

CodeBox

A CodeBox ou CodeRPIBox é uma jiga de teste multiprotocolo que utiliza o Software Codesys embarcado em uma placa Raspberry PI 3. Ela pode ser utilizada como escravo Modbus TCP, escravo Ethernet/IP, escravo Profinet IO ou mestre/escravo OPC-UA. A figura abaixo mostra uma imagem do CodeBox.

Figura 1 – Frontal do CodeBox com os Botões, Leds e a entrada HDMI



O módulo possui quatro botões de entrada digital, quatro leds de saída digital e uma saída relé do tipo triac (contato seco, podendo ligar carga AC somente até 220Vac, a lógica do relé é NF, ou seja, zero liga o relé e 1 desliga). O potenciômetro está ligado em um conversor AD (ADS1115) e se comunica via interface I2C.

Para comunicar o CodeBox na rede, basta ligar um cabo de rede em uma rede DHCP ou configurar o IP localmente.

Figura 2 – Lateral do CodeBox com acesso ao conector RJ45 e as USBs



A tabela 1 mostra a Ligação física do hardware da CodeBox.

Tabela 1 – Ligações físicas da raspberry PI no Codebox

External IO	GPIO	Pinos		GPIO	External IO
Vcc Botões	3.3V	1	2	5V	
		3	4	5V	RELE1(DC+)
		5	6	GND	RELE1(DC-)
RELE1(CH1)	GPIO4 (1) (output)	7	8		
	GND	9	10		
BOTAO1	GPIO17 (input)	11	12	GPIO18 (input)	BOTAO3
BOTAO4	GPIO27 (input)	13	14	GND	
LED1	GPIO22 (output)	15	16	GPIO23 (output)	LED2
	3.3V	17	18	GPIO24 (output)	LED3
		19	20	GND	COMUM LEDS
		21	22	GPIO25 (output)	LED4

Obs.: (1) o GPIO4 esta por default (no config.txt da raspberry) configurado como Output. Isso porque sem esta configuração, ele estava como input e por default é um sinal de clock. Somente consegui fazer funcionar como saída, como entrada não consegui inibir o clock.

(2) o Botão 2 não foi mapeado pois ele está com problema na mola. No final tem mais itens no modulo que GPIOs disponíveis (somente 8 GPIOs é suportado nesta placa pelo Codesys). Preferi tirar o botão e colocar todos os leds e o Relé.

(3) O Relé1 é do tipo NF. Ou seja, somente vai ligar quando a lógica for 0. Ele é um relé de estado sólido (SSR – FC 80 G3MB-202P). Ele suporta somente carga AC (240Vac) até 2A.

Seguir para os próximos passos:

- 1) Se ainda não instalou o Codesys na sua máquina vá para o item 1 – Instalando o codesys e os plugin
- 2) Se vc já tem o Codesys e os plugins da Raspberry Pi instalado vá para o item 2 – Configurando o Codebox

Para configurar o IP localmente é necessário de ligar um monitor (entrada HDMI no frontal) além do teclado e mouse nas portas USBs, conforme figura acima. Quando ligado o monitor e o teclado, o Codesys não funciona.



Codesys licensas

https://store.codesys.com/en/howto_applicationbasedlicenses

<https://store.codesys.com/en/softplcs-for-applicationbased-licenses>

1 - Instalando o Codesys

Baixar a última versão do Codesys no site do fabricante (Codesys Development System)

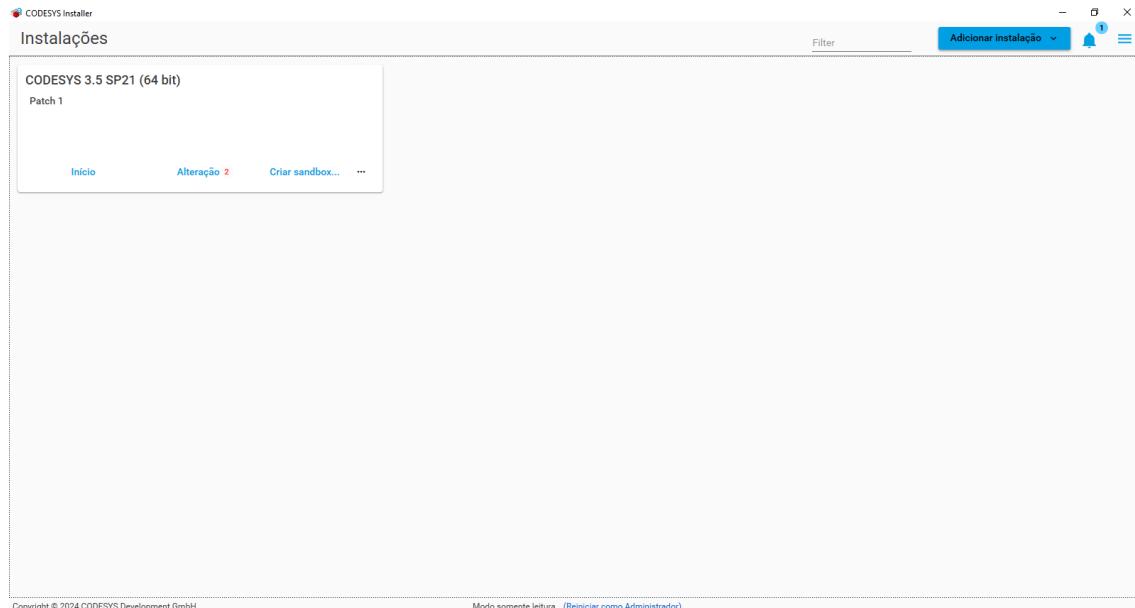
<https://store.codesys.com/en/codesys.html>

O plugin para a raspberry pi (RPI)

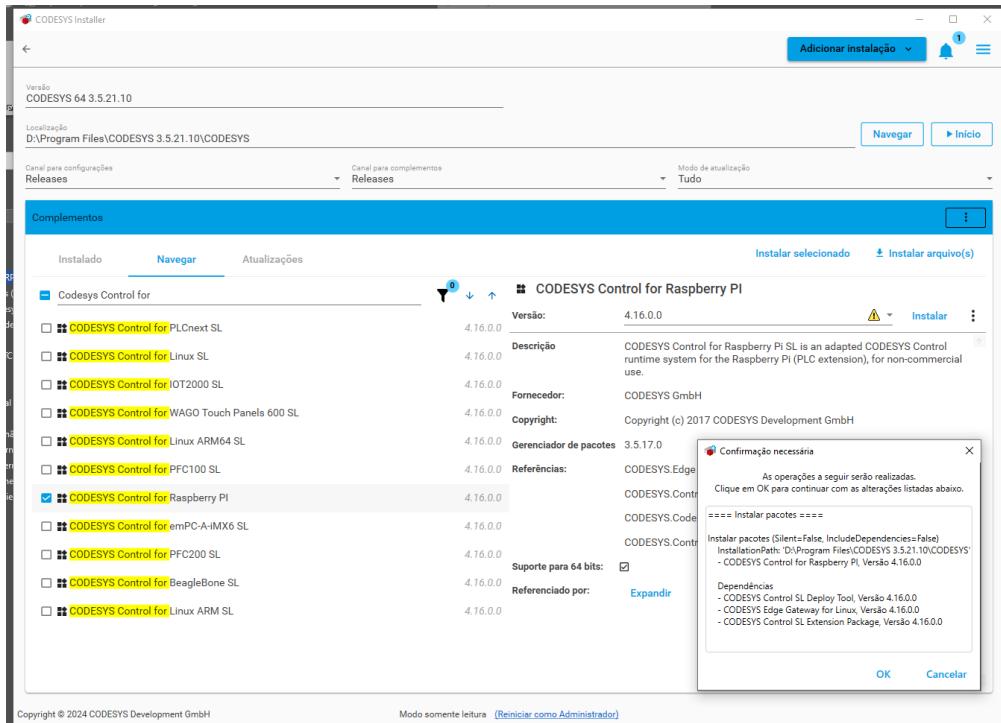
Video do youtube : <https://www.youtube.com/watch?v=uNpAhK9QdWQ>

Para o codesys conseguir rodar na Raspberry PI (estamos usando aqui a versão Raspberry PI 3 Modelo B) deve usar o Control Linux SL (ou SoftPLC), ou seja, desta forma, o codesys roda sobre um sistema embarcado baseado em ambiente Linux. Para a integração com o Sistema operacional padrão da Raspberry Pi (3 ou 4) (a raspberry roda uma versão modificada do debian) existe um plugin específico chamado “Codesys Control for Raspberry Pi”

Executar pgma “Codesys Installer” conforme tela abaixo.



Escolher alteração (change) e na janela de complementos escolher “Codesys Control for Raspberry Pi”, conforme figura abaixo. E então clicar em instalar.



e escolher raspberry pi, ele vai sugerir o que deve ser instalado no Codesys para rodar a raspberry. Depois tirar a opção licença no canto inferior esquerdo da tela e clicar em continuar.

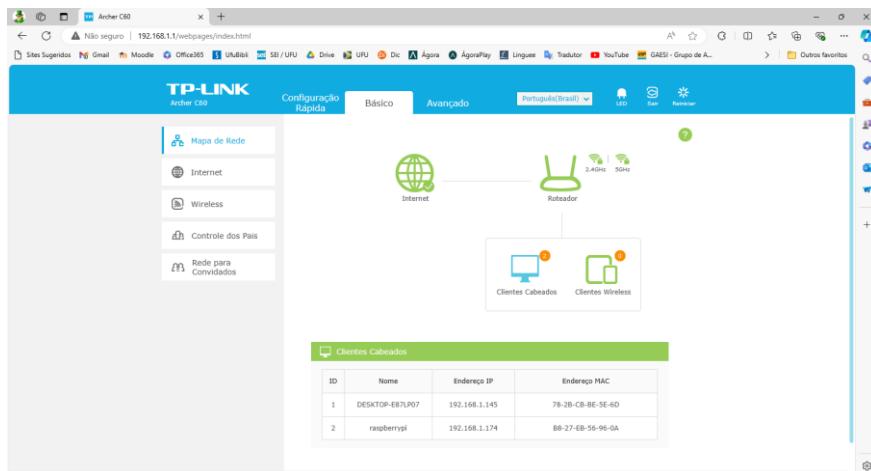
Key	Issue Type	Summary	Resolution	Note
RTSL-3261	Improvement	Linux SL products - change logger default settings	Fixed	
RTSL-3239	Improvement	Redundant Time Provider	Fixed	
RTSL-3062	Improvement	Linux packages (deb/rpm) restrict folder permissions for "other" users	Fixed	[[GENERAL]] Access to directories related to CODESYS deb packages is now restricted for "other" users.
RTSL-3022	Bug	Deploy Tool 2FA with key-based login fails on first try	Fixed	[[KNOWN_LIMITATIONS]] CODESYS Control for PLCNext SL working directory has not been restricted, because Phoenix firmware needs access for Axoline.

Eu aceito o(s) contrato(s) de licença

CODESYS Group | We software Automation.

Conectando a RPI no Codesys (primeira vez)

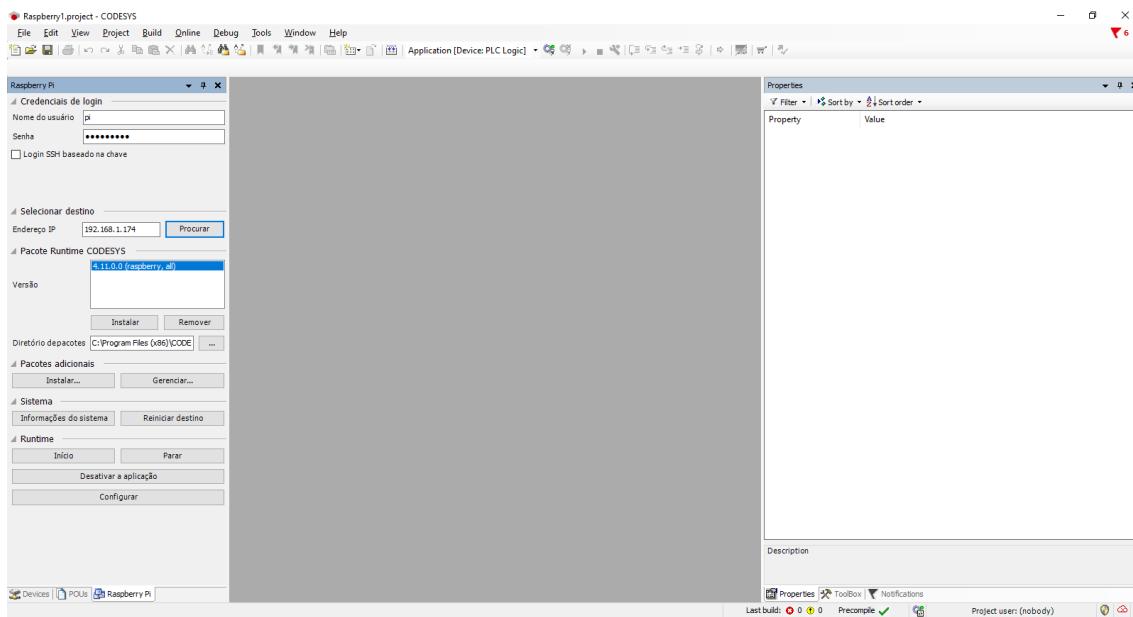
Primeiro devemos descobrir o IP do RPI. Ele esta conectado com o cabo ethernet no mesmo roteador (switch) do computador. Acessando o roteador eh possivel ver o IP do RPI, conforme figura abaixo.



Criar uma configuração no Codesys com o hardware raspberry pi.

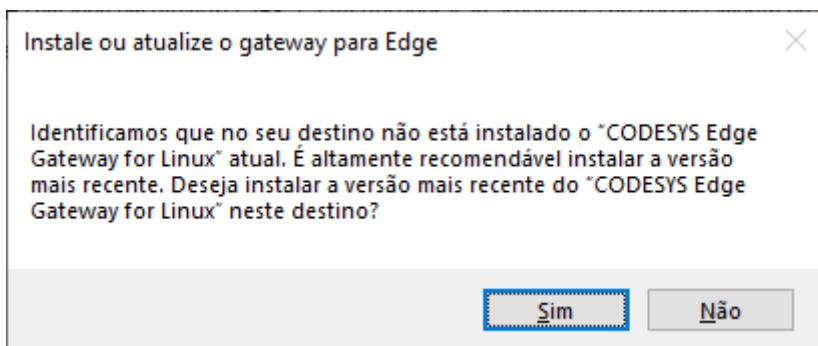
Escolher a opção no menubar “Tools/Raspberry Update”

Coloquei usuário e senha ufu



Messages - Total 0 error(s), 0 warning(s), 10 message(s)		
Runtime Deploy Tool		x
Description	Project	
① Executando SSH comando em ufu@192.168.1.174: Recuperação arquitetura do gerenciamento de pacote...		
① Saída padrão: armhf		
① Executando SSH comando em ufu@192.168.1.174: Verifique os pré-requisitos: confirme se há pacotes d...		
① Executando SSH comando em ufu@192.168.1.174: Verificar pré-requisitos: obter arquitetura de CPU		
① Executando SSH comando em ufu@192.168.1.174: Parar o runtime		
① Executando SSH comando em ufu@192.168.1.174: Pacote: salvar o arquivo cfg existente		
① Executando SSH comando em ufu@192.168.1.174: Pacote: salvar o arquivo cfg existente		
① Executando SSH comando em ufu@192.168.1.174: Pacote: remova o antigo		
① Executando SCP comando em ufu@192.168.1.174: Pacote: transferir novo runtime		
① Executando SSH comando em ufu@192.168.1.174: Pacote: instalar novo runtime		

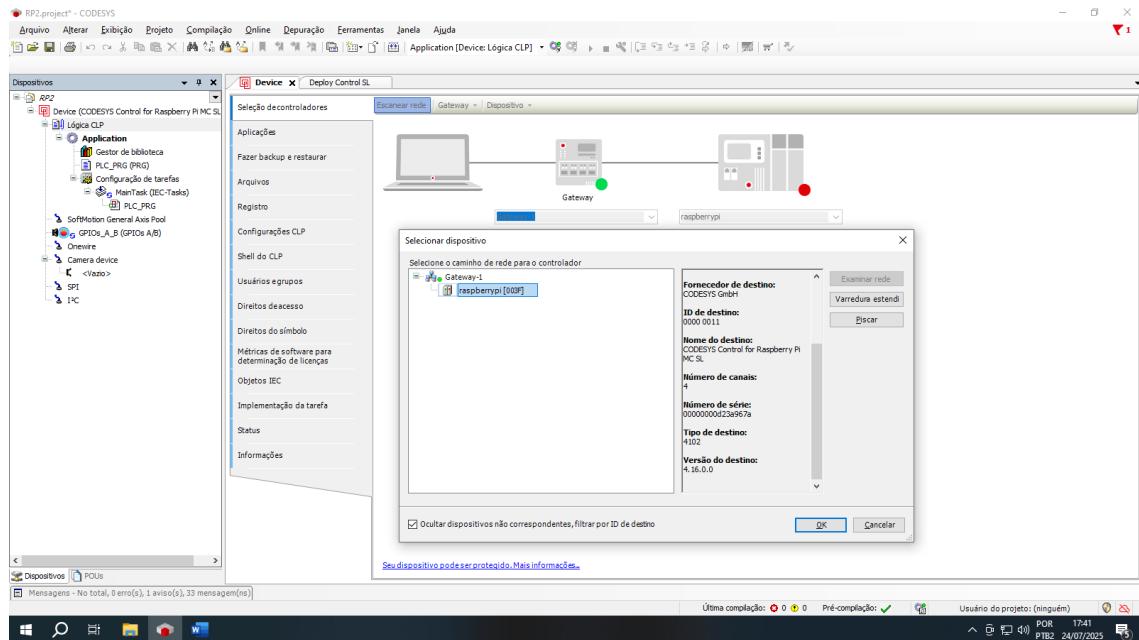
Na primeira instalação ele vai pedir para instalar o Codesys Edge. Deve aceitar esta instalação.



Messages - Total 0 error(s), 0 warning(s), 19 message(s)			
Runtime Deploy Tool		Project	Object
Description	Project	Object	Position
① Executando SSH comando em ufu@192.168.1.174: Recuperação arquitetura do gerenciamento de pacote...			
① Saída padrão: armhf			
① Executando SSH comando em ufu@192.168.1.174: Verifique os pré-requisitos: confirme se há pacotes d...			
① Executando SSH comando em ufu@192.168.1.174: Verificar pré-requisitos: obter arquitetura de CPU			
① Executando SSH comando em ufu@192.168.1.174: Parar o runtime			
① Executando SSH comando em ufu@192.168.1.174: Pacote: salvar o arquivo cfg existente			
① Executando SSH comando em ufu@192.168.1.174: Pacote: salvar o arquivo cfg existente			
① Executando SSH comando em ufu@192.168.1.174: Pacote: remova o antigo			
① Executando SCP comando em ufu@192.168.1.174: Pacote: transferir novo runtime			
① Executando SSH comando em ufu@192.168.1.174: Pacote: instalar ...			
① Saída padrão: Selecting previously unselected package codesysedge.			
① Saída padrão: (Reading database ... 134761 files and directories cur...			
① Saída padrão: Preparing to unpack .../codesysedge_edgarmhf_4.1...			
① Saída padrão: Preinst: Check architecture; info: detected armv7l			
① Saída padrão: Unpacking codesysedge (4.12.0.0) ...			
① Saída padrão: Setting up codesysedge (4.12.0.0) ...			
① Saída padrão: make config rw			
① Saída padrão: create and install daemon			
① Saída padrão: codesysedge started			

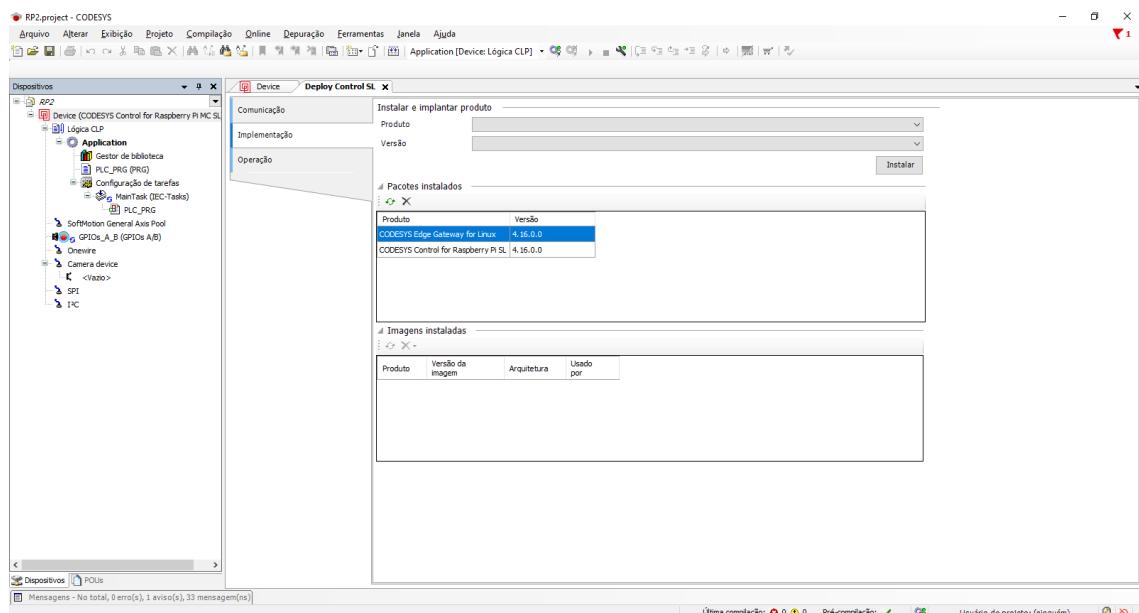
Atualizando a versão do Codesys na Raspberry

Caso a raspberry ja esteja com uma versao do codesys, vc pode acessa-la usando o botao scan



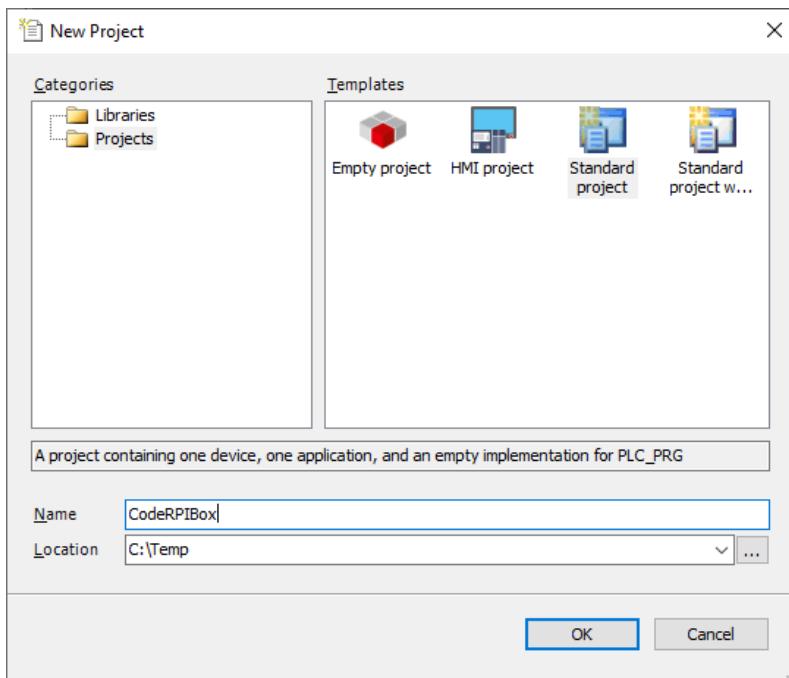
Neste caso a versao do drive da raspberry instalado deve ser o mesmo da que esta na raspberry. Veja que a versao da raspberry encontrada eh V4.16.0.0.

