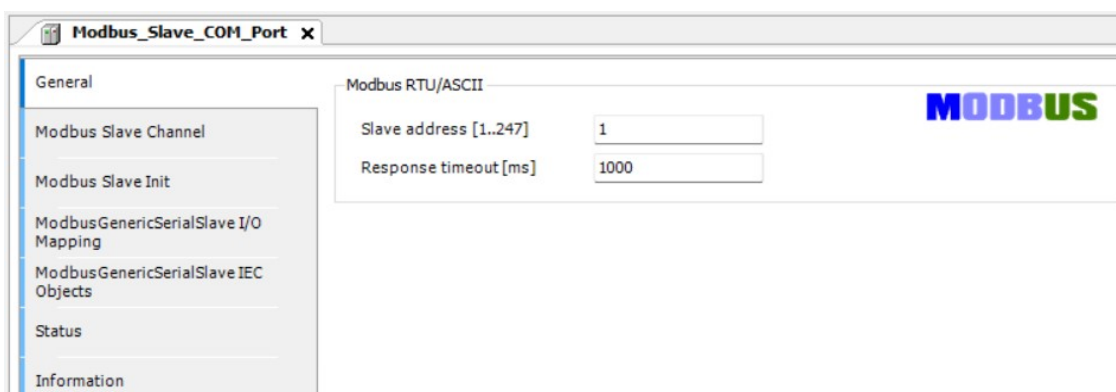


Figura 4 - Tempo de resposta escravo RTU

1.4. Configuração de canais I/Os

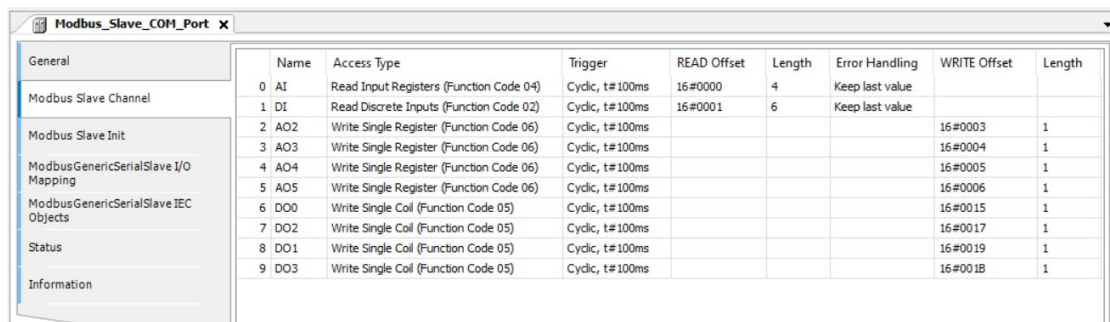
Nesta etapa declaramos os endereços de memória reservados para as variáveis de transmissão, fazendo a leitura de um valor no registrador do escravo de rede e escrita em outro como demonstrado na Figura 5. Na Figura 6 abaixo estão as funções Modbus para leitura e escrita destes dados no dispositivo escravo.

A versão Demo do Codesys utilizado neste projeto, só permite a criação de 10 canais de I/Os, por esse empecilho foi elaborado uma estratégia para alocação das I/Os. Nesta simulação do Codesys, não estão presentes todas as entradas e saídas analógicas e digitais do projeto físico.

As entradas digitais e analógicas foram agrupadas em um canal específico para cada tipo, possuindo as entradas analógicas, quatro espaços e as digitais com seis espaços.

Para as saídas digitais e analógicas foram utilizados um canal por saída, representando no total quatro espaços para saídas analógicas e quatro para saídas digitais. Todas as saídas são do tipo “escrita”, com a diferença que as analógicas são “Register” e as digitais “Coil”.

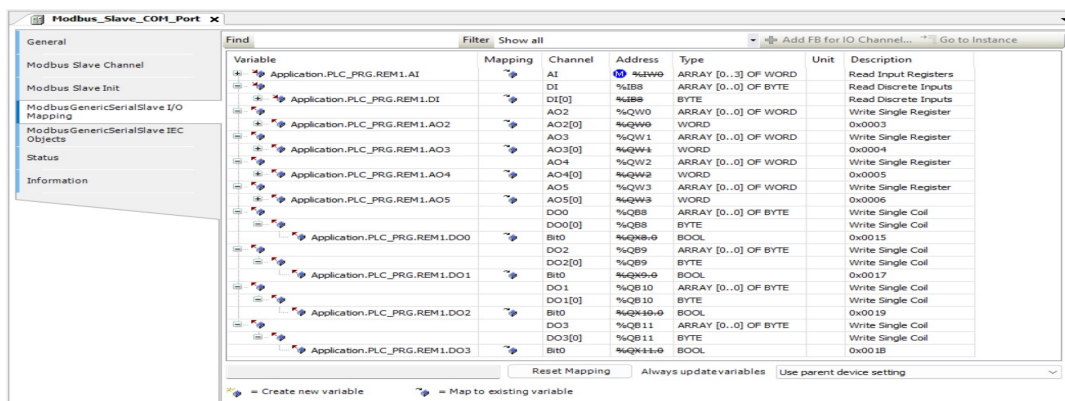
Levar em consideração a diferença entre os Register e Coils, pois as saídas do tipo Coil ficam em endereços separados.