Caso nao for igual eh possivel instalar o driver atual na raspberry da seguinte forma. Va no menu bar em ferramentas e entao Deploy Control SL. Escolher a versao de driver desejada.

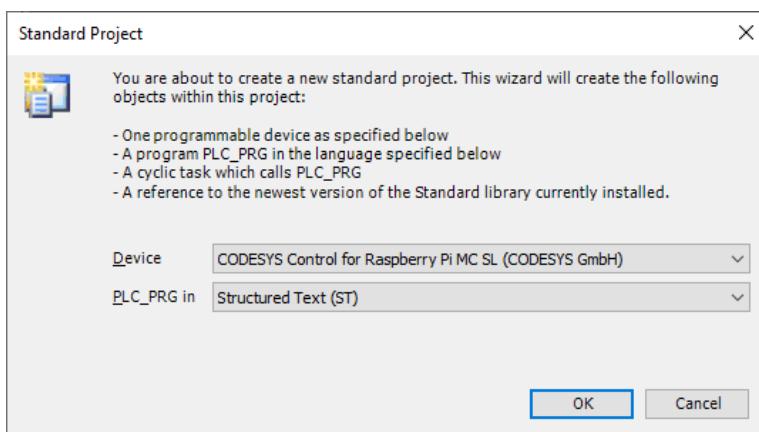


2 - Configurando a CodeBox no Codesys

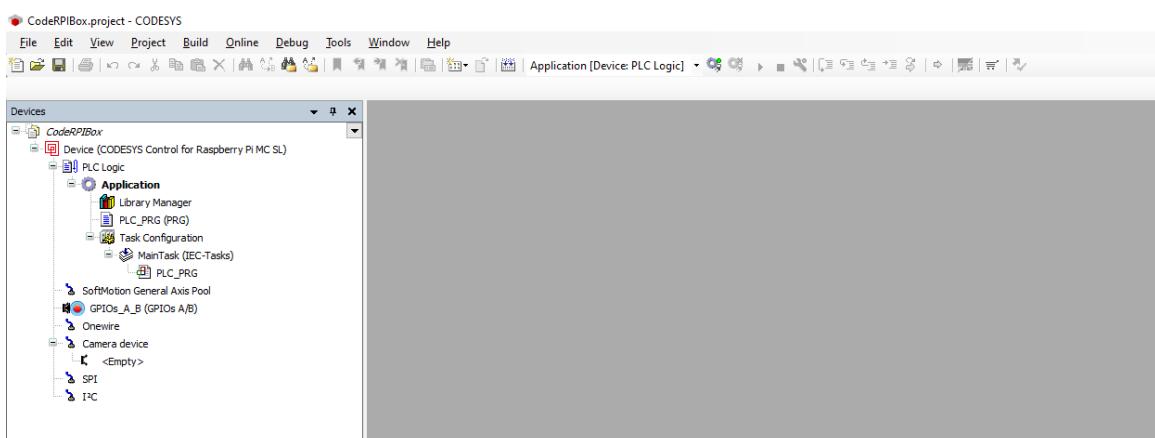
Abrir um novo projeto (Standard Project)



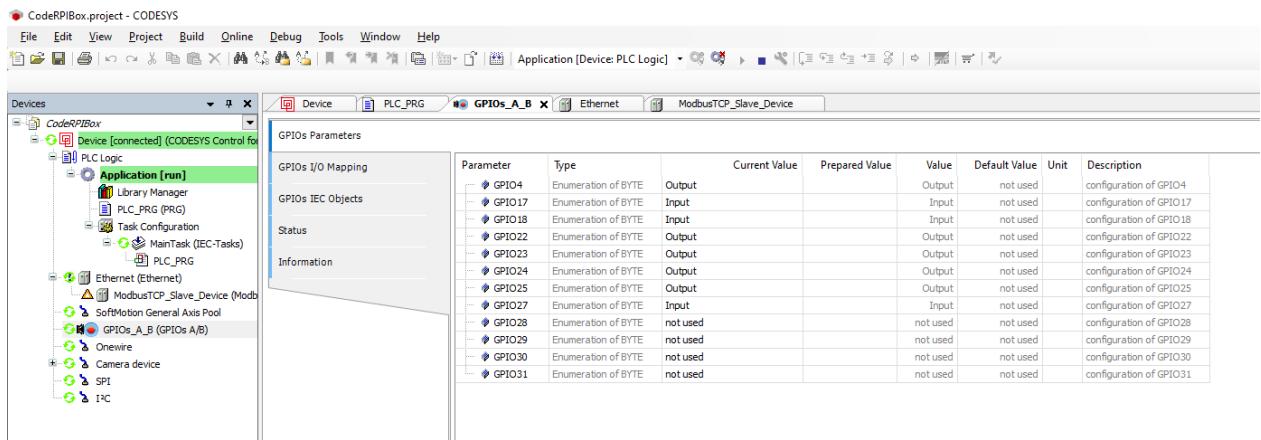
Escolher o device Raspberry PI MC SL. Lembrando que a Codebox esta usando a raspberry 3 Modelo B.



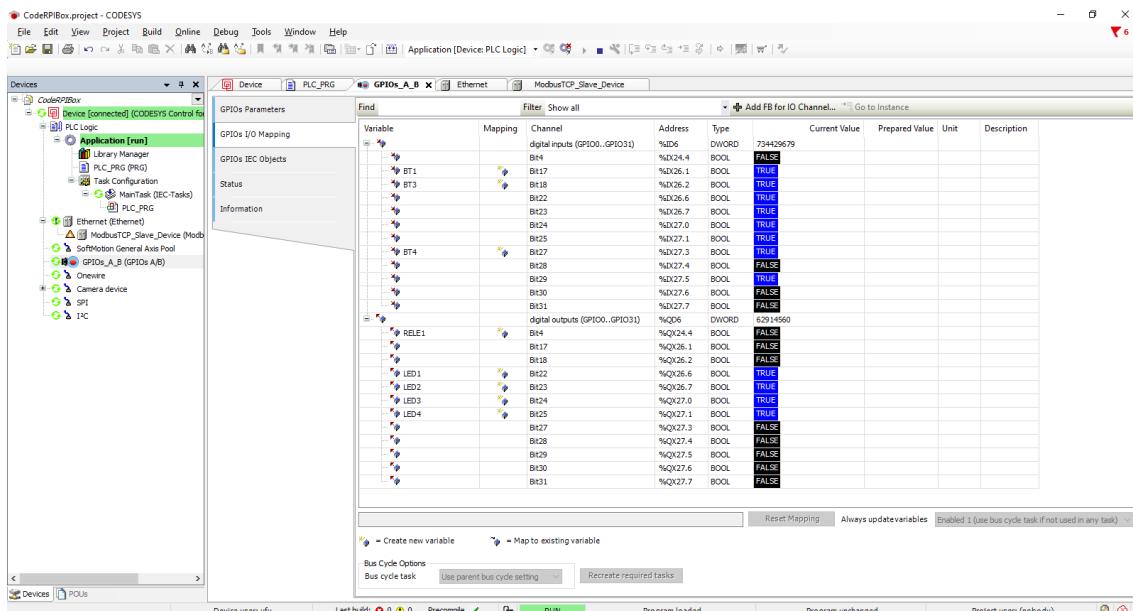
Após a criação do projeto aparecerá um projeto padrão conforme figura abaixo.



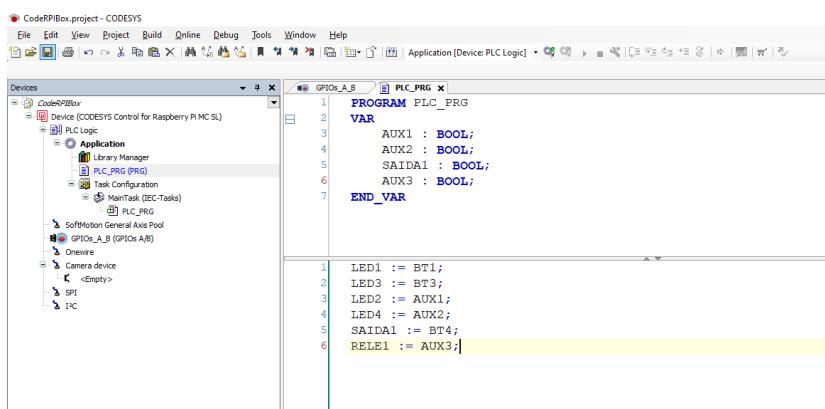
Configurando os IOs do Codebox de acordo com a tabela 1. Para isso clicar em GPIOs_A_B e então ir em GPIO Parameters para configurar os IOs como entradas ou saídas (“Current Value”).



Depois ir em GPIO I/O Mapping e configurar os tags dos pontos para ser usados na lógica.

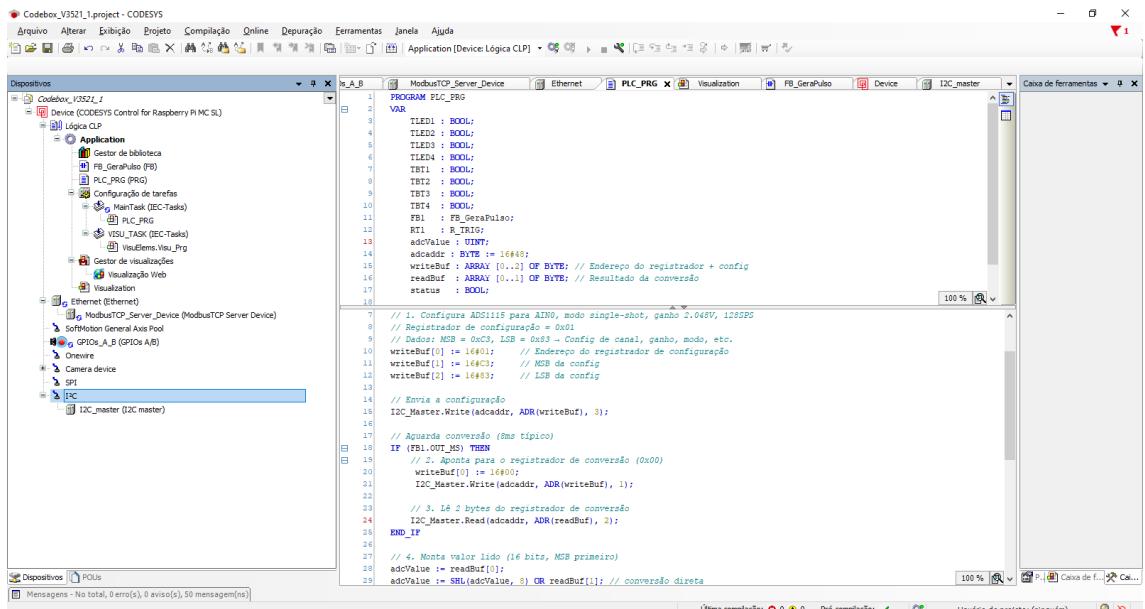


Por fim, utilizar os tags da tabela anterior na logica de controle (ladder, FBD, ST ou SFC).

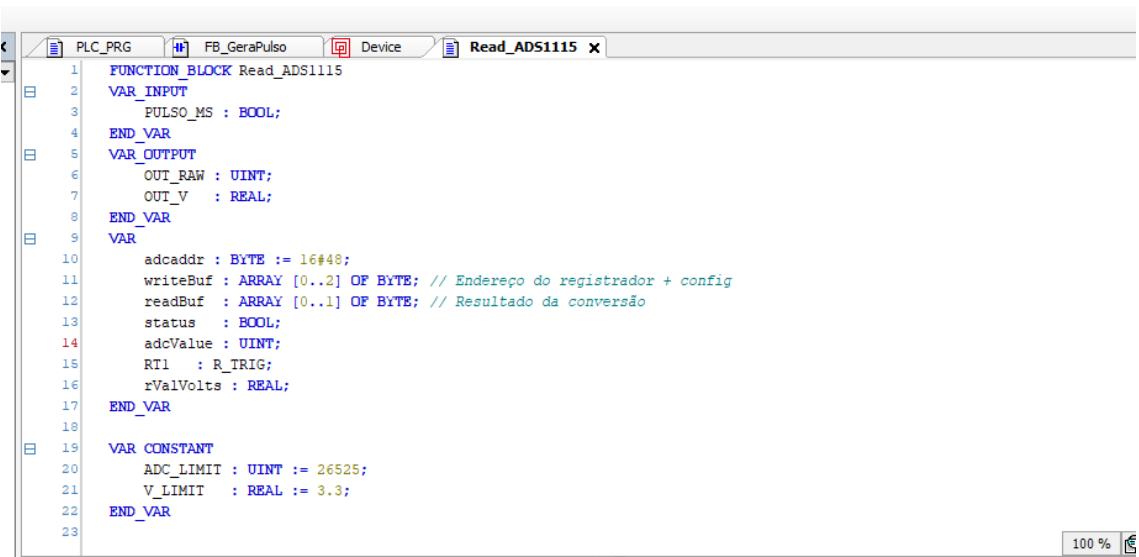


Para configurar o potenciometro (com módulo ADS 1115)

O modulo AD1115 esta conectado na porta I2C-1 da raspberry (default) (pinos 3 e 5). Neste caso, para configurar a porta I2C clicar em I2C e pedir para incluir um novo device. Entao selecionar I2C_Master.



Depois eh necessario incluir o seguinte codigo em ST (Foi criado um FB chamado Read_ADS1115). (obs.: o codigo esta em formato texto no apendice).



```

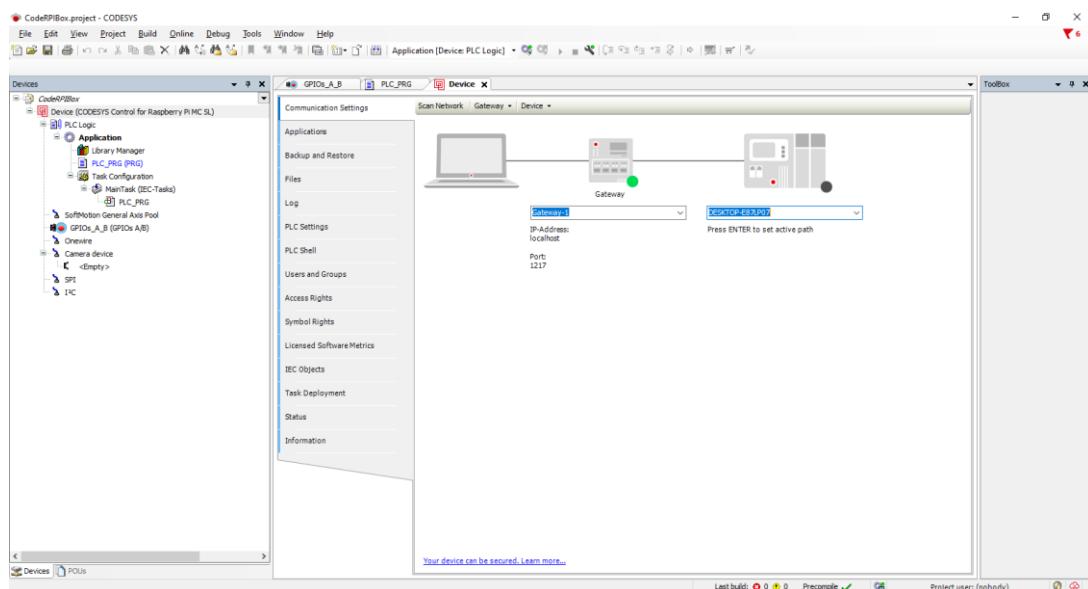
4 // Dados: MSB = 0xC3, LSB = 0x83 → Config de canal, ganho, modo, etc.
5 writeBuf[0] := 16#01;           // Endereço do registrador de configuração
6 writeBuf[1] := 16#C3;          // MSB da config
7 writeBuf[2] := 16#83;          // LSB da config
8
9 // Envia a configuração
10 I2C_Master.Write(adccaddr, ADR(writeBuf), 3);
11
12 // Aguarda conversão (8ms típico)
13 RT1(CLK:=PULSO_MS);
14 IF (RT1.Q) THEN
15     // 2. Aponta para o registrador de conversão (0x00)
16     writeBuf[0] := 16#00;
17     I2C_Master.Write(adccaddr, ADR(writeBuf), 1);
18
19     // 3. Lê 2 bytes do registrador de conversão
20     I2C_Master.Read(adccaddr, ADR(readBuf), 2);
21 END_IF
22
23 // 4. Monta valor lido (16 bits, MSB primeiro)
24 adcValue := readBuf[0];
25 adcValue := SHL(adcValue, 8) OR readBuf[1]; // conversão direta
26
27 (* convert raw value in float *)
28 rValVolts := INT_TO_REAL(adcValue);
29 rValVolts := (rValVolts / ADC_LIMIT) * V_LIMIT;
30
31 (* update the outputs *)
32 OUT_RAW := adcValue;
33 OUT_V   := rValVolts;
34

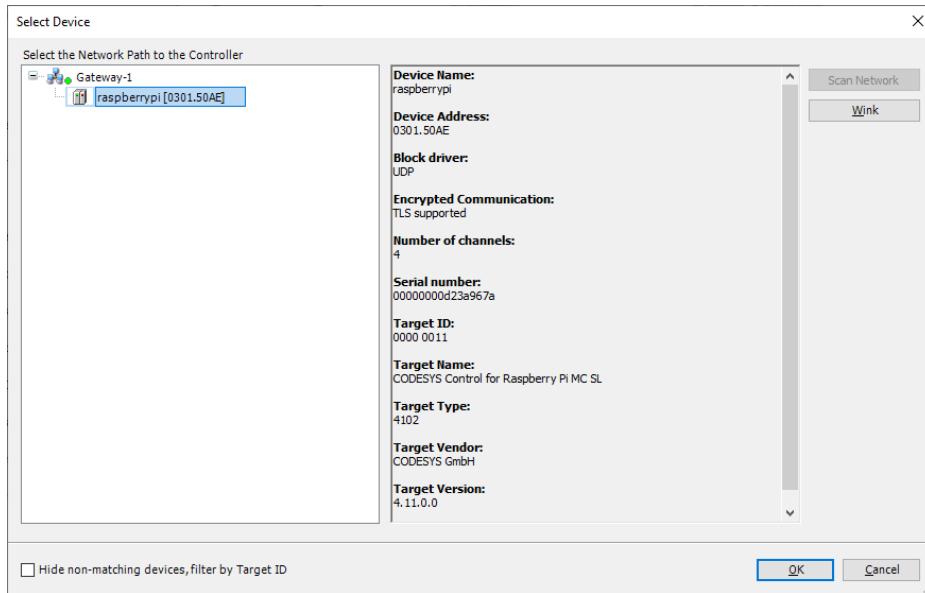
```

Conectando com a CodeBox

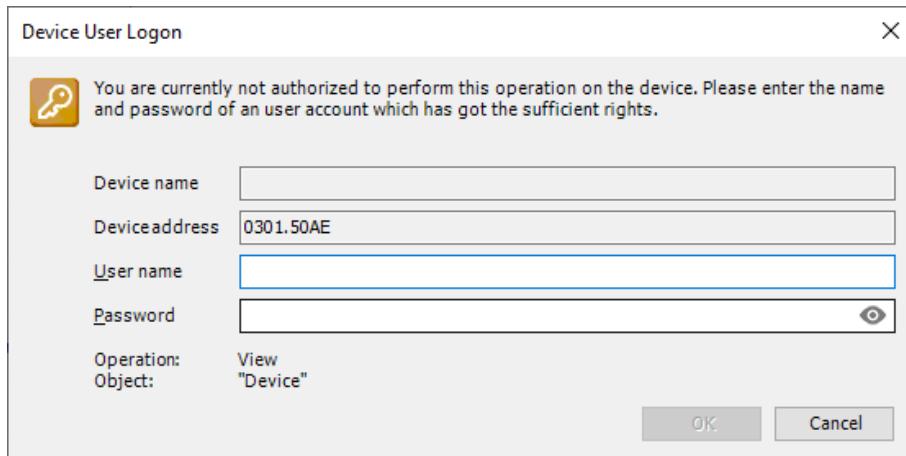
Agora vamos fazer download da configuração para o device.

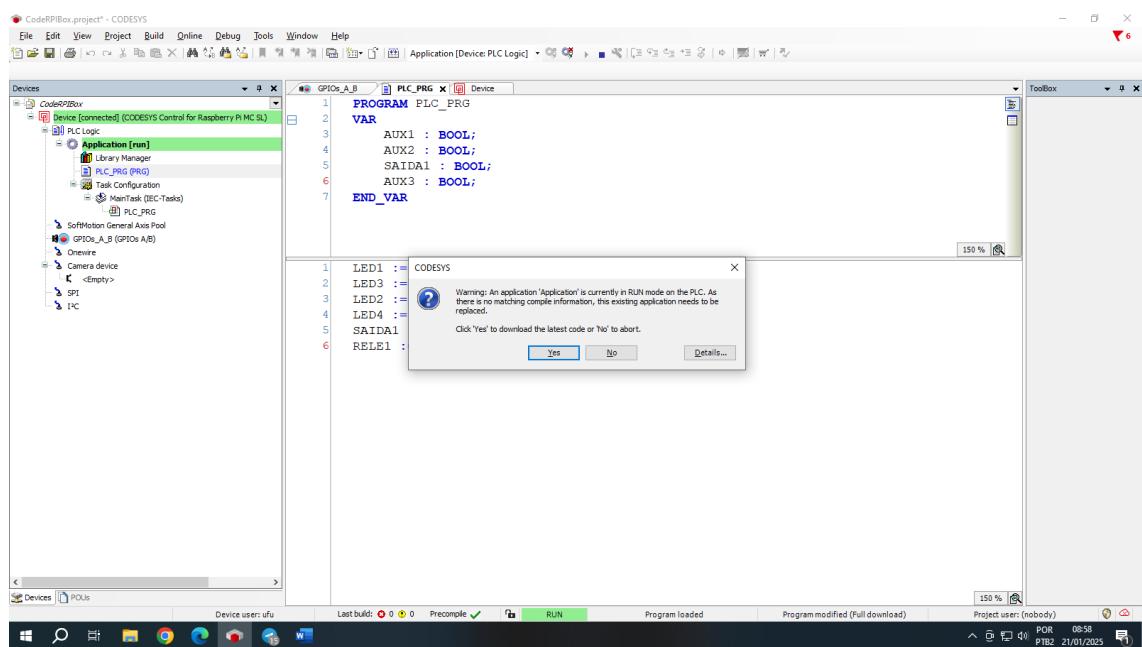
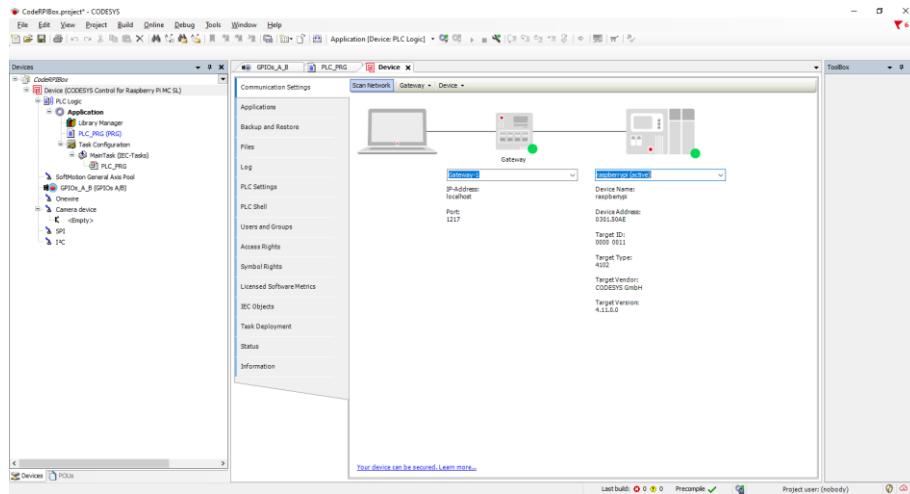
Primeiramente, clicar em ScanNetwork. Ele vai reconhecer uma RPI e o correspondente IP.

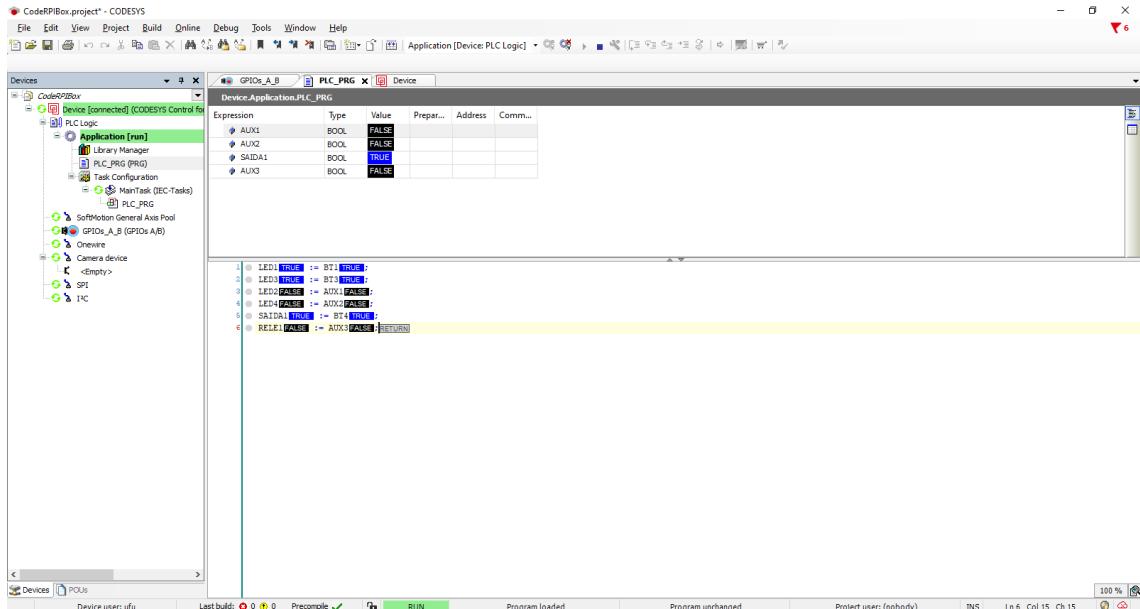
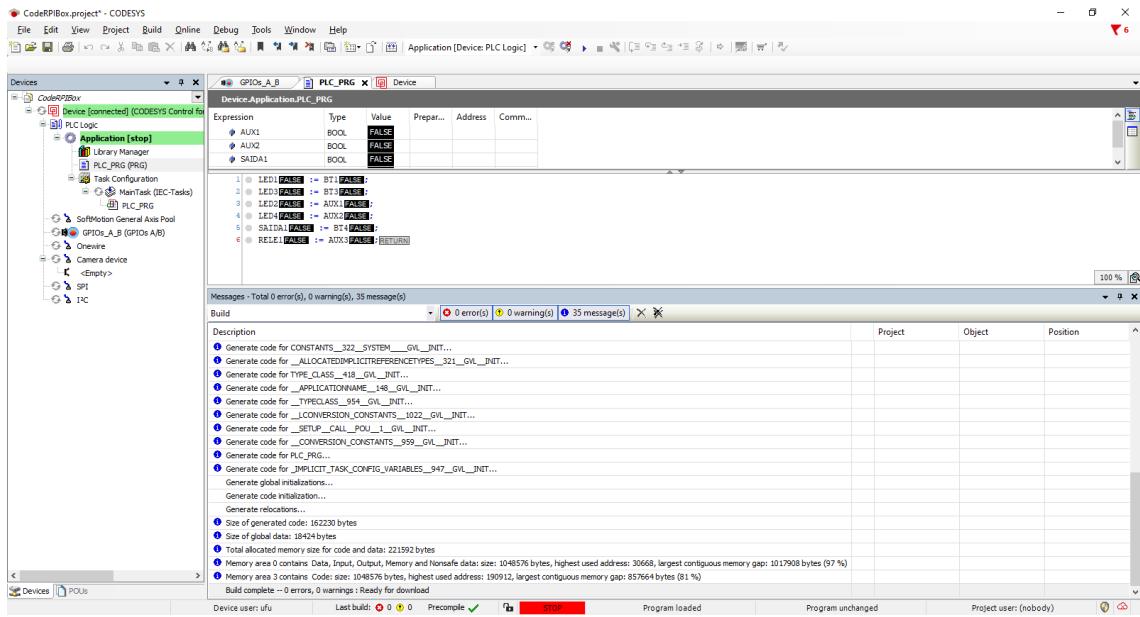




Obs.: Na 1^a vez ele vai pedir para configurar um usuário e senha. Coloquei user:ufu e senha:ufu. Ele pede para conectar duas vezes. Caso vc fez a configuração correta ele já vai logar e ficar online. Se a configuração eh incompatível com o device ele vai reclamar, mas ai eh necessário ir online e descobrir qual o tipo do device que vc configurou. No meu caso o device era uma Raspberry PI MC SL. Entao a configuração deve ser deste mesmo tipo (target ID e type) do device.

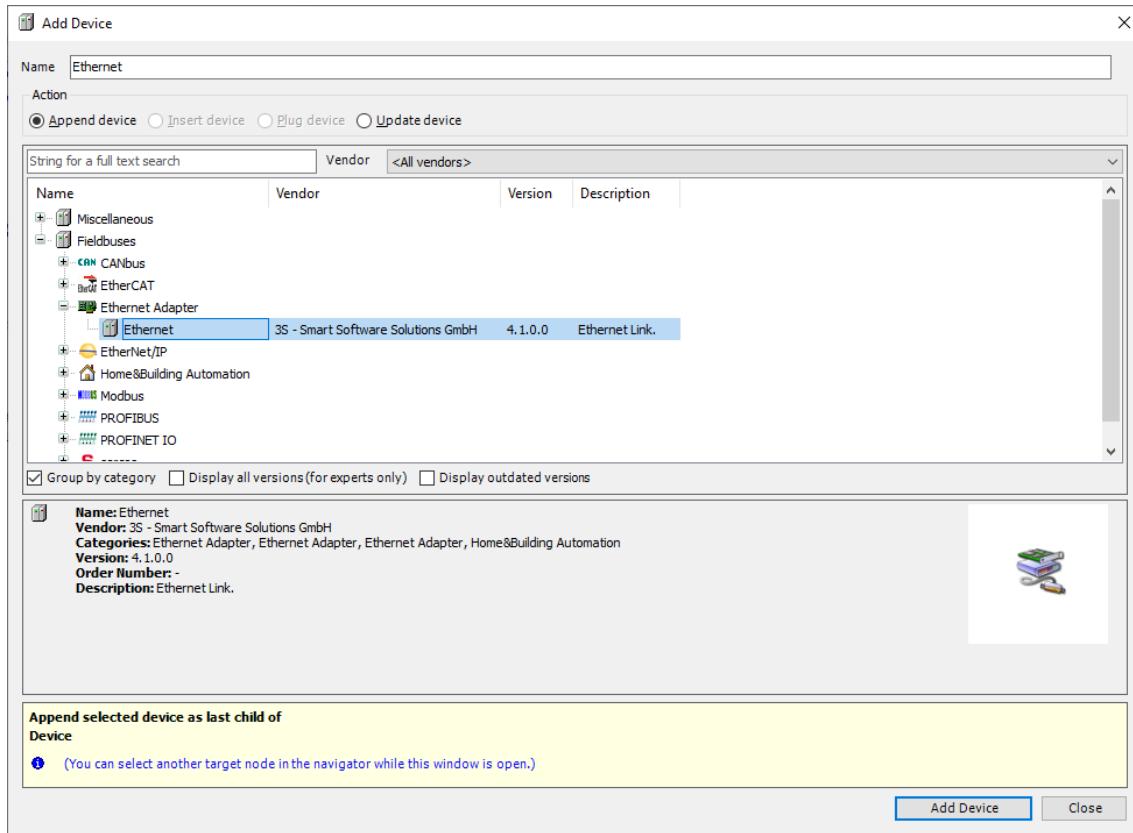




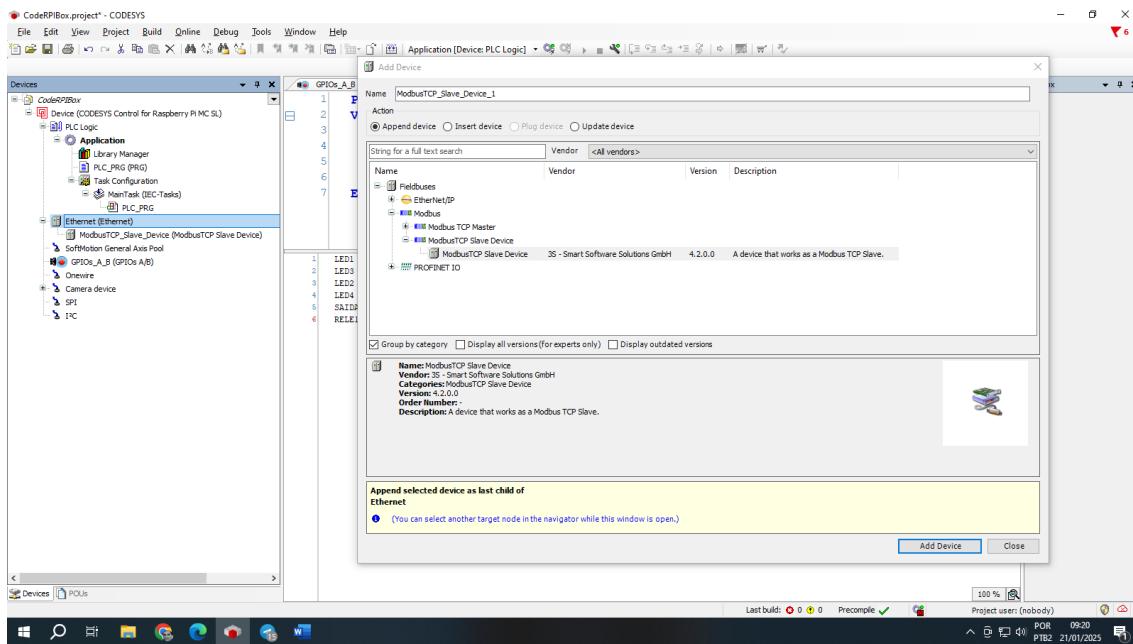


3 -Configurando Escravo Modbus TCP

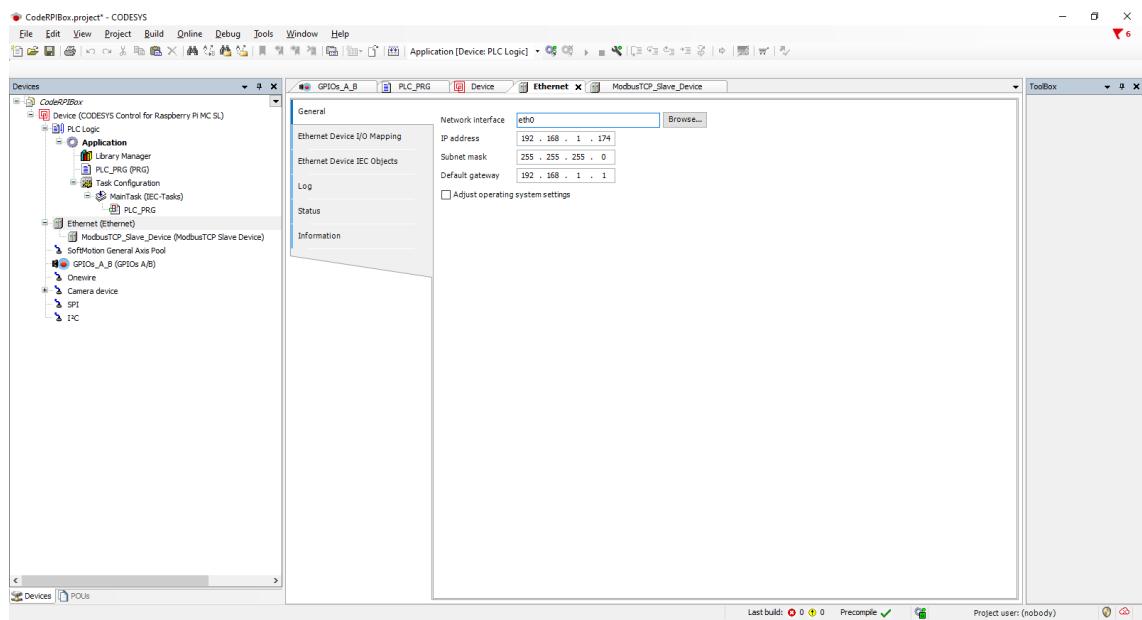
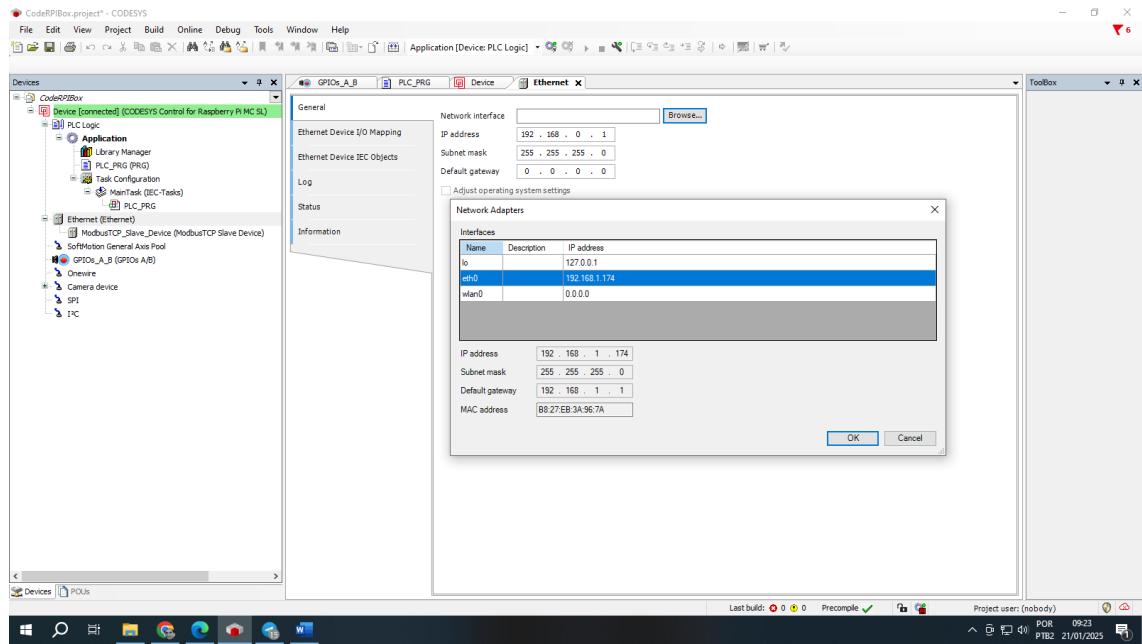
Incluindo porta ethernet. Clicar com o botão direito sobre o device (CODESYS) e clicar em “add device”. Escolher a porta ethernet.

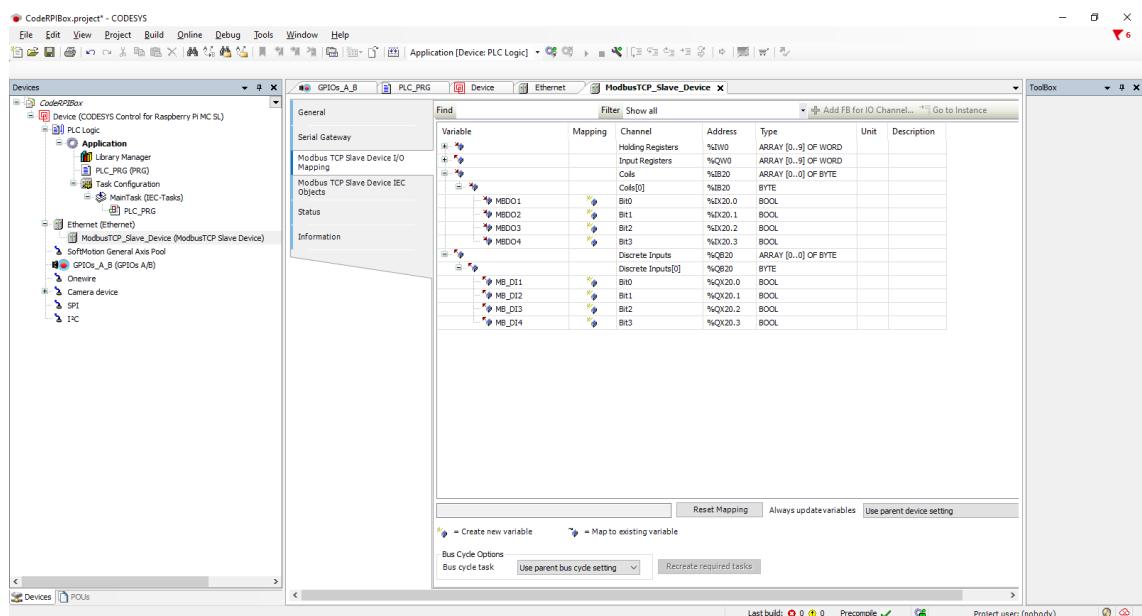
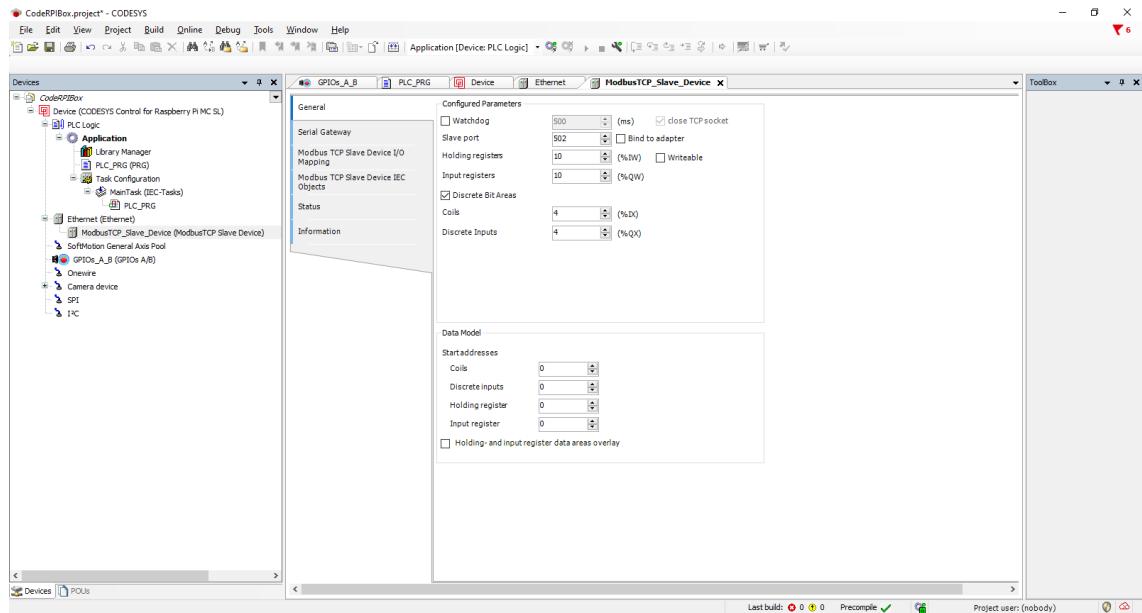


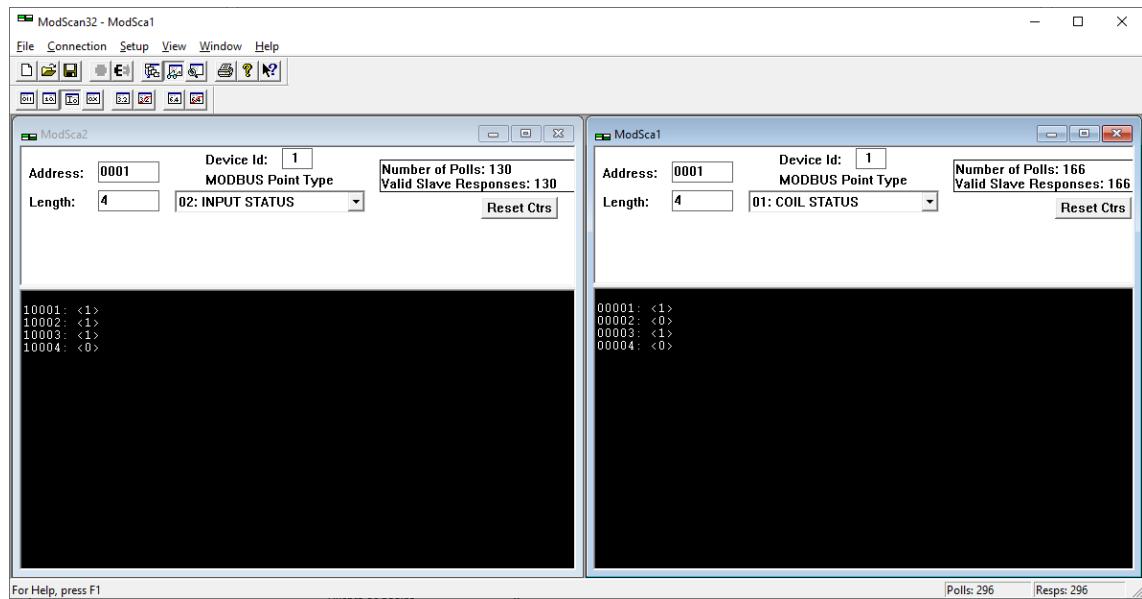
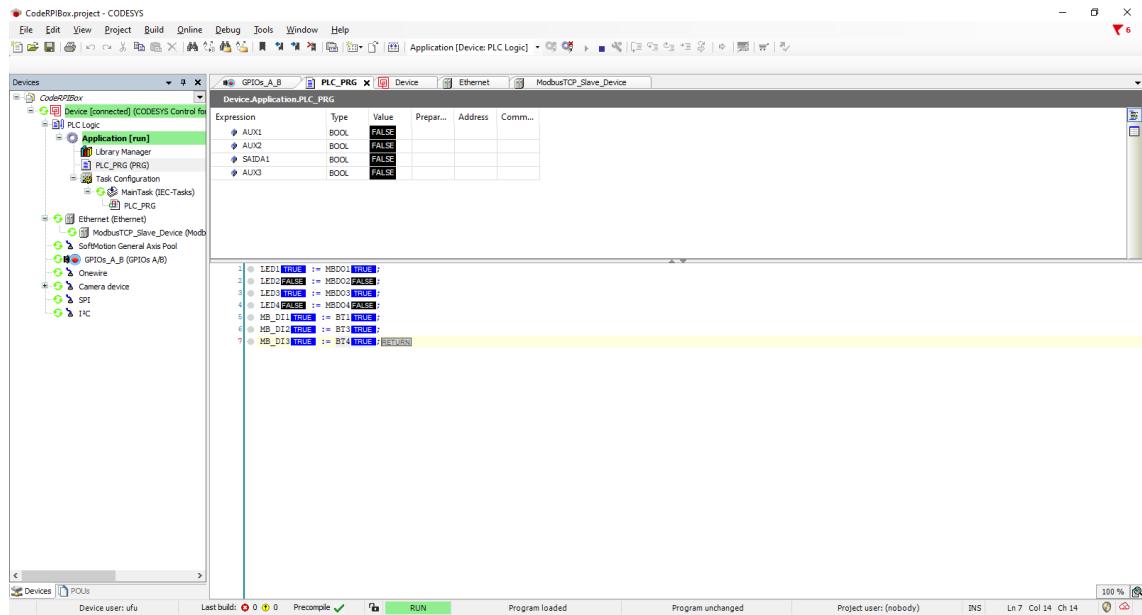
Clicar com o botão direito sobre a porta Ethernet e clicar em Add Device. Escolher ModbusTCP Slave.