Name	Access Type	Trigger	READ Offset	Length	Error Handling	WRITE Offset	Length
0 AI	Read Input Registers (Function Code 04)	Cyclic, t#100ms	16#0000	4	Keep last value		
1 DI	Read Discrete Inputs (Function Code 02)	Cyclic, t#100ms	16#0001	6	Keep last value		
2 AO2	Write Single Register (Function Code 06)	Cyclic, t#100ms				16#0003	1
3 AO3	Write Single Register (Function Code 06)	Cyclic, t#100ms				16#0004	1
4 AO4	Write Single Register (Function Code 06)	Cyclic, t#100ms				16#0005	1
5 AO5	Write Single Register (Function Code 06)	Cyclic, t#100ms				16#0006	1
6 DO0	Write Single Coil (Function Code 05)	Cyclic, t#100ms				16#0015	1
7 DO2	Write Single Coil (Function Code 05)	Cyclic, t#100ms				16#0017	1
8 DO1	Write Single Coil (Function Code 05)	Cyclic, t#100ms				16#0019	1
9 DO3	Write Single Coil (Function Code 05)	Cyclic, t#100ms				16#001B	1

Figura 5 – Canais de I/Os

Na figura 6 está representado o mapeamento das I/Os do escravo RTU, que são criadas automaticamente logo após as declarações dos canais, nelas pode-se fazer o monitoramento das variáveis ao rodar o programa.



Variable	Mapping	Channel	Address	Type	Unit	Description
Application.PLC_PRG.REM1.AI		AI	%I0	ARRAY [0..3] OF WORD		Read Input Registers
Application.PLC_PRG.REM1.DI		DI	%I1	ARRAY [0..0] OF BYTE		Read Discrete Inputs
Application.PLC_PRG.REM1.AO2		AO2	%Q0	ARRAY [0..0] OF WORD		Write Single Register
Application.PLC_PRG.REM1.AO3		AO3	%Q1	ARRAY [0..0] OF WORD		Write Single Register
Application.PLC_PRG.REM1.AO4		AO4	%Q2	ARRAY [0..0] OF WORD		Write Single Register
Application.PLC_PRG.REM1.AO5		AO5	%Q3	ARRAY [0..0] OF WORD		Write Single Register
Application.PLC_PRG.REM1.DO0		DO0	%Q4	WORD		Write Single Register
Application.PLC_PRG.REM1.DO1		DO1	%Q5	WORD		Write Single Register
Application.PLC_PRG.REM1.DO2		DO2	%Q6	WORD		Write Single Register
Application.PLC_PRG.REM1.DO3		DO3	%Q7	WORD		Write Single Register

Figura 6 – Mapeamentos de I/Os escravos

1.5. Configuração Modbus/TCP

1.5.1. Configuração da rede

Cada dispositivo escravo deve ser configurado com endereço único, e estar dentro da faixa de endereços IPv4 permitida, caso contrário, a rede não completará um enlace. Na Figura 7 são definidas as configurações do dispositivo mestre, selecionando a interface para configurar o endereço IPv4 e o restante necessário.

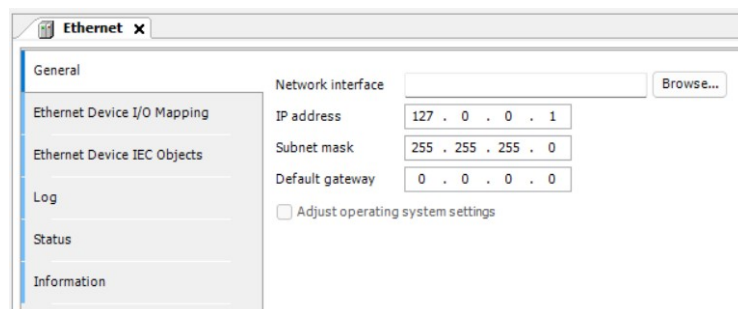


Figura 7 – Configuração de rede no mestre Modbus TCP

Realizada esta etapa, configura-se a rede para o dispositivo escravo. Cada um deve possuir seu endereço próprio e estar dentro do intervalo de endereços IPv4 disponíveis, o restante das configurações, como taxa de transmissão, paridade, bits de dados e bits de parada serão gerenciadas automaticamente pela rede.

1.5.2. Configuração de parâmetros

Após definidos os canais que são demonstrados na figura 5, é necessário configurar o número de armazenamento de cada variável.