Na porta ethernet escolher a corresponde porta ethernet da placa. Neste caso, clicar em Browse e escolher "Eth0". Clicar em OK.

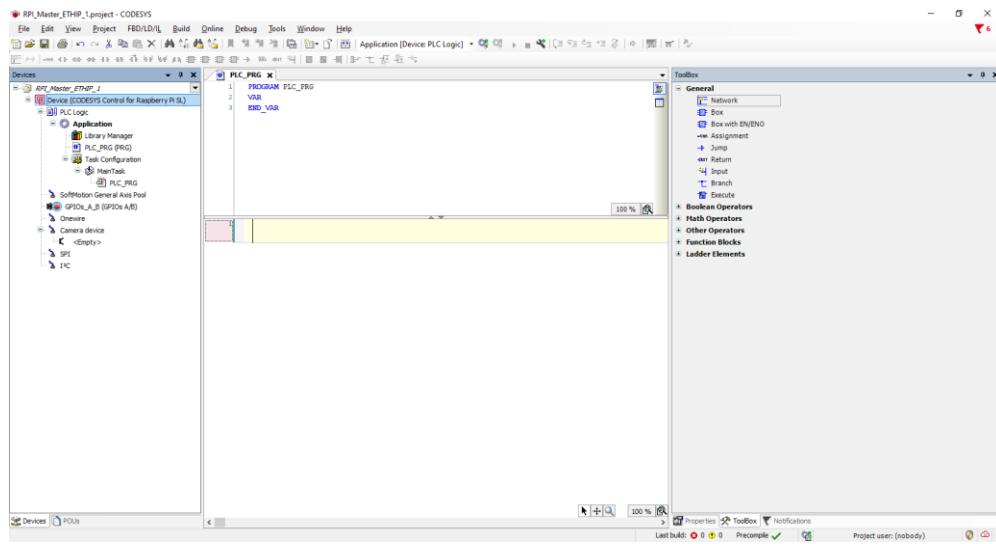




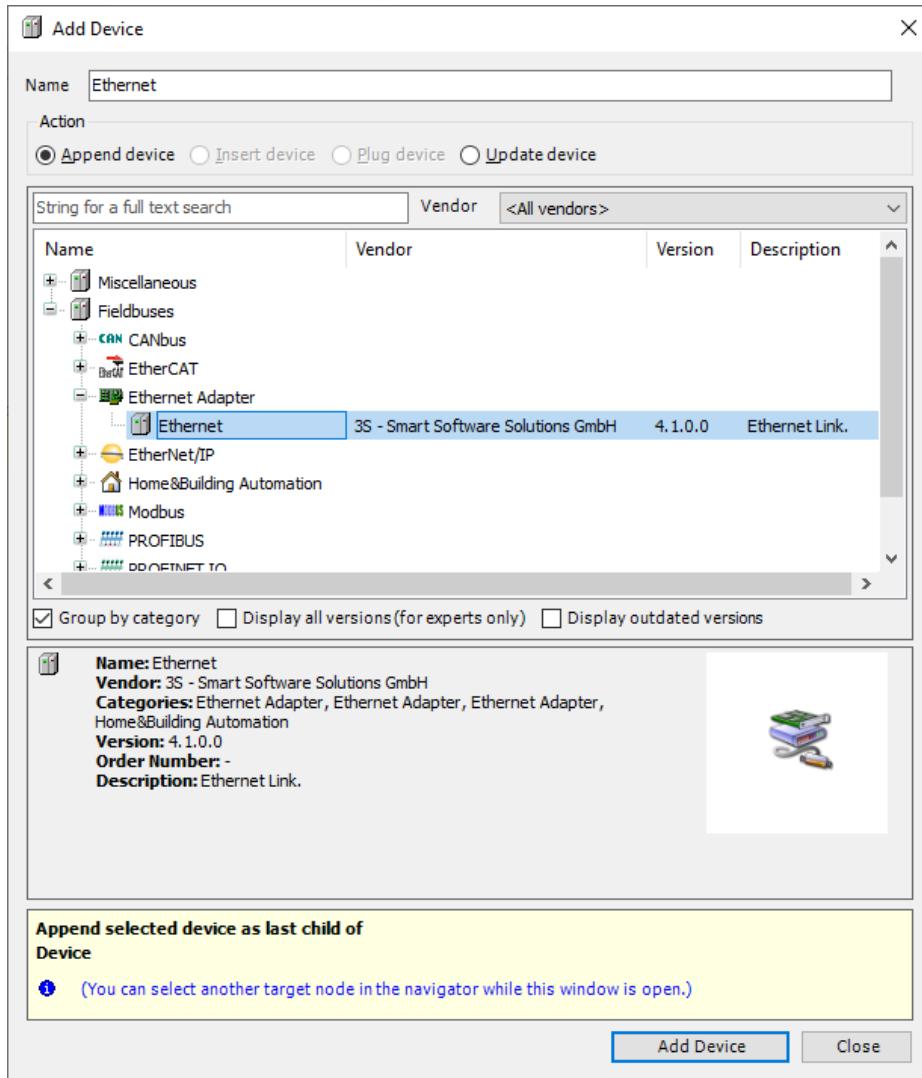


4 – Configurando Codebox como Mestre Modbus TCP

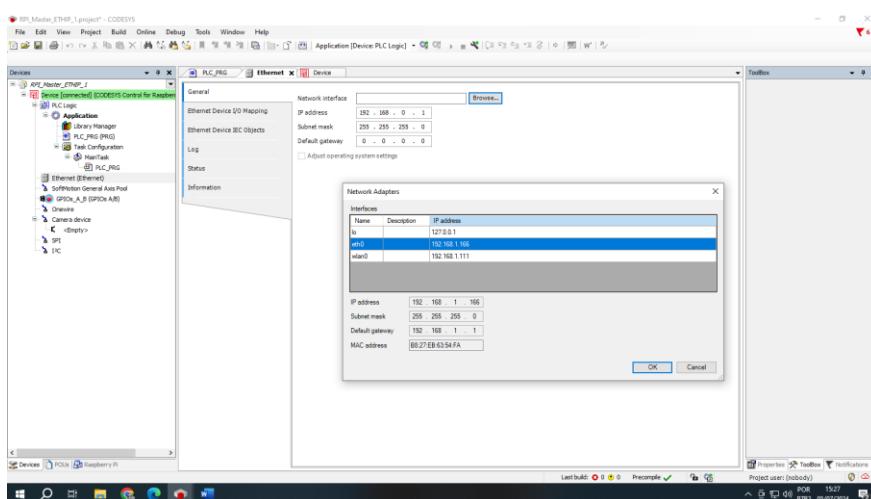
5 – Configurando Codebox como Mestre Ethernet/IP



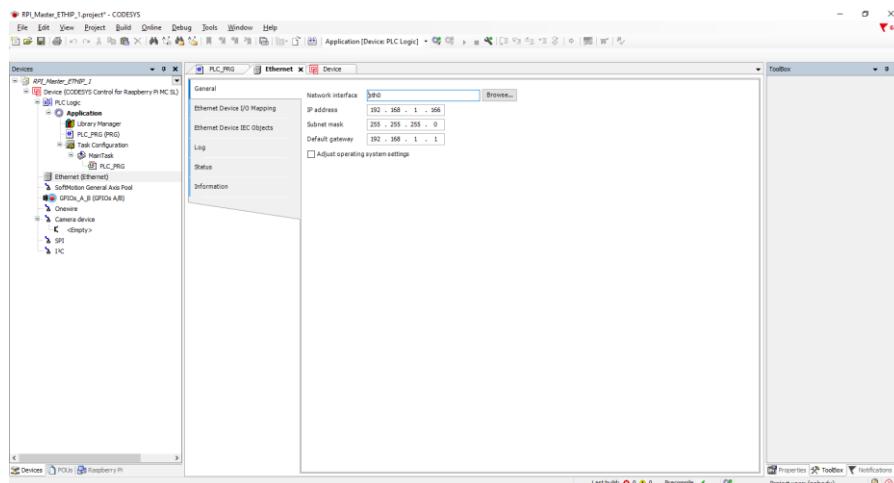
Adicionar porta Ethernet



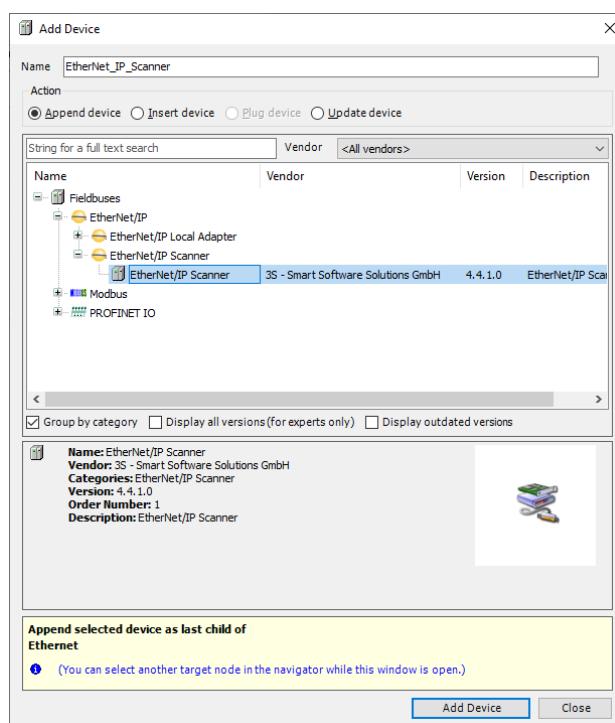
Depois deve configurar o ip do RPI, como mostrado abaixo. Neste caso, usei o browse para buscar alguma placa RPI disponível.



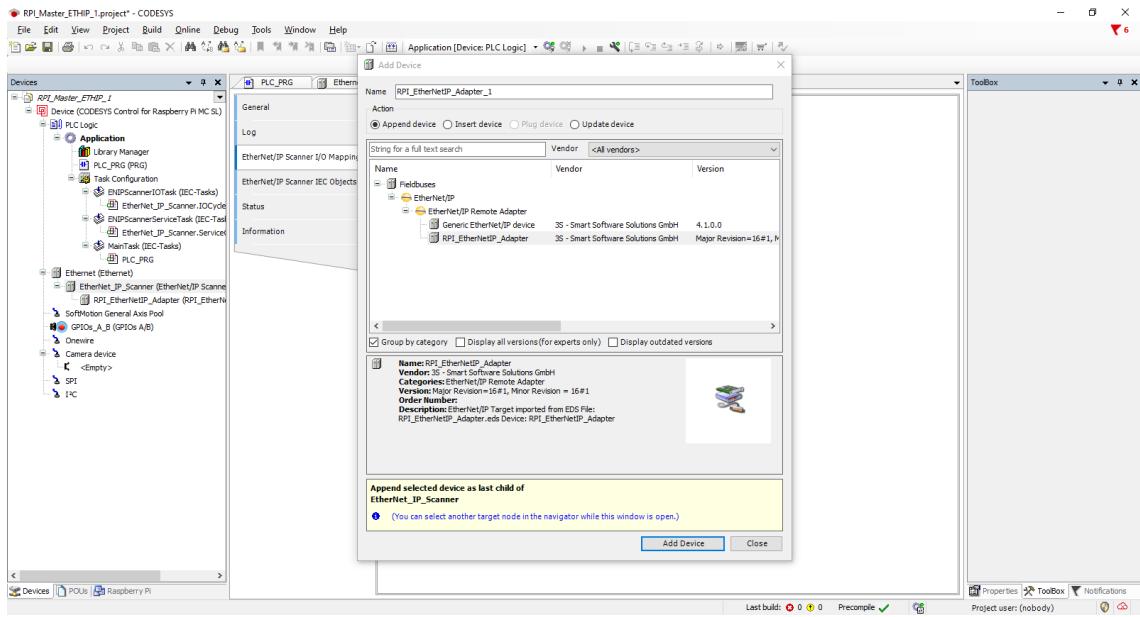
Após a configuração



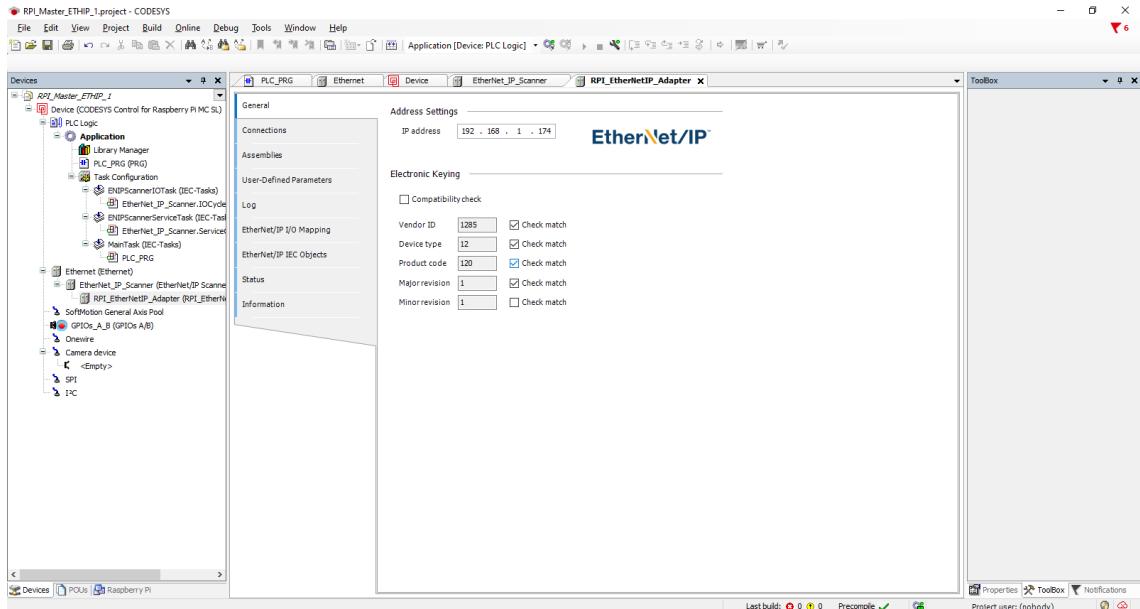
Agora deve selecionar a placa ethernet e adicionar um Scanner EthernetIP



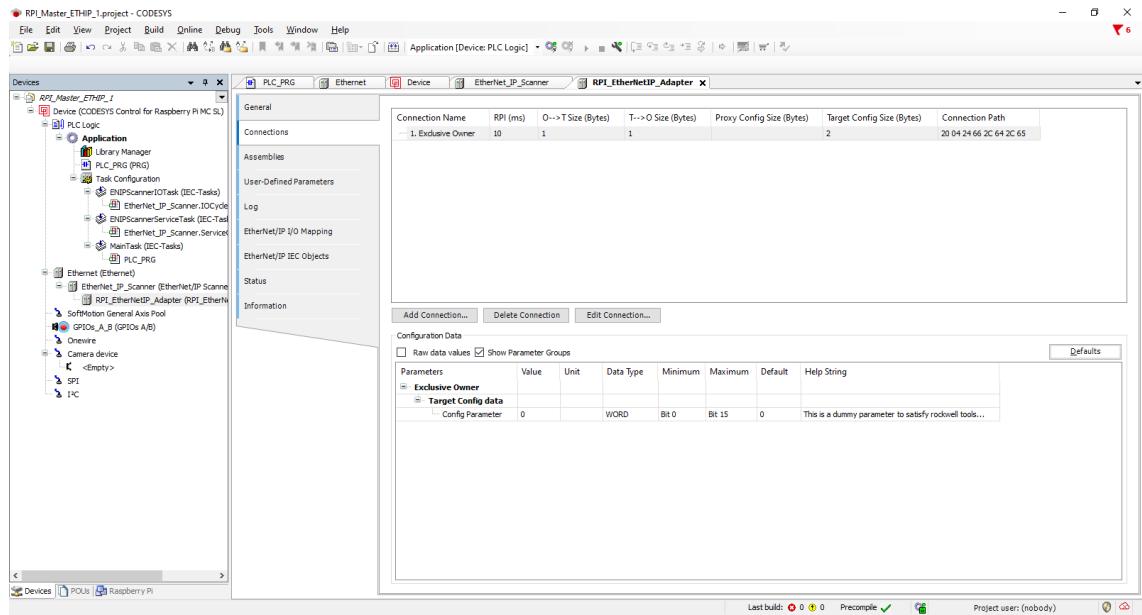
Agora deve adicionar os devices. Neste caso, como eu criei inicialmente um Adapter e instalei ele no repositório do Codesys. Ele já está disponível nesta configuração e eh ele que eu vou utilizar.



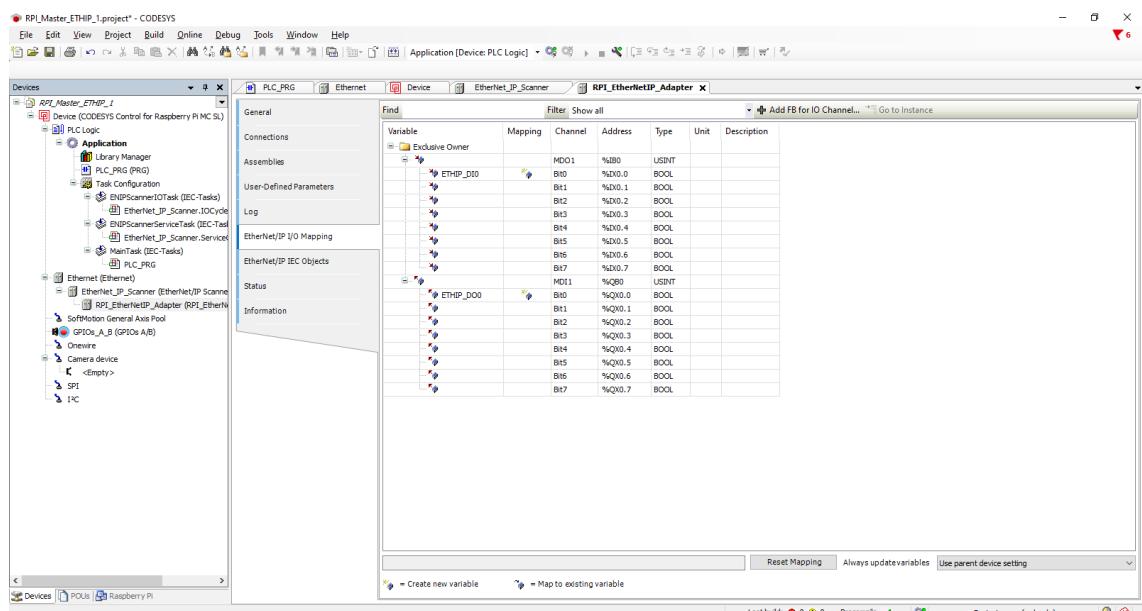
Então deve colocar o IP do Adapter.



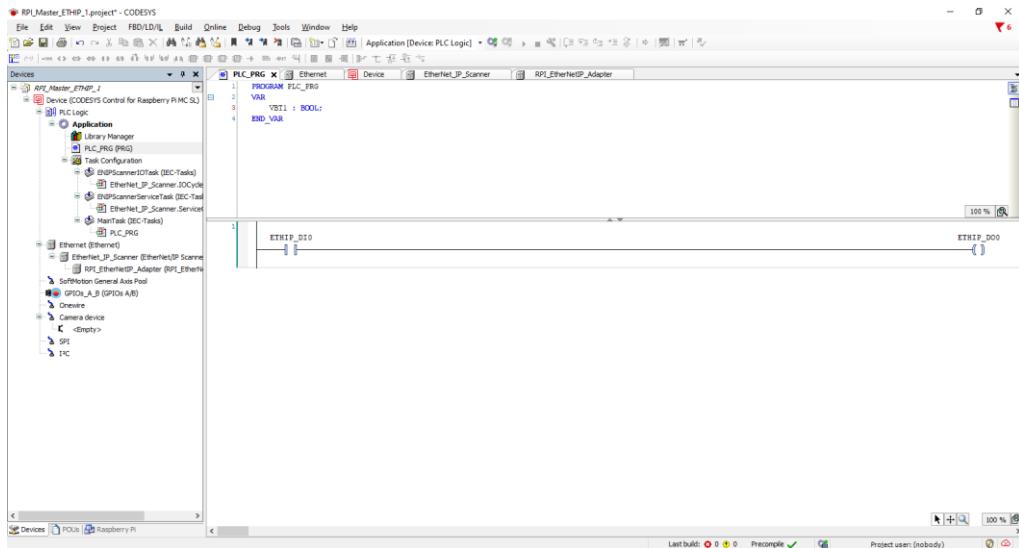
E como já foi importado o correto EDS do Adapter, a configuração de IO já esta pronta.



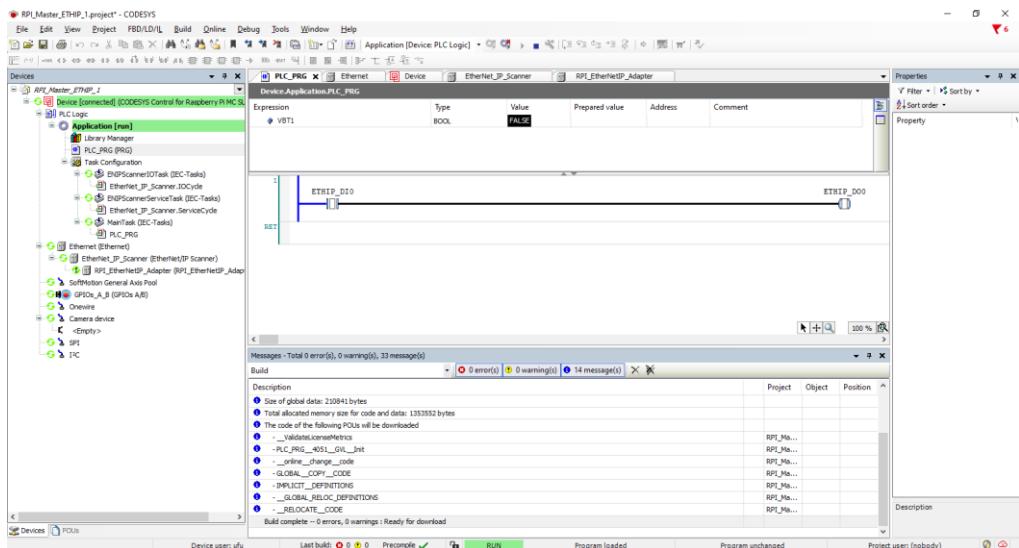
Os dados do scan vao estar nestas variáveis mapeadas (IO Mapping) definidas de acordo com a configuração acima.



Para funcionar os controladores, deve ter uma logica ladder mínima, senão ele mostra erro. Neste caso, fiz uma logica lendo e escrevendo o valor lido.



Se ele estiver funcionando... tudo ficara verinho... como a figura abaixo

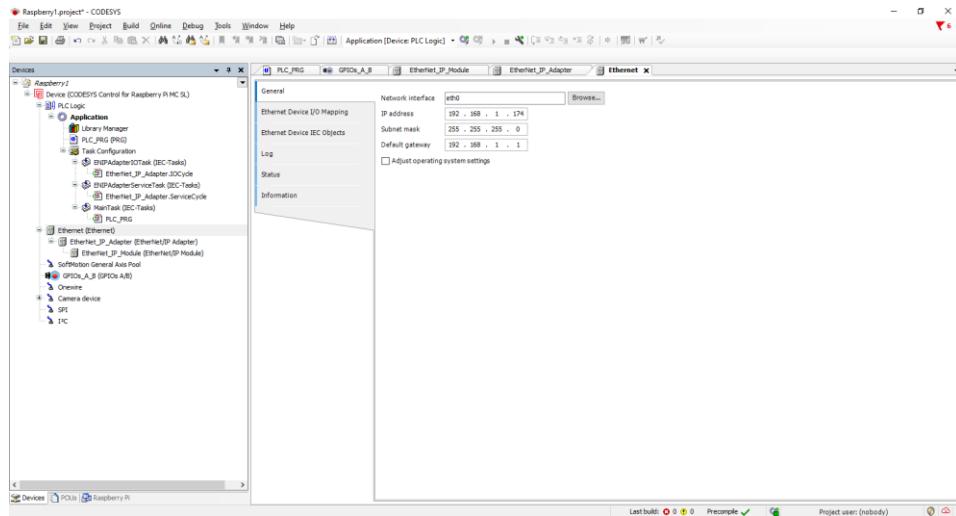


6 - Configurando Codebox como Escravo Ethernet/IP

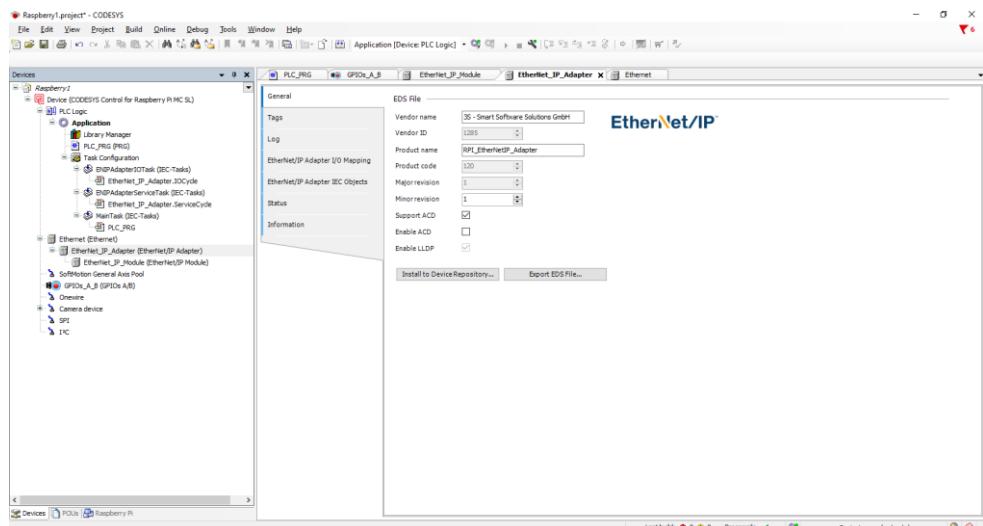
Aqui foi feito uma outra configuração (abri outro Codesys), no mesmo PC. Neste caso, agora com uma outra raspberry como escravo (esta estava ligada nos pinos de IO).

E novamente deve fazer a mesma coisa do que no mestre.

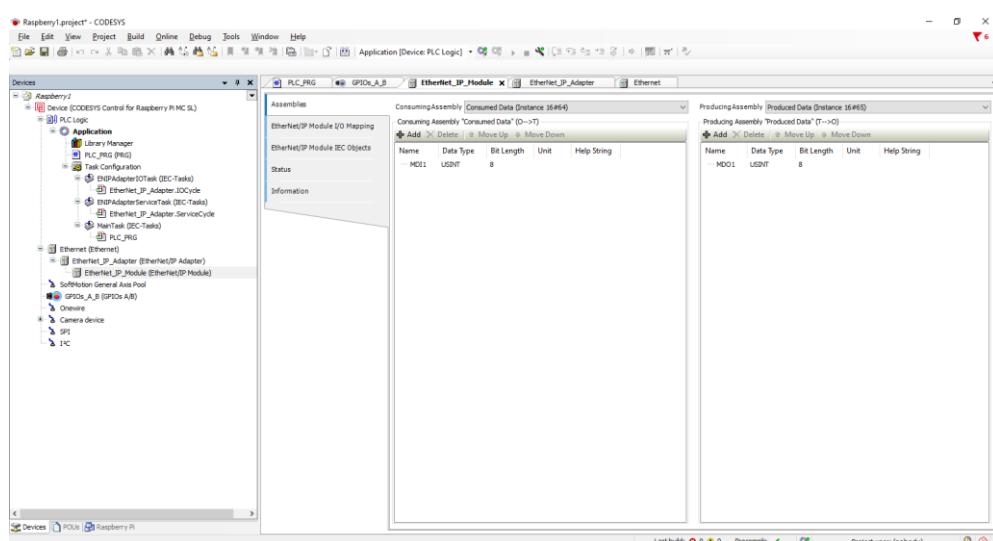
Criar porta ethernet



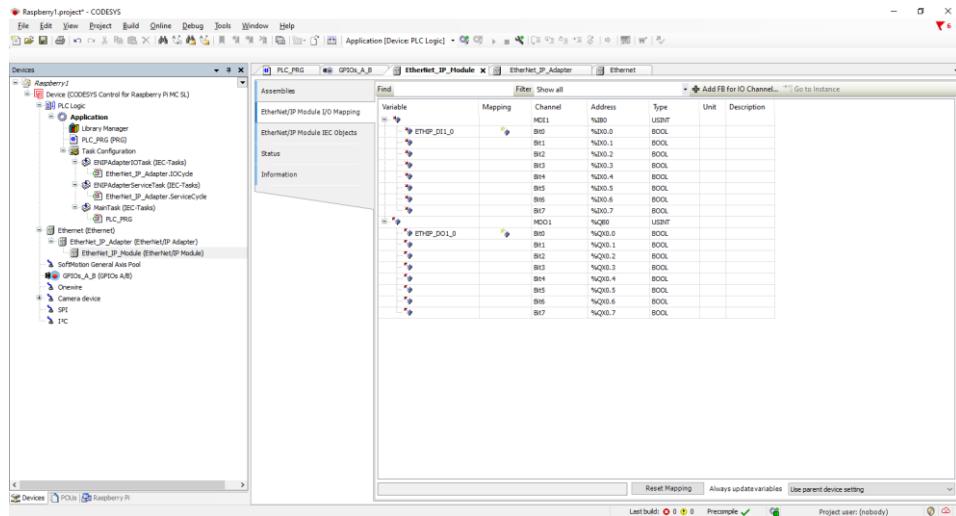
Criar um ethernet/IP Adapter (pois agora ele vai ser escravo). Importante nesta tela eh o botão Install to device repositor e Export EDS File. Mas isso deve ser feito depois que for feito o próximo passo de configurar os dados disponíveis do escravo.



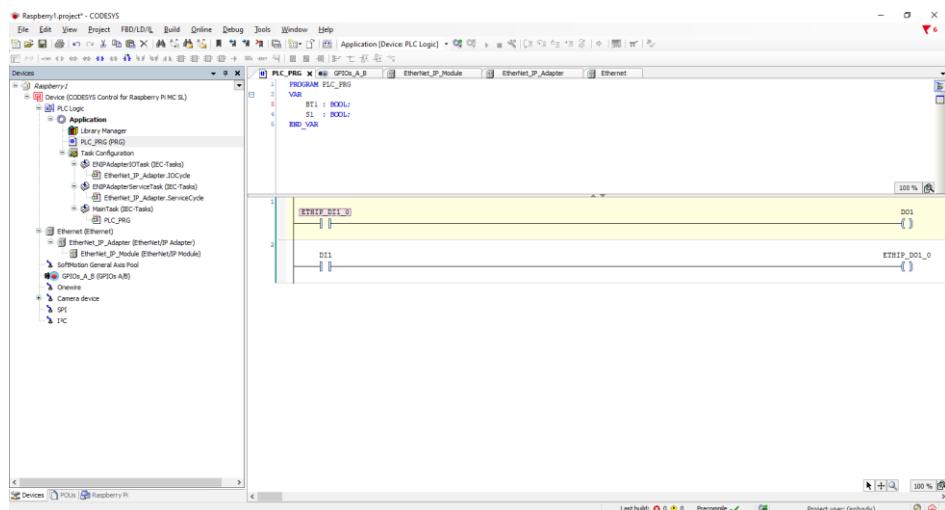
E criar um ethernet/IP Modulo. Neste caso eu escolho a configuração de Produtor e consumidor que eu quiser. Escolhi 1 byte de entrada e 1 byte de saída (onde somente tenho um bit de entrada e um bit de saída associado em cada 1).



Aqui estão os pontos mapeados



Aqui está a ladder.



Importante é agora salvar a configuração no repositório (botão install to device repository ou gerar o EDS file) para ser usado depois pelo Mestre.

7 - Configurando Codebox como Profinet IO device

Aqui o objetivo é fazer com a Raspberry PI trabalhe como um Profinet IO device e seja acessado pelo S71200 da Siemens (IO Controller)

Incialmente na raspberry deve ser feita uma atualização para suportar VLAN...deve seguir estes passos desta figura onde a raspberry deve ter acesso a internet.

If you use a Pi3B+ you Need the following steps:

Using VLAN-Tags need to be enabled:

```
sudo apt install vlan # install VLAN-Packetsudo modprobe 8021q # load 8021q Kernelmodul
```

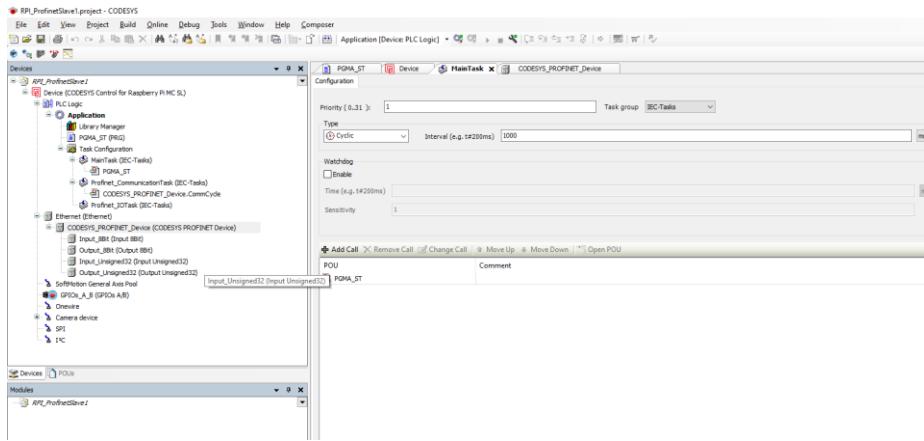
for having this in every Startup automatically:

```
sudo su -c 'echo "8021q" >> /etc/modules'
```

2RaspiProfinetController_PNDevice.project [100.18 KiB]

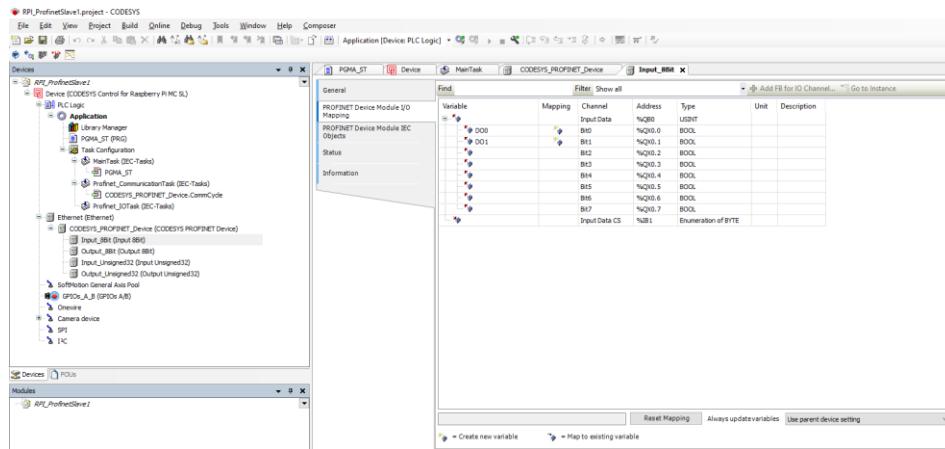
Após esta configuração (talvez seja somente feita uma vez)...Ligar a raspberry e pingar com o respectivo IP configurado (fixo manual).

Criar uma configuração com uma porta Ethernet. Então Add Device Profinet IO Device, forme mostrado na figura abaixo.

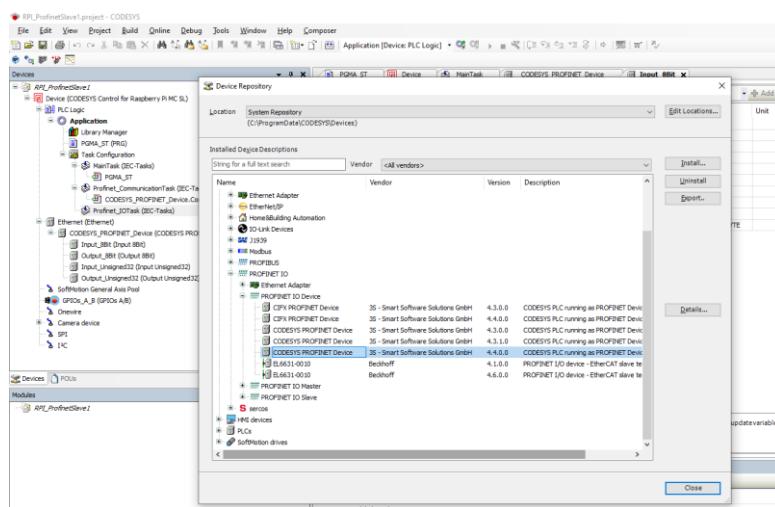
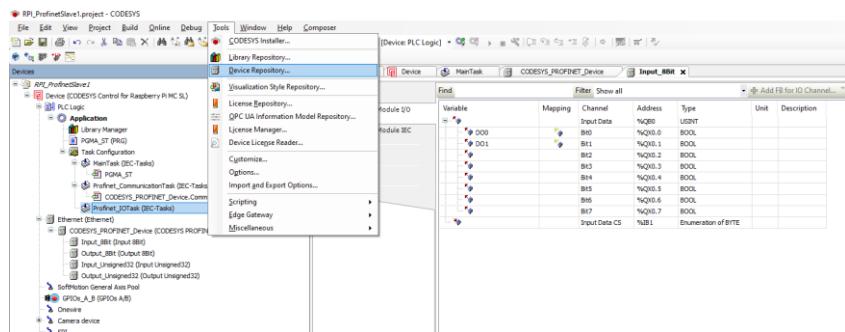


Agora é necessário incluir os pontos. Clicar em Add Device e escolher os datatype. No exemplo foi criado:

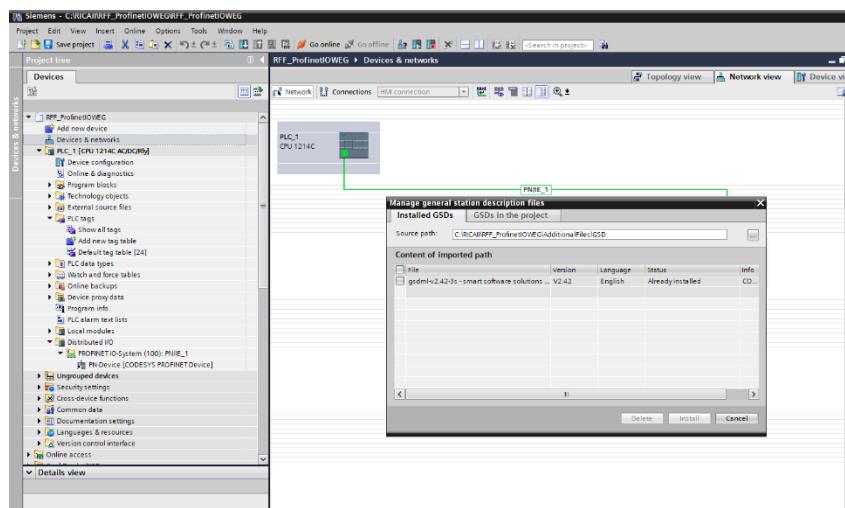
- entrada digital : 8Bits de Entrada
- Saida digital : 8 Bits de Saida
- Entrada analógica :Unsigned 32
- Saida Analogica: Unsigned 32

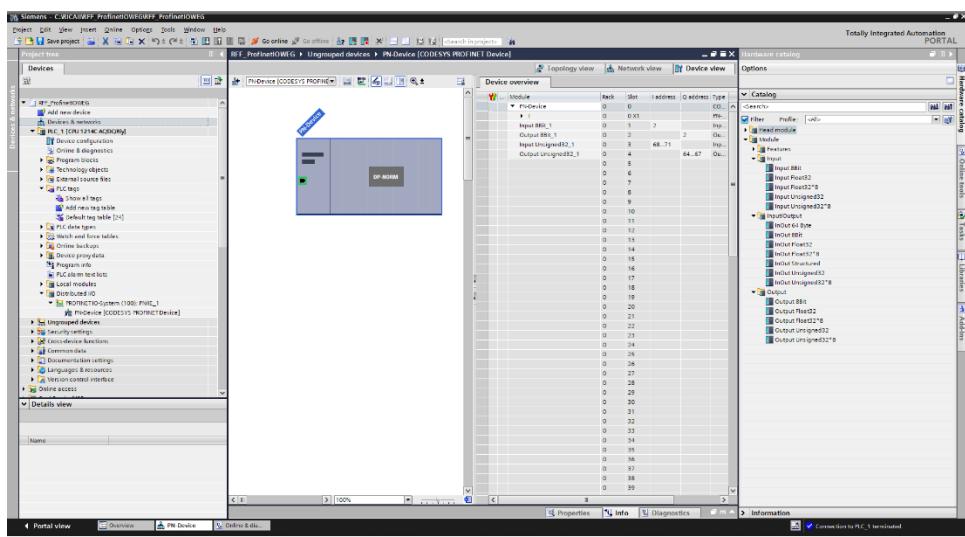
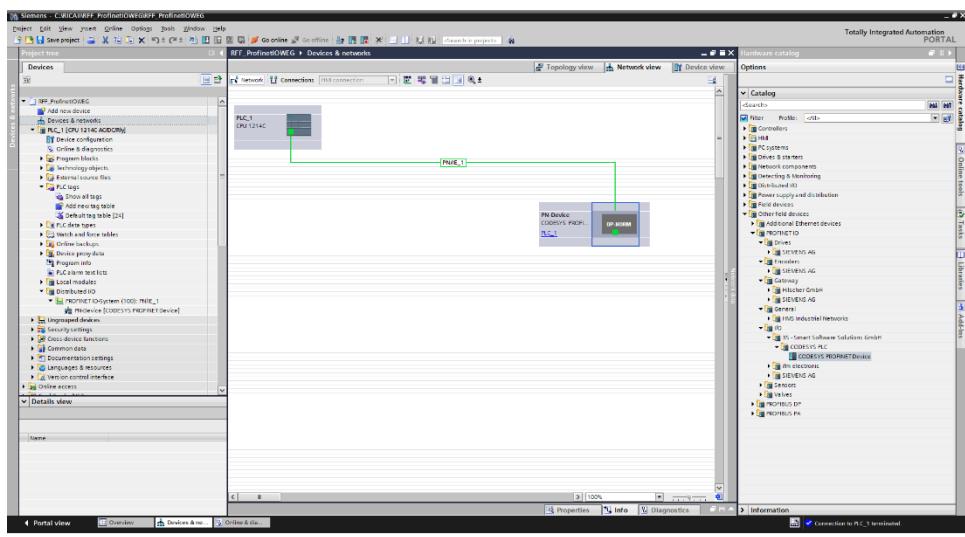


Por fim, é necessário criar o GSDML file para ser usado no Tia Portal. Para isso, vá em Tools/Device Repository/Profinet IO/Profinet IO Device. E pedir para exportar o arquivo.