Holding Registers – Saídas analógicas

Input Registers – Entradas analógicas

Coils – Saídas digitais

Discrete Inputs – Entradas digitais

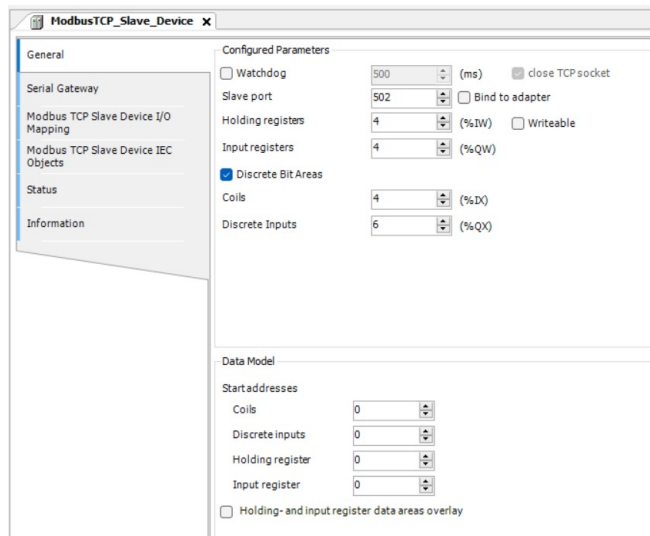


Figura 8 – Parâmetros dos canais

Na figura 9 está representado o mapeamento das I/Os do mestre, nelas pode-se fazer o monitoramento das variáveis ao rodar o programa.

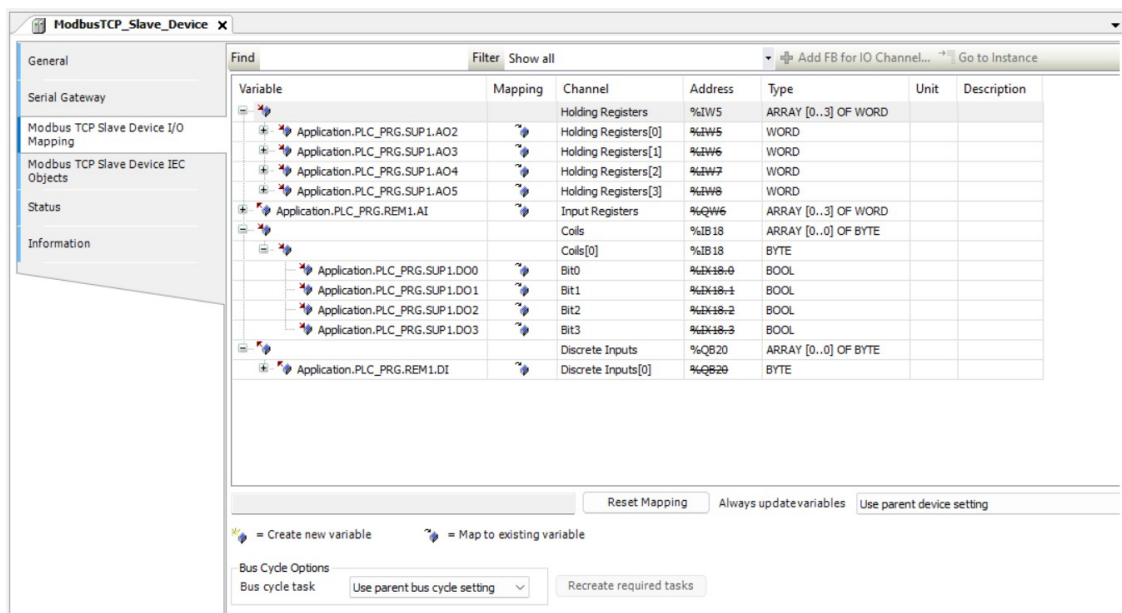


Figura 9 – Mapeamentos de I/Os mestre

1.5.3. Configuração de endereços das I/Os no Arduino

Abaixo estão descritos os endereços das I/Os no código do Arduino, onde definimos todos os pinos que serão utilizados fisicamente para ler ou escrever valores nos registradores, compatíveis com o protocolo Modbus.

```

//Set Slave ID
regBank.setId(1); //Set Slave ID

//Add Digital Output registers
regBank.add(28);
regBank.add(26);
regBank.add(24);
regBank.add(22);

//Add Digital Input registers
regBank.add(10002);
regBank.add(10003);
regBank.add(10004);
regBank.add(10005);
regBank.add(10006);
regBank.add(10007);

//Analog Input registers
regBank.add(30001);
regBank.add(30002);
regBank.add(30003);
regBank.add(30004);

//Analog Output registers
regBank.add(40002);
regBank.add(40003);
regBank.add(40004);
regBank.add(40005);
regBank.add(40006);
regBank.add(40007);

```

Figura 10 – Código Arduino, declaração das variáveis

1.5.4. Configuração de comunicação Ethernet no Elipse

Passo a passo de como configurar um drive de comunicação Ethernet em modo escravo no Elipse E3.

Passo 1

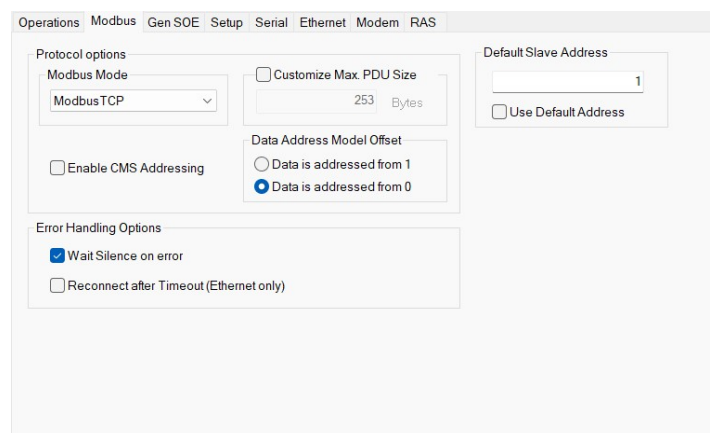


Figura 11 – Configuração Modbus TCP, guia “Modbus”.

Passo 2

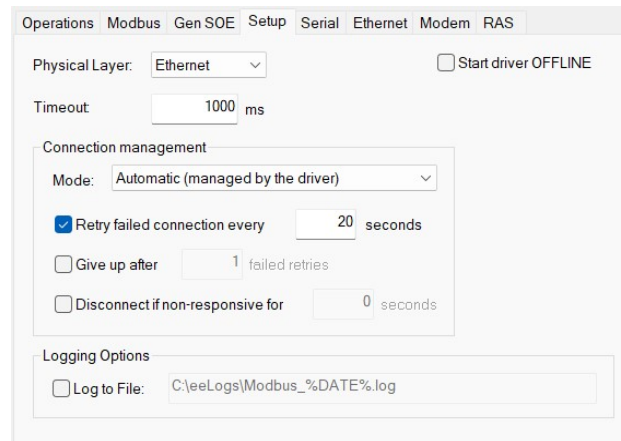


Figura 12 – Configuração Modbus TCP, guia “Setup”.

Passo 3

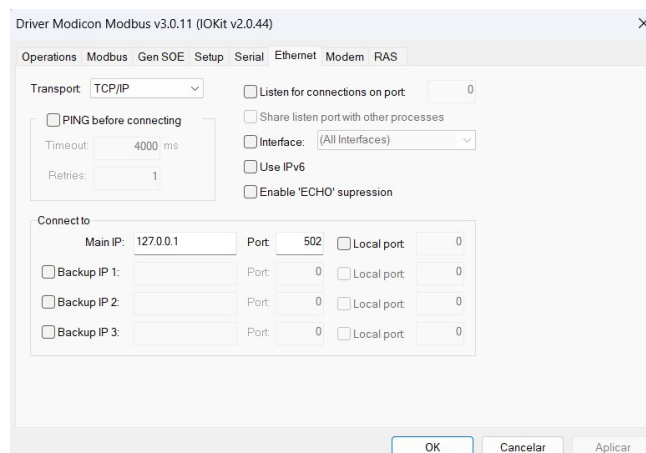


Figura 13 – Configuração Modbus TCP, guia “Ethernet”.

Passo 4 – Configuração dos atributos de cada I/O para leitura e escrita

Oper.	Read	Write	Data	Size
01	01	05	bit	00
02	02	None	bit	00
03	03	06	word	02
04	04	None	word	02

Figura 14 – Configuração dos operadores Elipse

A lista abaixo representa a relação de I/Os no Driver Elipse E3

Nome	P1/N1/B1	P2/N2/B2	P3/N3/B3	P4/N4/B4	...	Varredura	Leitura?	Escrita?	Escala?
Drive1	0	0	0	0					
• Ard_Pot	1	4	0	0		100	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
• Ard_Pot_2	1	4	0	1		100	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
• Ard_Pot_3	1	4	0	2		100	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
• Ard_Pot_4	1	4	0	3		100	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
• Ard_PushButton_0	1	2	0	0		100	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
• Ard_PushButton_1	1	2	0	1		100	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
• Ard_PushButton_2	1	2	0	2		100	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
• Ard_PushButton_3	1	2	0	3		100	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
• Ard_PushButton_4	1	2	0	4		100	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
• Ard_PushButton_5	1	2	0	5		100	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
• LedVd_Ard_1	1	1	0	0		100	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
• LedVd_Ard_2	1	1	0	1		100	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
• LedVd_Ard_3	1	1	0	2		100	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
• LedVd_Ard_4	1	1	0	3		100	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
• Led_Verm_Ard_1	1	3	0	0		100	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
• Led_Verm_Ard_2	1	3	0	1		100	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
• Led_Verm_Ard_3	1	3	0	2		100	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
• Led_Verm_Ard_4	1	3	0	3		100	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>

Figura 15 – Configuração de I/Os no Elipse

1.6. Parametrização Numérica

N1/B1: Endereço do equipamento escravo na rede (Slave ID).

N2/B2: Código da operação.

- 1- Saídas digitais
- 2- Entradas digitais
- 3- Saídas analógicas
- 4- Entradas analógicas

N3/B3: Parâmetro adicional. Normalmente deixado em “0”, só é usado em 4 situações.

N4/B4: Endereço do registrador, variável ou bit no equipamento escravo (pino do Arduino).

1.7. Monitoramento de variáveis

Na figura 16 podemos ver como fica a estrutura Modbus em modo simulação sem apresentar falhas de comunicação.