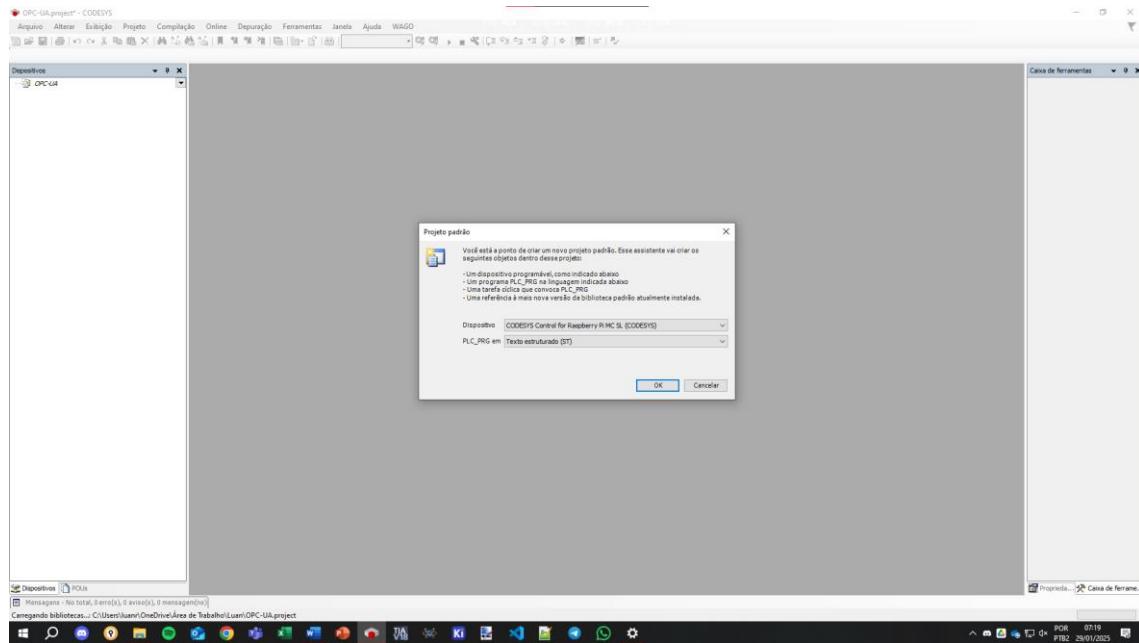
Depois é necessário instalar o GSD file no Tia Portal.



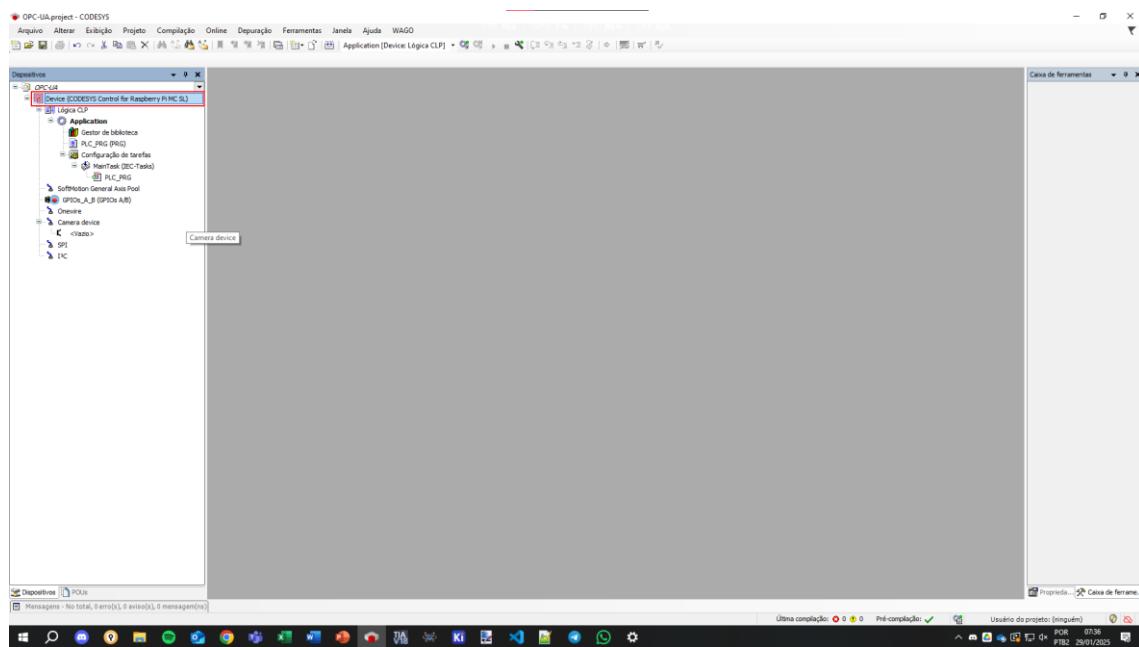


8 - Configurando OPC UA Server/Client

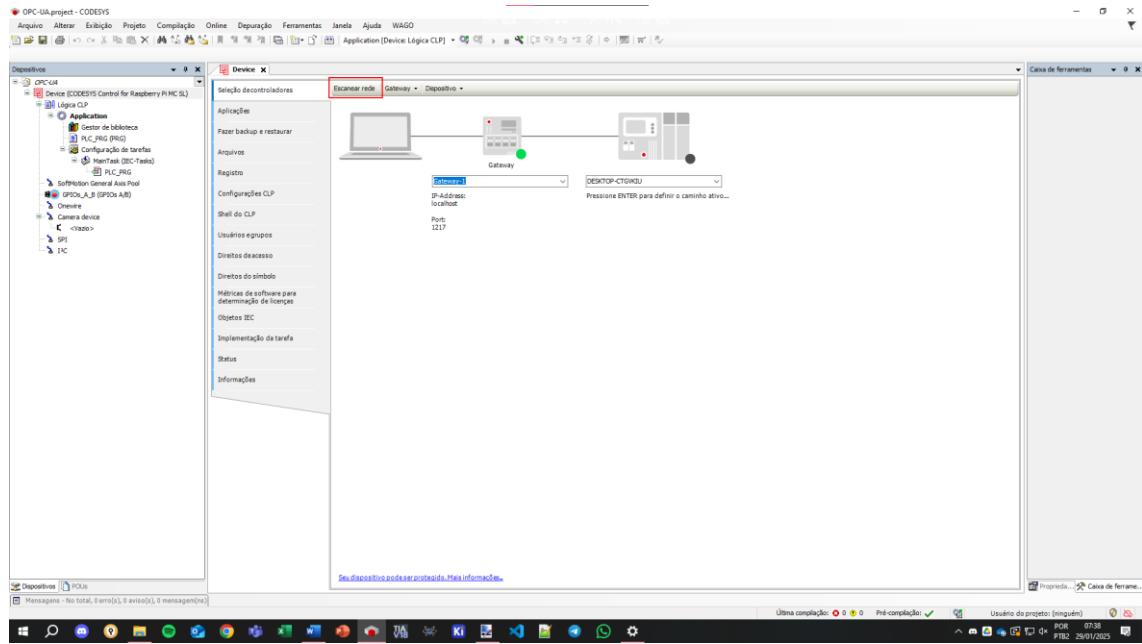
Após realizar a criação do projeto no codesys, seleciona-se o tipo da CPU, nesse caso **Raspberry PI MC SL** como mostrado na figura abaixo. Recomenda-se utilizar a linguagem ST, visto que ela permite uma implementação mais simples, com mais recursos comparado as demais linguagens disponíveis.



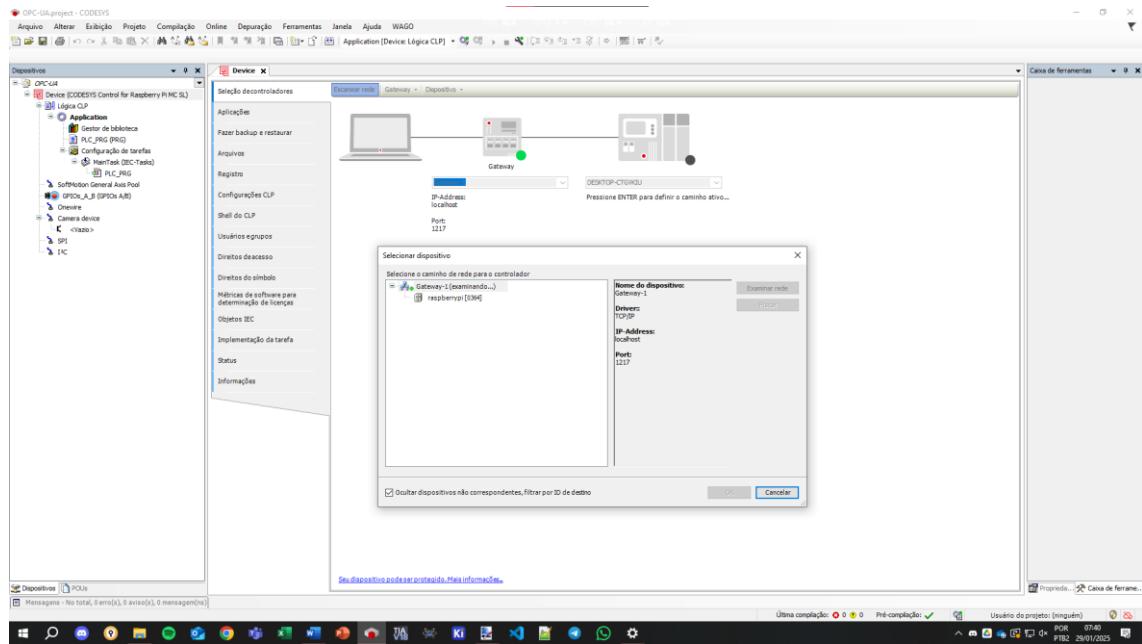
Com o projeto criado, agora basta conectar a CPU, com dois cliques em **Device**.



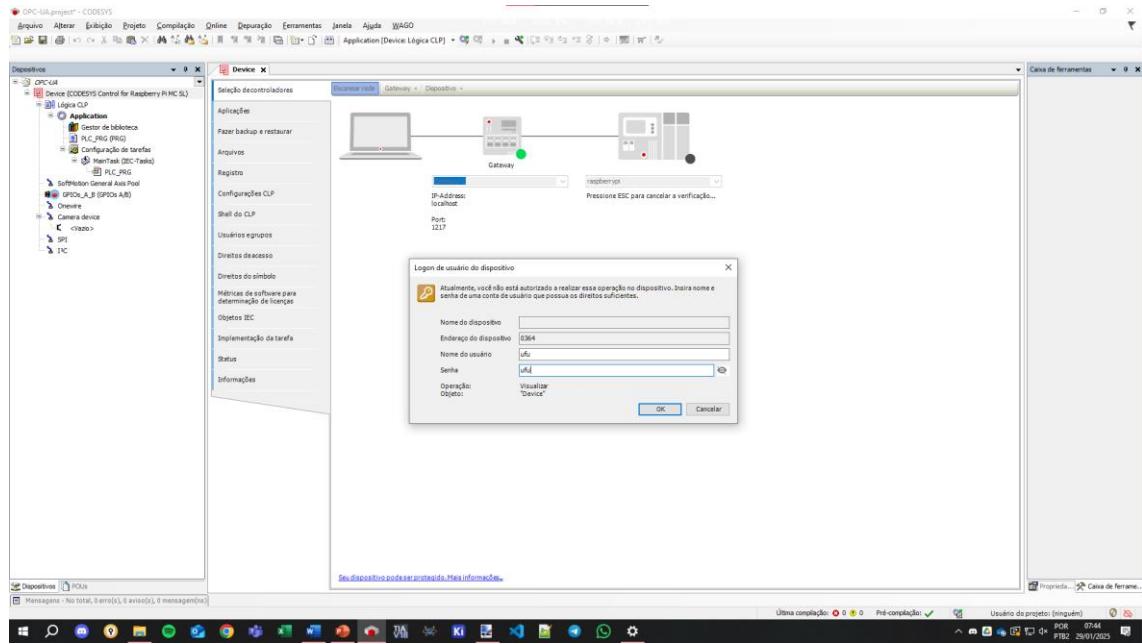
Para encontrar a CPU, é necessário realizar o scan da rede em **Escanear rede**.



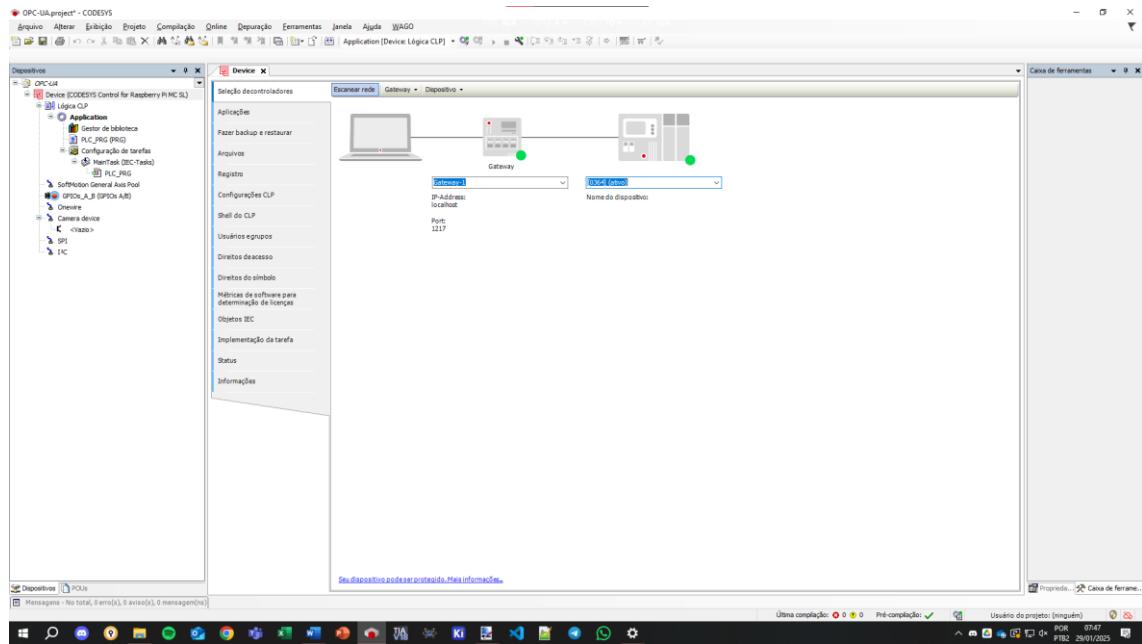
Feito isso, irá abrir a janela abaixo, onde basta selecionar a CPU, nesse caso, **raspberrypi [0364]** e dar um **OK**. O nome da CPU pode variar, mas o procedimento é o mesmo.



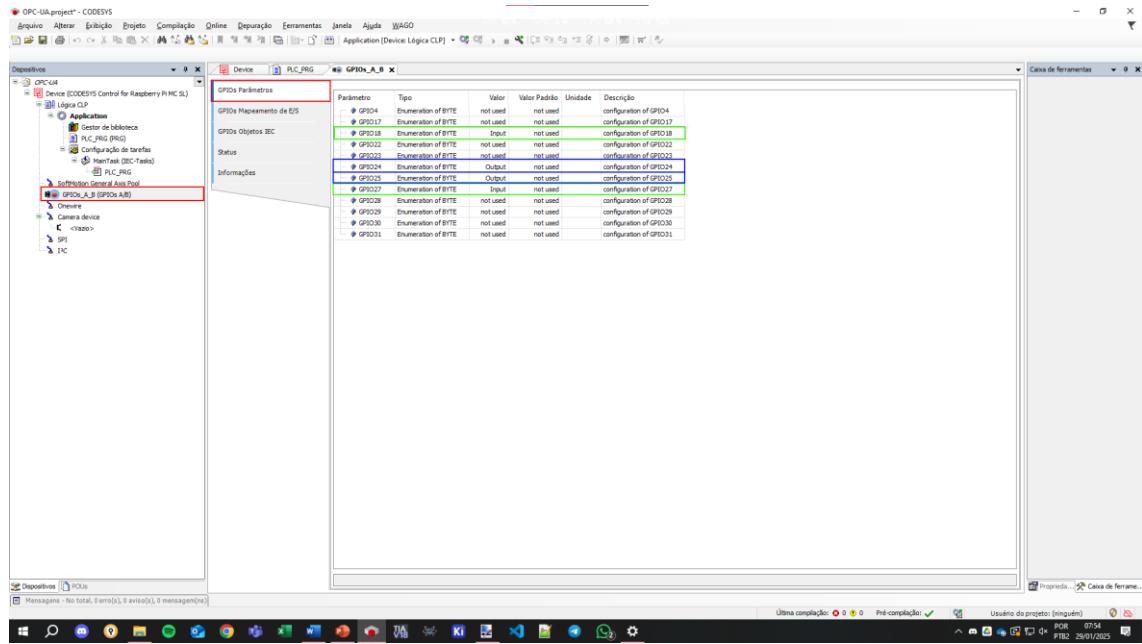
Com a CPU selecionada, o Codesys tentará fazer uma conexão, para isso deverá ser informadas as credenciais de acesso do dispositivo. Para o equipamento em questão, o usuário é **ufu** e a senha **ufu**.



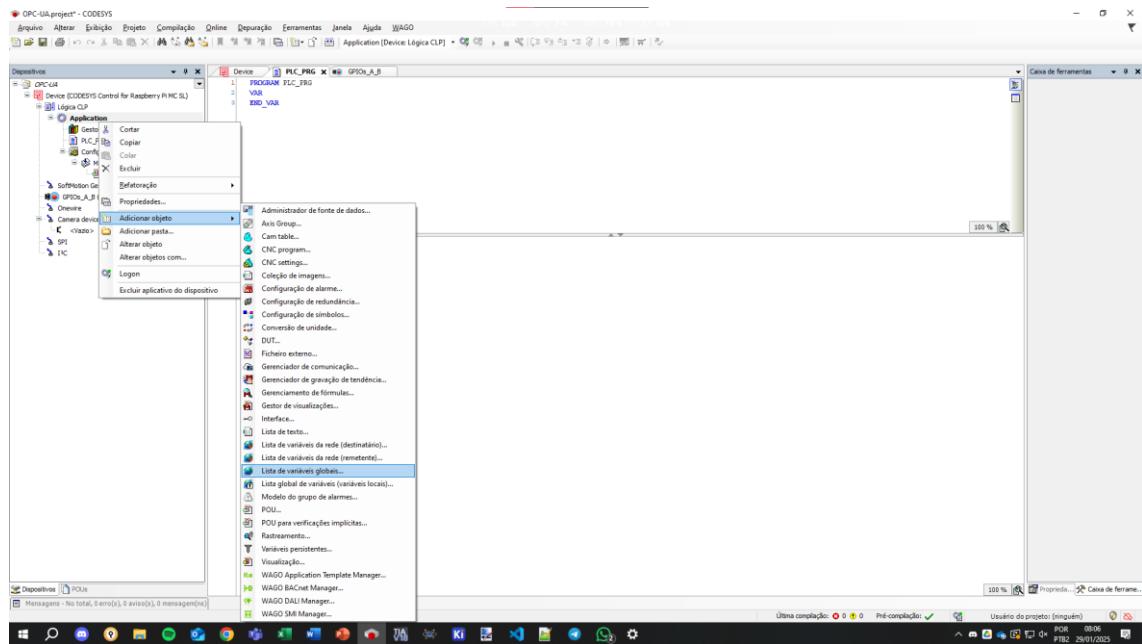
Se as credenciais informadas estiverem corretas, a conexão será realizada com sucesso, como mostra na figura abaixo

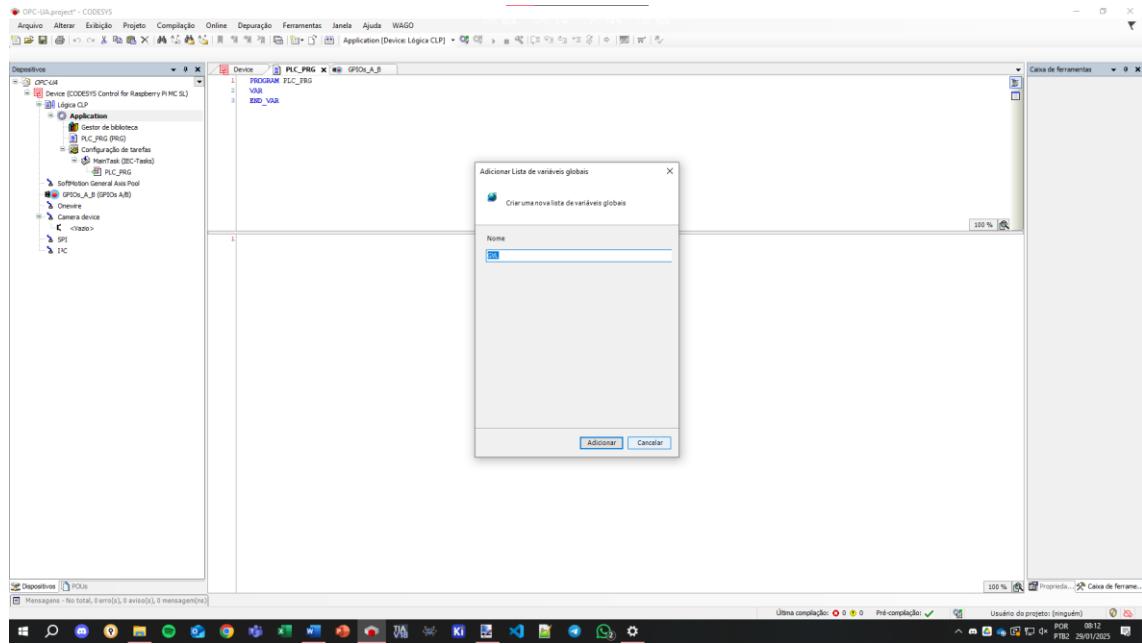


Realizada a conexão, agora será feito uma lógica simples em ST, onde será lido 2 botões físicos, e cada botão irá acionar seu respectivo led. Como é impossível escrever em entradas, será criado uma variável auxiliar, que acionará todas as saídas de uma vez só, e outra que desligará todas as respectivas saídas. Mas antes de programar a lógica, deve ser definido se o GPIO é do tipo entrada ou saída, clicando em **GPIOs_A_B** e em seguida **GPIOs Parâmetro**.