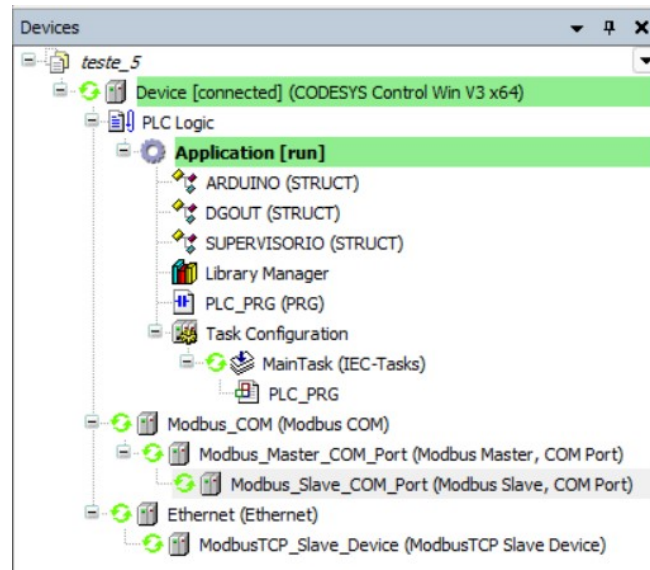


Figura 16 – Estrutura Modbus em modo simulação

Após a configuração da rede Modbus/TCP e declarados os endereços de transmissão, podemos monitorar e controlar os dados transmitidos pelos dispositivos. Por fim, devemos declarar as variáveis que desejamos atribuir aos endereços, no modo *online*, na aba **Mapping**, é possível verificar e escrever os valores nas variáveis de rede.

A seguir, o mapeamento das entradas e saídas das comunicações RTU e TCP online.

Modbus_Slave_COM_Port x						
Find Filter Show all Add FB for IO Channel...						
Variable	Mapping	Channel	Address	Type	Curr	
Application.PLC_PRG.REM1.AI		A1	%I0	ARRAY [0..3] OF WORD	[0,0,89,138]	
Application.PLC_PRG.REM1.DI		DI	%I8	ARRAY [0..0] OF BYTE	Only subelements up: 0	
		DI[0]	%I8	BYTE	0	
		Bit0	%I8-0	BOOL	FALSE	
		Bit1	%I8-1	BOOL	FALSE	
		Bit2	%I8-2	BOOL	FALSE	
		Bit3	%I8-3	BOOL	FALSE	
		Bit4	%I8-4	BOOL	FALSE	
		Bit5	%I8-5	BOOL	FALSE	
		AO2	%QW0	ARRAY [0..0] OF WORD	Only subelements up: 0	
		AO2[0]	%QW0	WORD	0	
		AO3	%QW1	ARRAY [0..0] OF WORD	Only subelements up: 0	
		AO3[0]	%QW1	WORD	0	
		AO4	%QW2	ARRAY [0..0] OF WORD	Only subelements up: 0	
		AO4[0]	%QW2	WORD	0	
		AO5	%QW3	ARRAY [0..0] OF WORD	Only subelements up: 0	
		AO5[0]	%QW3	WORD	0	
		DO0	%QB8	ARRAY [0..0] OF BYTE	Only subelements up: 0	
		DO0[0]	%QB8	BYTE	Only subelements up: 0	
		Bit0	%QX8-0	BOOL	FALSE	
		DO2	%QB9	ARRAY [0..0] OF BYTE	Only subelements up: 0	
		DO2[0]	%QB9	BYTE	Only subelements up: 0	
		Bit0	%QX9-0	BOOL	FALSE	
		DO1	%QB10	ARRAY [0..0] OF BYTE	Only subelements up: 0	
		DO1[0]	%QB10	BYTE	Only subelements up: 0	
		Bit0	%QX10-0	BOOL	FALSE	
		DO3	%QB11	ARRAY [0..0] OF BYTE	Only subelements up: 0	
		DO3[0]	%QB11	BYTE	Only subelements up: 0	
		Bit0	%QX11-0	BOOL	FALSE	