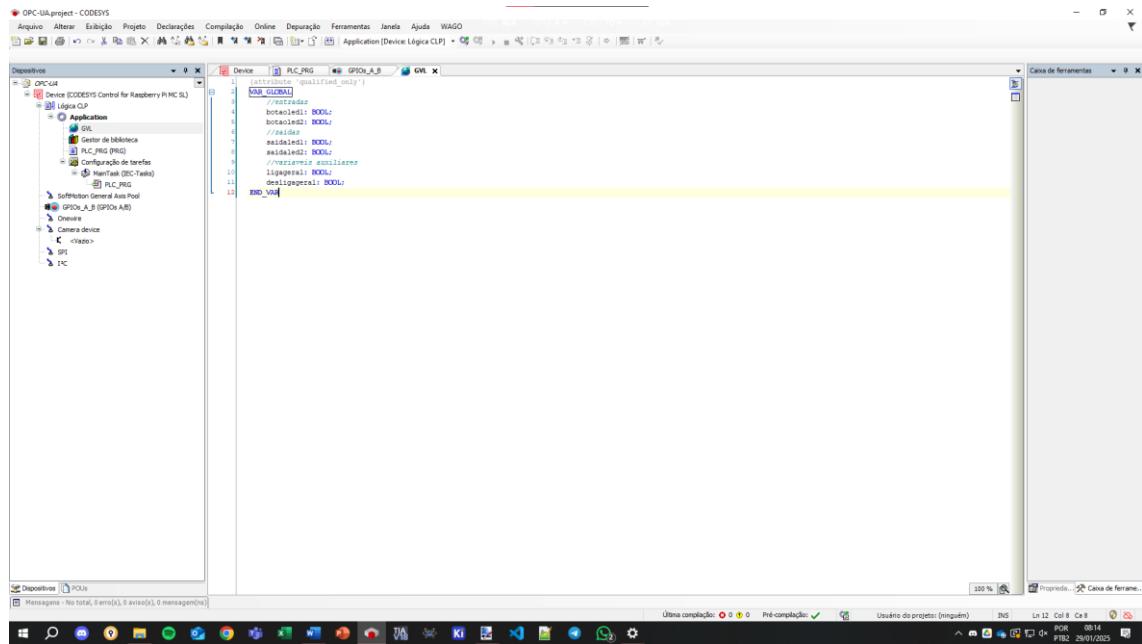


Feito essa configuração, agora é feita a declaração das variáveis da lógica de forma global. Para isso, deve-se clicar com o botão direito em **Application**, adicionar objeto, e adicionar uma **Lista de variáveis globais**, o nome da lista pode ser o padrão do Codesys, GVL.

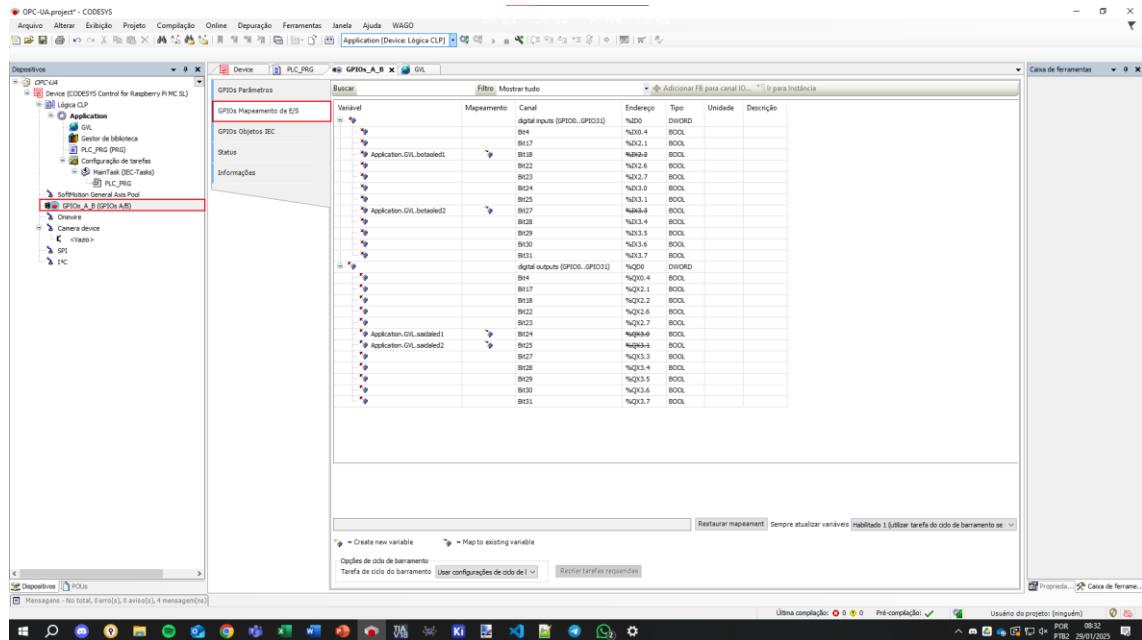




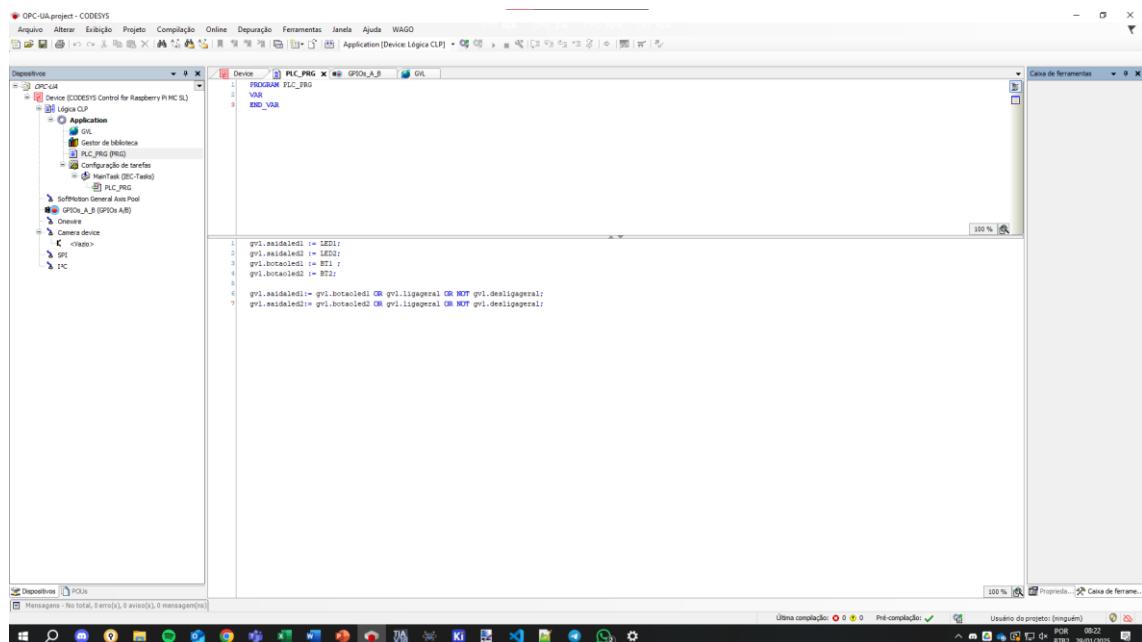
As variáveis para a lógica em questão ficaram dispostas como mostrado na figura abaixo, onde todas elas são do tipo booleano.



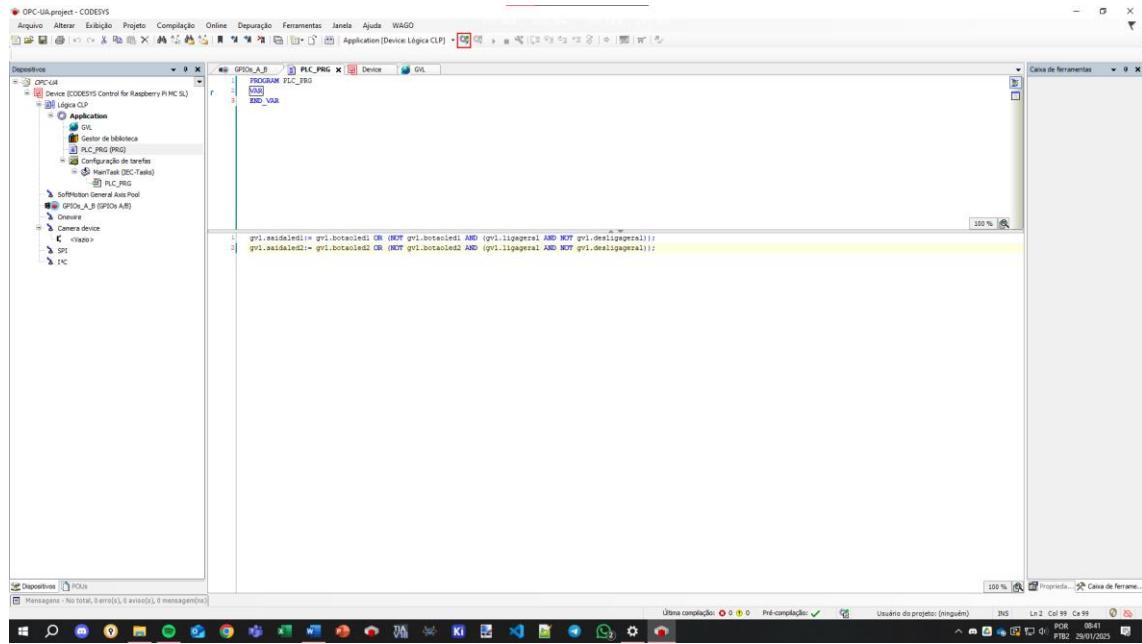
Com as variáveis declaradas, em GPIOs **Mapeamento de E/S para o BIT 24** será definido **Application.GVL.saidaled1** e **BIT 25 Application.GVL.saidaled2**, isso para as saídas, já para as entradas **BIT 18** será definido **Application.GVL.botaoled2** e **BIT 27** define-se **Application.GVL.botaoled2**, como mostra a figura abaixo.



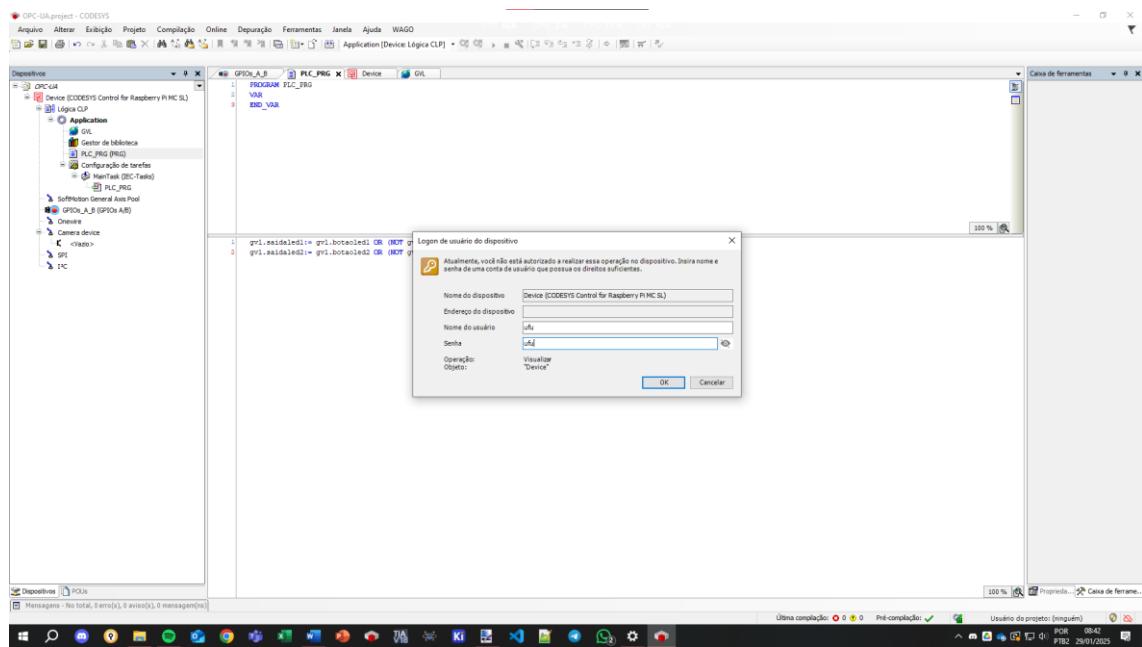
Em seguida, foi feita a lógica abaixo em ST, com as condições para os leds ligarem e desligarem.



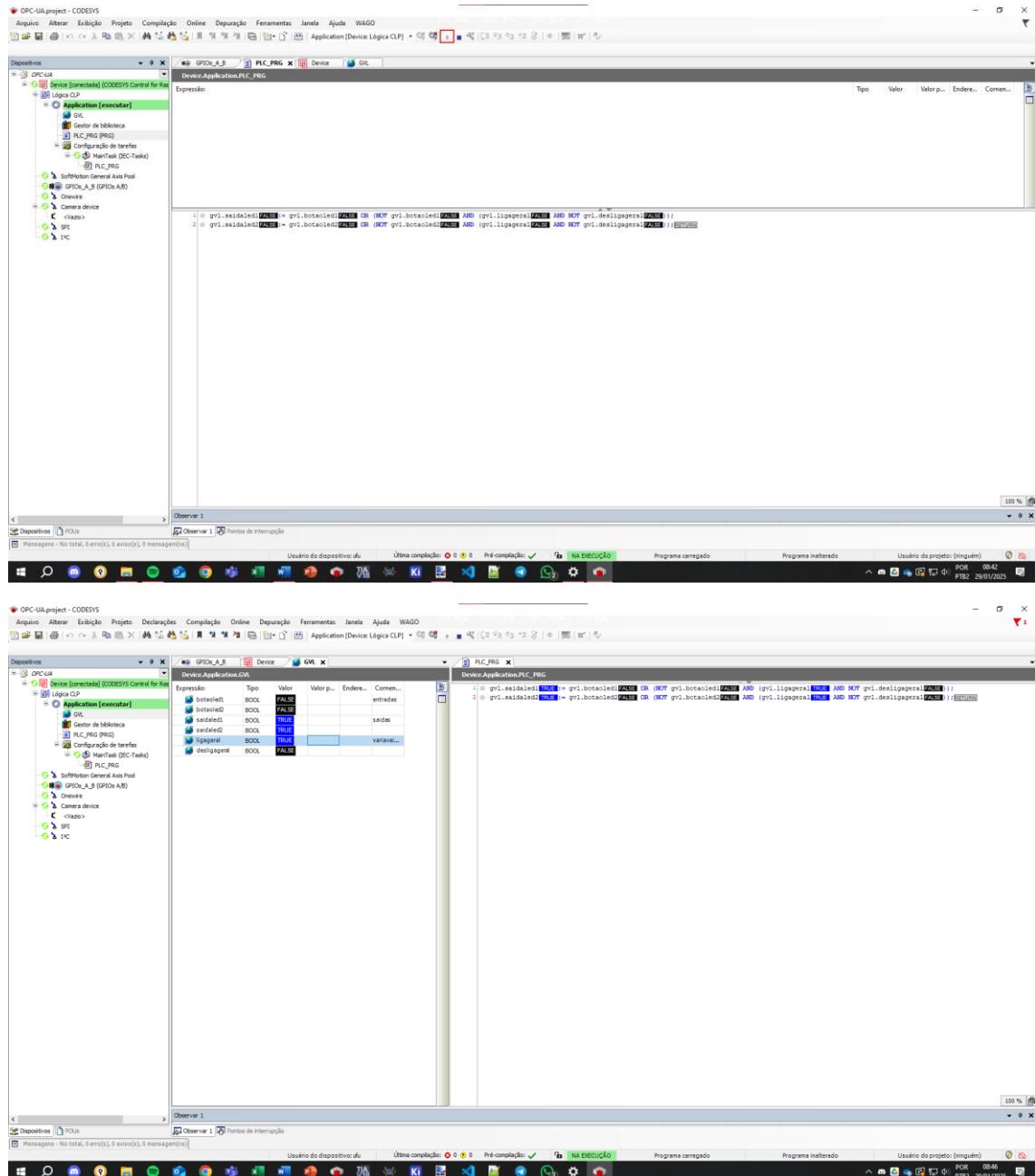
Com a lógica pronta, é só fazer o login e descarregar o programa para o clp clicando no botão circulado de vermelho.



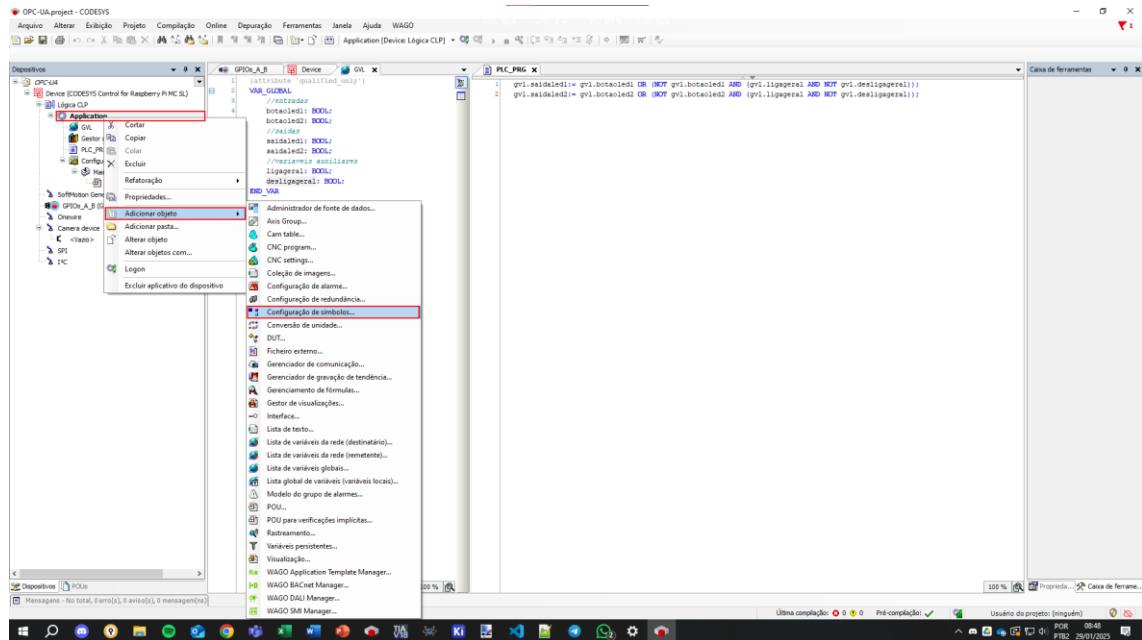
Ao clicar será pedido novamente as credenciais da CPU.



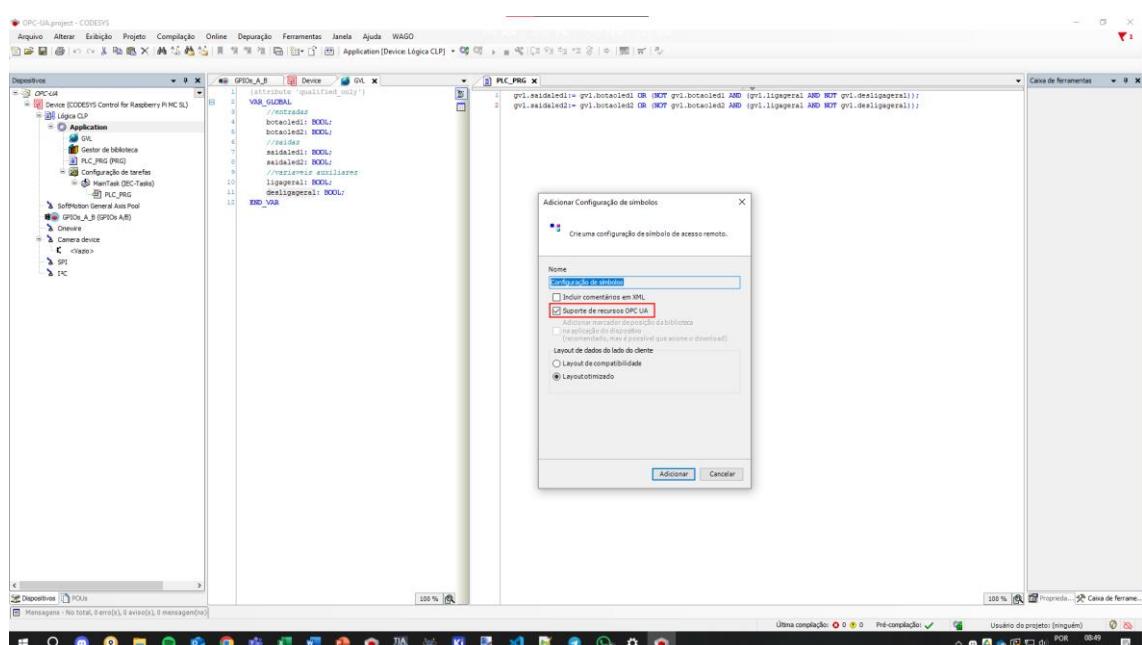
Feito o download a lógica já está rodando na CPU. Uma observação é conferir se a CPU está em mudo RUN, sempre conferir apertando no play circulado em vermelho na figura abaixo.



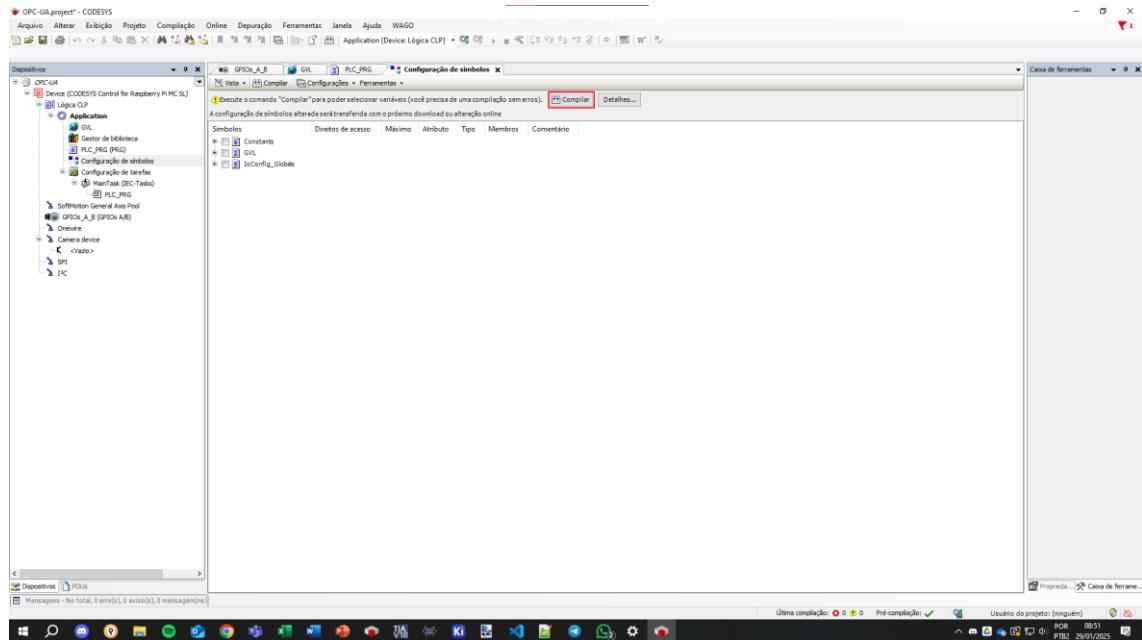
O próximo passo é configurar o servidor OPC UA, adicionando um objeto do tipo configuração de símbolos.