Figura 17 - Mapeamento I/Os Modbus RTU

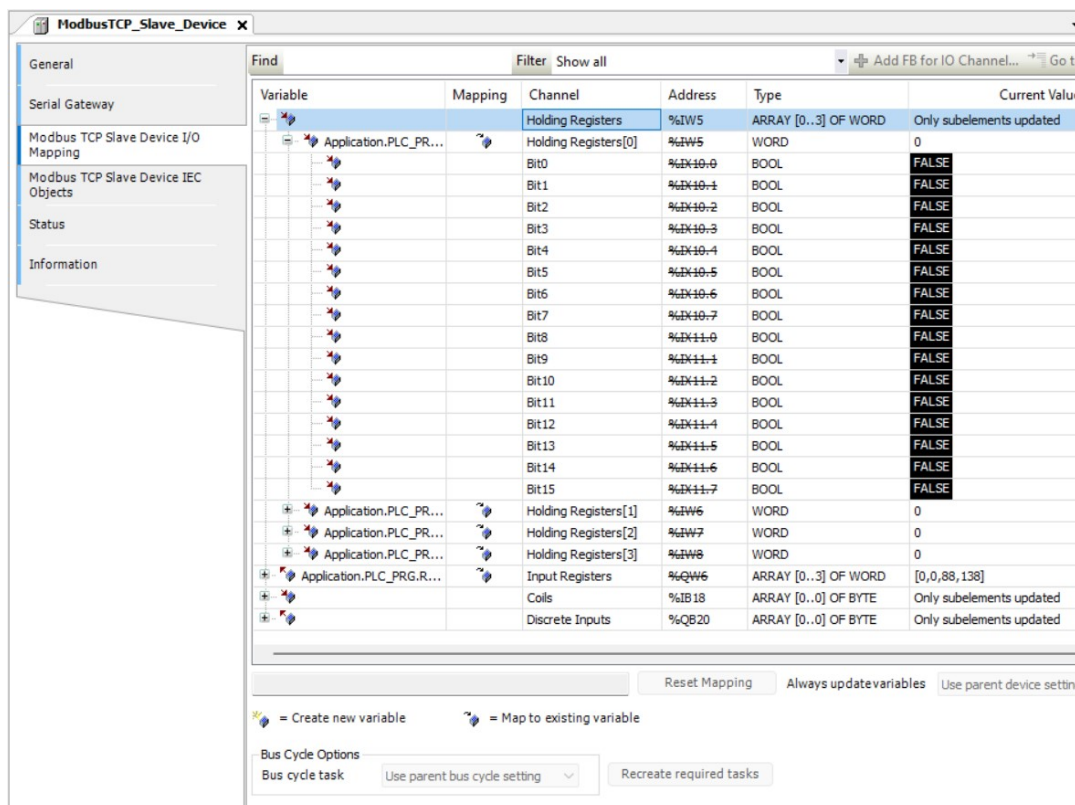


Figura 18 – Mapeamento I/Os Modbus TCP

Na imagem da Figura 19 está apresentado o supervisório em modo simulação.

Foi elaborada uma interface no software Eclipse E3 para realizar a simulação e acompanhamento das entradas e saídas digitais e analógicas, num ambiente virtual, como exemplificado na figura.

A operação da tela do supervisório destina-se à validação das I/Os e foi organizada da seguinte maneira.

Saídas digitais: O acionamento da saída física é feito ao pressionar o botão verde e seu led vermelho correspondente acende, ambos na tela do supervisório. E o oposto ocorre ao pressionar o botão vermelho também no supervisório, desativando a saída física na placa Arduino.

Entradas digitais: Ao ativar uma das entradas digitais físicas, o estado na tela do supervisório será alterado de “desligado” para “ligado”, e ao desativar a entrada física, o oposto também ocorre.

Saídas analógicas: Ao mover o cursor do “Slider” virtual no supervisório, pode-se excursionar um valor inteiro de 8 bits (0 a 255) que se reflete em variação proporcional na largura de pulso do sinal PWM, para controlar cargas nos níveis de potência, velocidade ou brilho desejados.

Entradas analógicas: Visualiza um valor inteiro com range de 10 bits (0 a 1023) que varia proporcionalmente com a tensão aplicada à sua entrada, compreendida entre 0 e 5V ou 0 a 10V, dependendo da posição do jumper de seleção na placa.

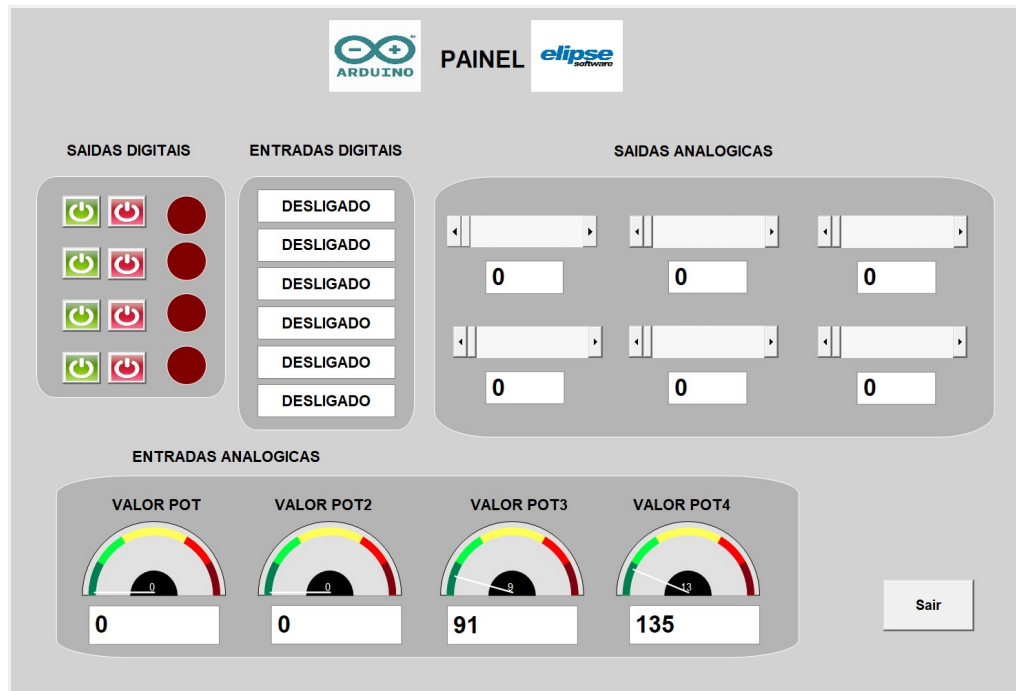


Figura 19 – Supervisório Elipse

Na Figura 20 o Drive está em modo simulação, dessa maneira é possível visualizar se as portas estão funcionando corretamente na aba (Valor).

Nome	P1/N1/B1	P2/N2/B2	P3/N3/B3	P4/N4/B4	...	Varredura	Valor	Qualida...	Estampa de tempo	Valor (sem escala)
Drive1	0	0	0	0						
• Ard_Pot	1	4	0	0		100 9	0	192	13/12/2023 15:28:51,473	9 0
• Ard_Pot_2	1	4	0	1		100 9	0	192	13/12/2023 15:28:51,473	9 0
• Ard_Pot_3	1	4	0	2		100 9	88	192	13/12/2023 15:29:07,280	9 88
• Ard_Pot_4	1	4	0	3		100 9	140	192	13/12/2023 15:29:07,280	9 140
• Ard_PushButton_0	1	2	0	0		100 9	0	192	13/12/2023 15:28:51,513	9 0
• Ard_PushButton_1	1	2	0	1		100 9	0	192	13/12/2023 15:28:51,513	9 0
• Ard_PushButton_2	1	2	0	2		100 9	0	192	13/12/2023 15:28:51,513	9 0
• Ard_PushButton_3	1	2	0	3		100 9	0	192	13/12/2023 15:28:51,513	9 0
• Ard_PushButton_4	1	2	0	4		100 9	0	192	13/12/2023 15:28:51,513	9 0
• Ard_PushButton_5	1	2	0	5		100 9	0	192	13/12/2023 15:28:51,513	9 0
• LedVd_Ard_1	1	1	0	0		100 9	0	192	13/12/2023 15:28:51,534	9 0
• LedVd_Ard_2	1	1	0	1		100 9	0	192	13/12/2023 15:28:51,534	9 0
• LedVd_Ard_3	1	1	0	2		100 9	0	192	13/12/2023 15:28:51,534	9 0
• LedVd_Ard_4	1	1	0	3		100 9	0	192	13/12/2023 15:28:51,534	9 0
• Led_Verm_Ard_1	1	3	0	0		100 9	0	192	13/12/2023 15:28:51,554	9 0
• Led_Verm_Ard_2	1	3	0	1		100 9	0	192	13/12/2023 15:28:51,554	9 0
• Led_Verm_Ard_3	1	3	0	2		100 9	0	192	13/12/2023 15:28:51,554	9 0
• Led_Verm_Ard_4	1	3	0	3		100 9	0	192	13/12/2023 15:28:51,554	9 0

Figura 20 – Drive em comunicação

1.8. ERROS DE COMUNICAÇÃO

Monitorar o estado das redes no *software Codesys* pode ser feito em **Devices** também, indicando o estado de cada uma das etapas de comunicação e reportando o estado (**Status**). Caso encontre problemas de conexão, aparecerá um triângulo vermelho no ícone de tipo de comunicação que está em falha.

Falha na comunicação Modbus RTU: causado por falta de conexão serial com o escravo, erro de configuração da porta COM ou endereçamento do escravo diferente do direcionado pelo mestre *Codesys*.

Falha de comunicação Modbus TCP: causado por erro de configuração do direcionamento do IP do mestre dentro do Elipse, além da porta de comunicação que por padrão é 502. No caso específico onde os dois softwares rodarem na mesma máquina, o IP padrão interno do mestre é 127.0.0.1. E quando rodando em máquinas diferentes na mesma rede LAN, o IP do mestre deve ser configurado com o mesmo Host IP da máquina onde roda o *Codesys*.

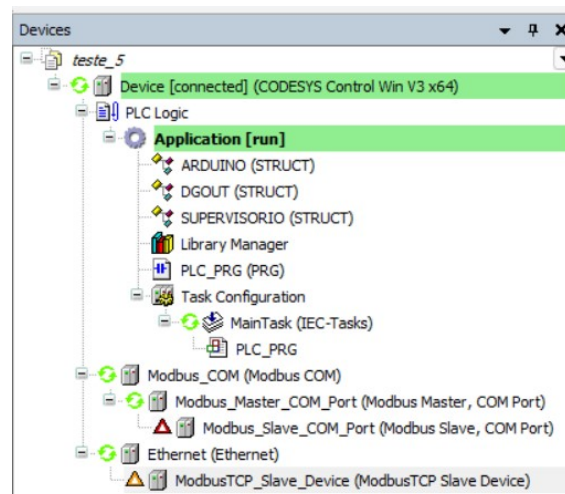


Figura 21 – Estrutura com falha de comunicação

2. Esquema de Ligação

Abaixo temos o diagrama de ligações mais comuns entre os periféricos e o protótipo.

Nas 6 entradas digitais (DI0 a DI5), podemos conectar diversos tipos de sensores, botoeiras, dispositivos ativos e passivos com contato seco. A configuração desse tipo de entrada é do tipo que ativa em nível alto.

As entradas analógicas podem medir grandezas de tensão e corrente, nos ranges de 0 a 5V, 0 a 10V para tensão e 0 a 20mA e 4 a 20mA, onde no modo tensão, utilizamos a referência GND no pino COM e entrada de tensão no pino V0 ou V1, e no modo corrente, primeiramente unir os pinos V0 com I0 ou V1 com I1 e nessa união podemos medir o sinal de retorno do transmissor de 4 a 20mA, e a alimentação positiva deste mesmo transmissor é proveniente da saída +24V presente na placa. Os jumpers JP1 e JP2 são responsáveis por selecionar o range de 0 a 5V ou 0 a 10V no modo tensão, onde

o jumper conectado representa a seleção do range 0 a 10V. Observar também que no modo corrente, esse jumper deve permanecer desconectado.

As saídas digitais podem ser divididas em dois tipos neste protótipo, onde temos 2 saídas a relé e 2 saídas a transistor com coletor aberto. As saídas a relé podem acionar cargas de até 220V x 10A e deve ter seus pinos NA e COM ligados em série com a carga e a fonte de alimentação. As saídas a transistor fornecem tensão diretamente em seus pinos DO2 e DO3 com referência negativa nos pinos DO/M-, e para a seleção da tensão fornecida, o jumper JP5 chaveia para 5V ou 24V com corrente máxima de 500mA.

A placa possui 6 saídas analógicas divididas em 3 tipos, onde duas são do tipo 4 a 20mA e o pino + fornece alimentação 24V para o atuador e o pino I- controla a corrente de controle. Temos também 2 saídas de tensão 0 a 10V, nos pinos GND como referência e VOUT como saída de tensão. O terceiro tipo, são as saídas PWM de potência, onde podemos controlar motores DC de até 48VDC x 5A, porém a alimentação deve ser de uma fonte DC externa, que é ligada aos pinos GNDM(-) e VCCM(+) e o motor deve ser ligado aos pinos M- e M+.

Para alimentar a placa, ligue o plugue de tomada à rede elétrica e pressione a chave localizada ao lado da saída do chicote de tomada e do porta fusível, todos localizados na lateral da caixa hermética. Para confirmação da alimentação adequada, a placa possui um led vermelho de 5mm para indicar o estado ligado e para proteção da etapa de 24V, ao lado deste led temos o fusível de 2A.

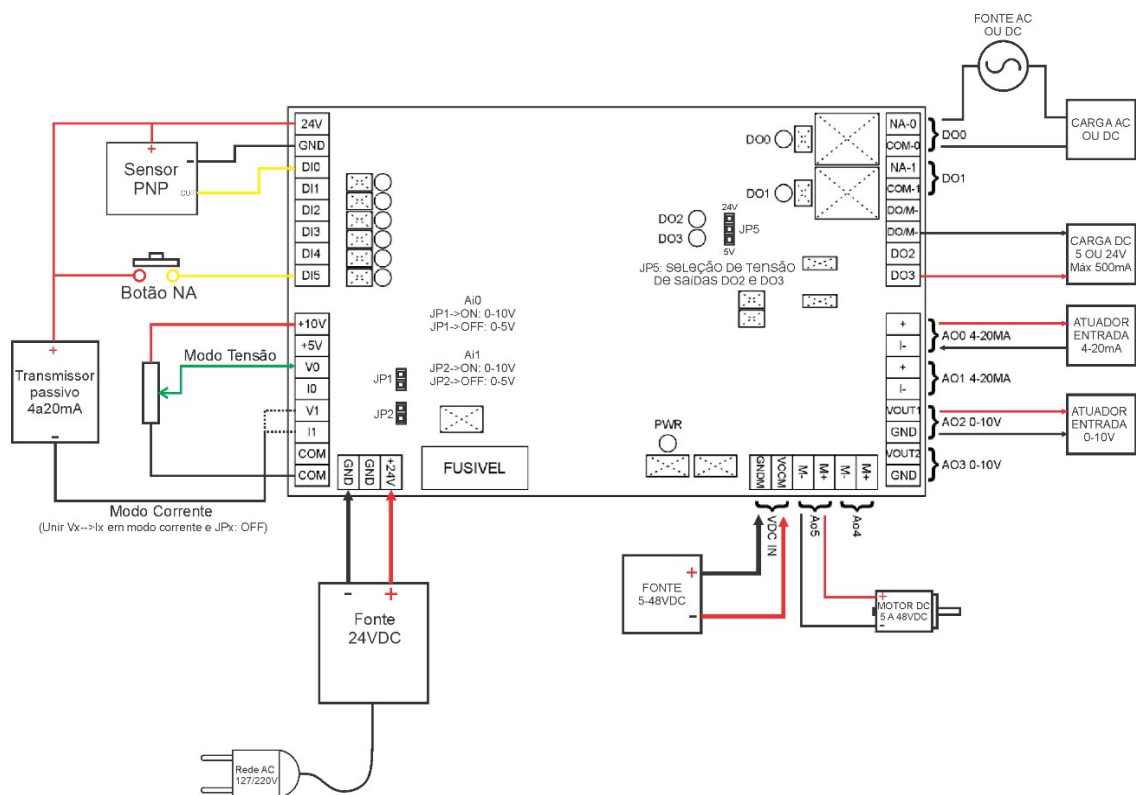


Figura 22 - Ligação dos periféricos com a placa de remotas de I/Os.