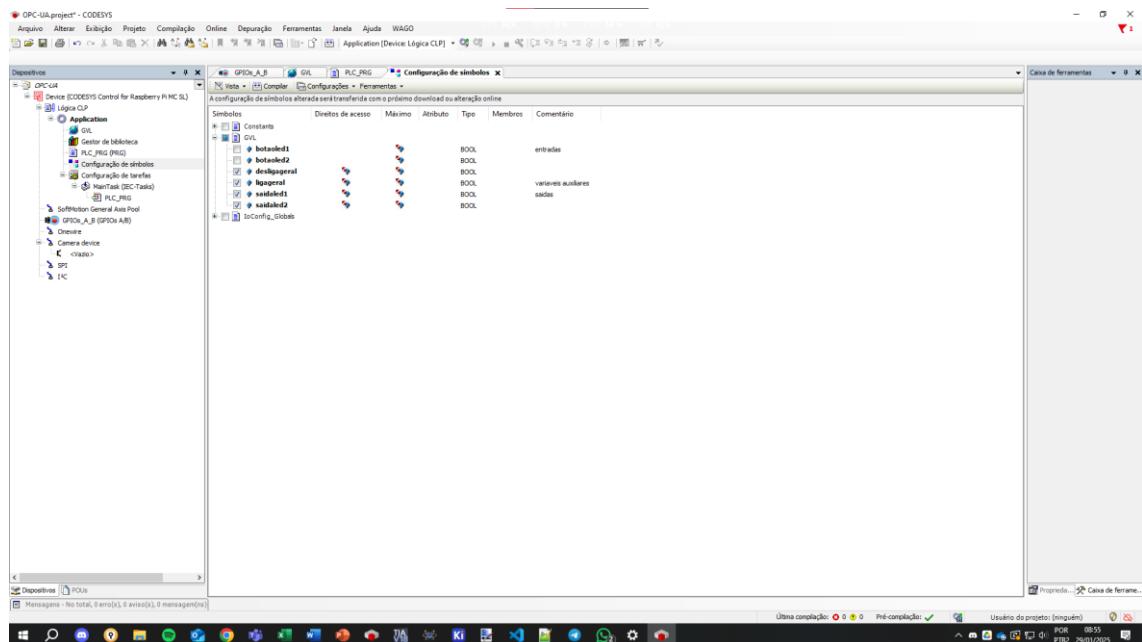
Sempre deve ser verificado se está selecionado o suporte para OPC UA.



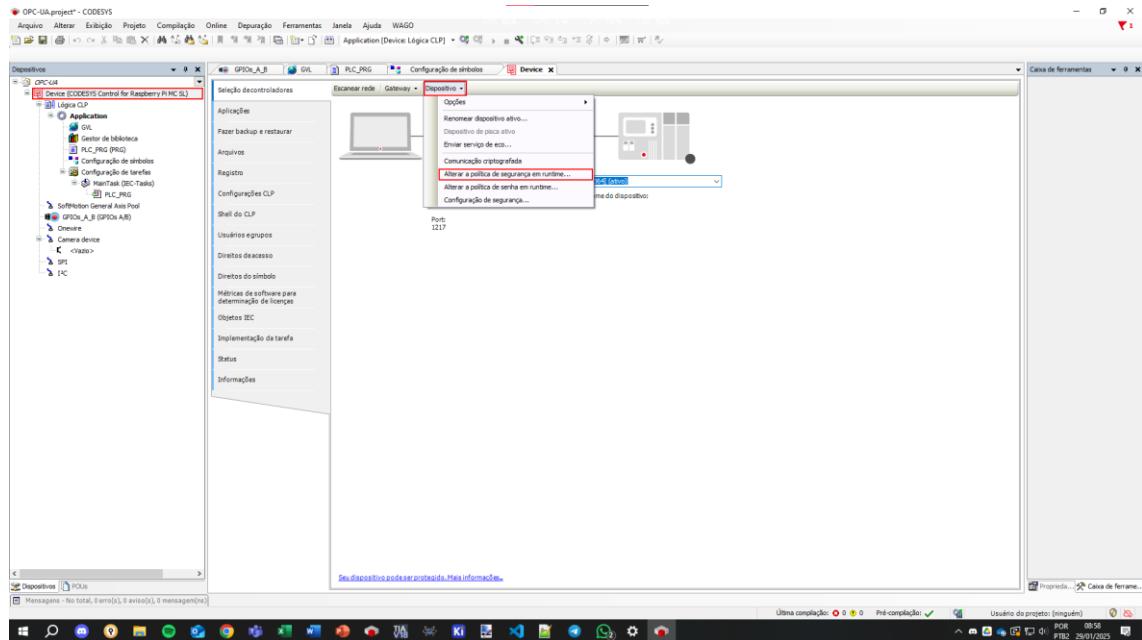
Após adicionar, o próximo passo é compilar.



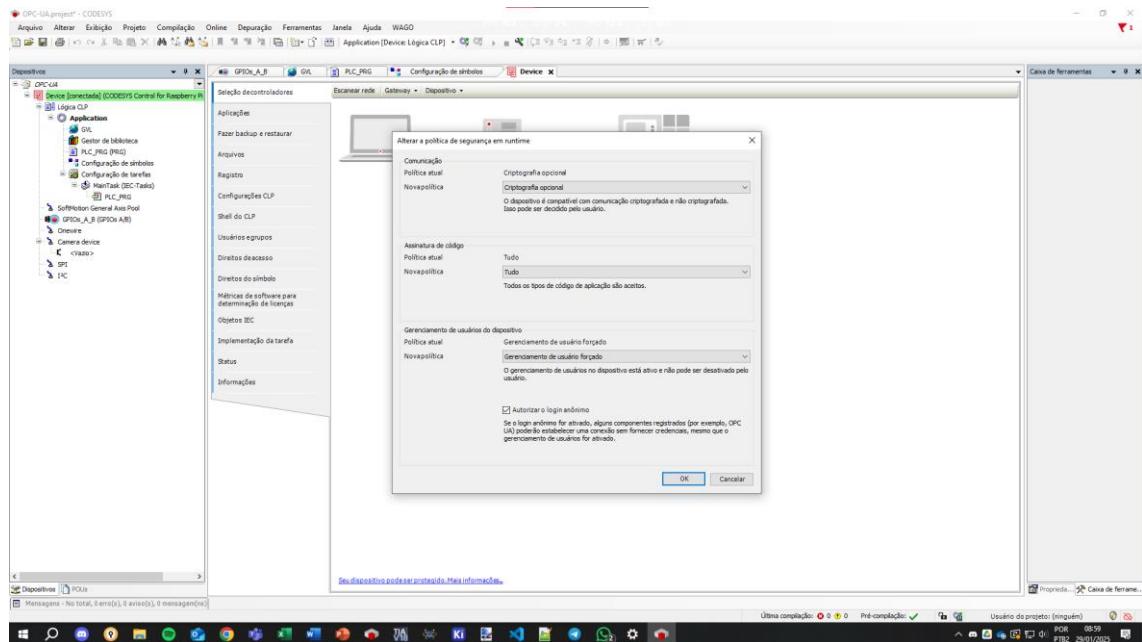
Com a compilação finalizada basta selecionar o **GVL** se quiser disponibilizar todas as variáveis via OPC UA, ou poderá disponibilizar variáveis específicas dentro do **GVL** clicando no + e selecionando variável por variável, como foi feito na figura abaixo.



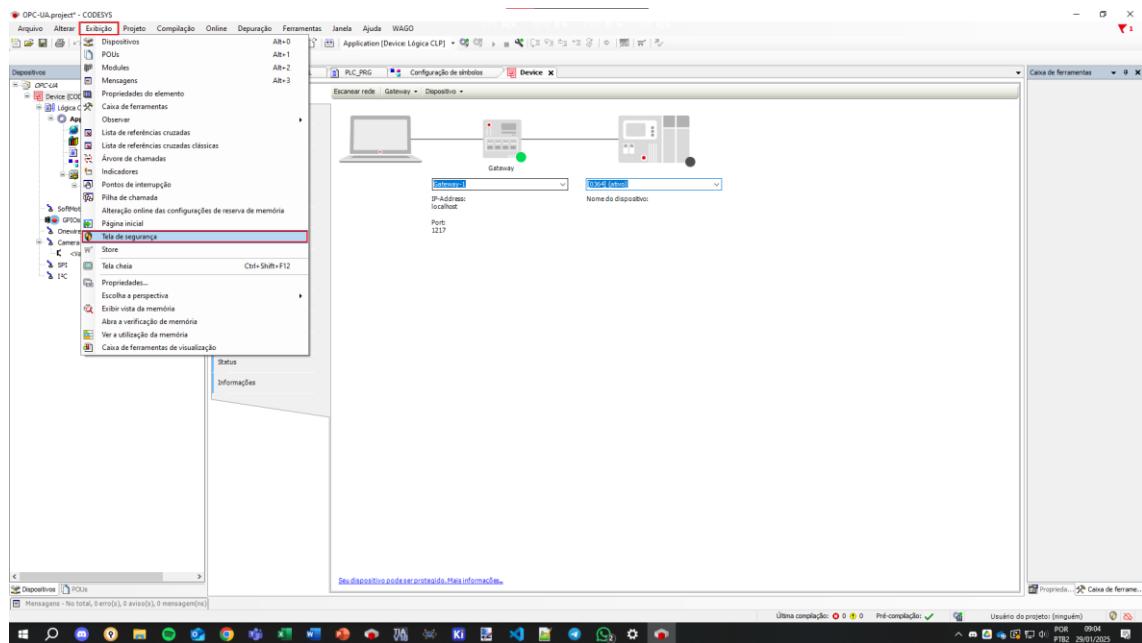
Feito esse procedimento, agora precisa ser configurado algumas permissões para o servidor OPC UA funcionar. Com a CPU em modo STOP, clique duas vezes em **Devices** e depois em **Dispositivos** e entrar em **Alterar a política de segurança em runtime...**



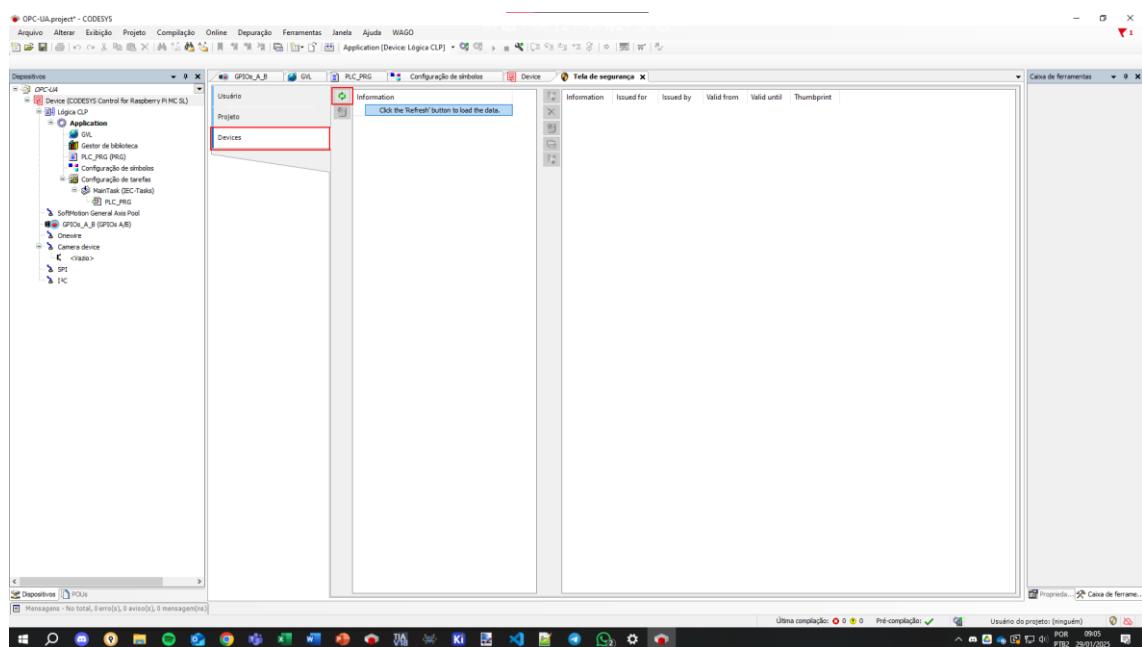
Nessa página de política de segurança, é importantíssimo que **Novapolítica** esteja em **Gerenciamento de usuário forçado**, e que **Autorizar login anônimo** esteja marcado, a Raspberry não suporta conexão com usuário e senha no cliente OPC UA, então a conexão vai ser sempre abortada caso essa opção não esteja marcada.



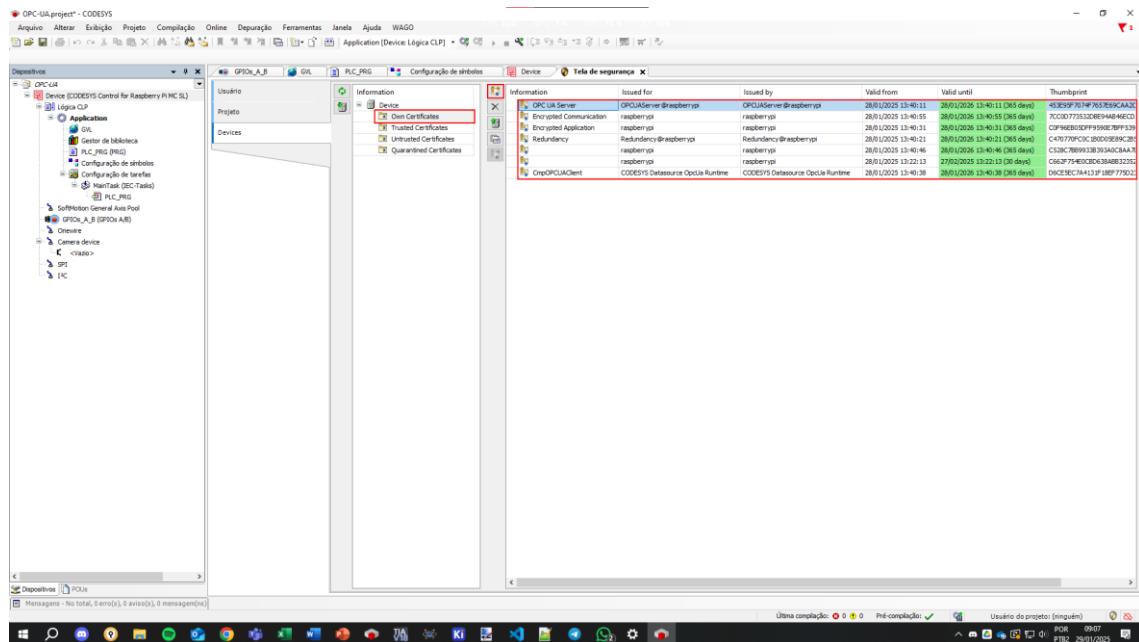
Feito isso, verificar os certificados SSL, é recomendado todos os certificados estarem válidos para que a conexão seja feita sem problemas de licença. Esses certificados podem ser encontrados na aba de **Exibição e Tela de segurança** como mostrado na figura abaixo.



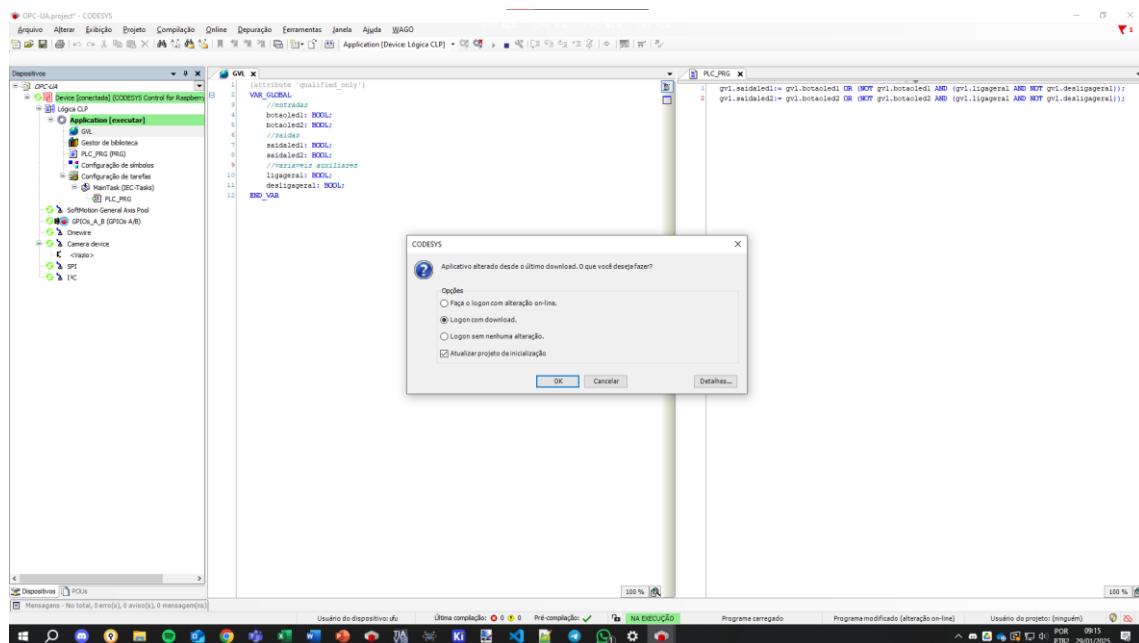
Clique em **Devices** depois no botão verde de atualizar.



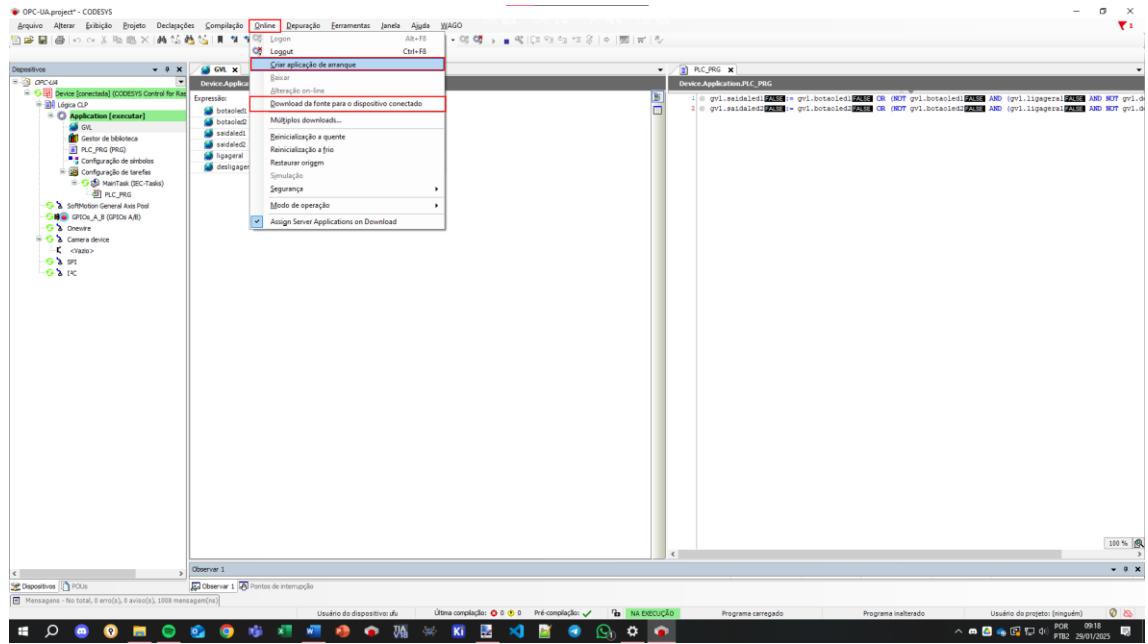
Clique em **Own Certificates**, isso vai abrir uma janela com diversos certificados, se todos estiverem válidos, não precisa renovar, mas se estiverem expirados, é necessário clicar em **novo**, que é esse pequeno ícone de certificado circulado de vermelho, e selecionar **Key length (bit)** para 2048, em **Validity period (days)** coloque 365 para um ano de licença ou a quantidade de dias de sua preferência. Após renovar, e todos os certificados estiverem verdes, agora basta realizar o download para a CPU.



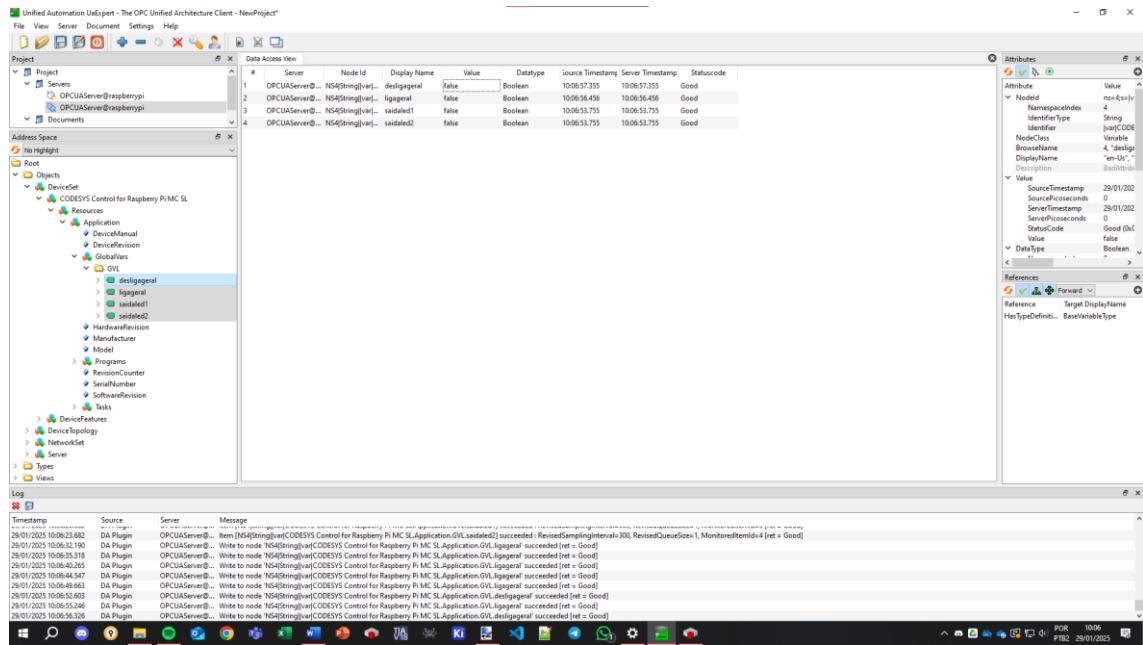
Para fazer o download, basta teclar **ALT + F8**, e selecionar **Logon com download**.



Feito o download, agora deve-se colocar a CPU em modo RUN clicando no símbolo de play. Após esses procedimentos o servidor irá funcionar perfeitamente, mas sempre que a CPU for reinicializada vai precisar fazer o download do programa. Para evitar isso, pode-se criar uma aplicação e mover para dentro da CPU. Para fazer esse procedimento, clique em **Online**, depois em **Criar aplicação de arranque**, e por fim, **Download da fonte para o dispositivo conectado**, como mostrado na figura abaixo.



Para verificar se de fato o servidor está funcionando, poderá ser utilizado qualquer cliente OPC UA, no teste abaixo foi utilizado o UAExpert, que é o software recomendado para ser utilizado pela própria Codesys.



Agora, será feito uma configuração da Raspberry, servindo como um cliente, para um servidor OPC UA, feito em Python.

Define Peso – Real, Contador – Inteiro, Ledx - Booleano, BTx – Booleano.

Na figura abaixo, tem-se o servidor OPC UA em Pyhton.

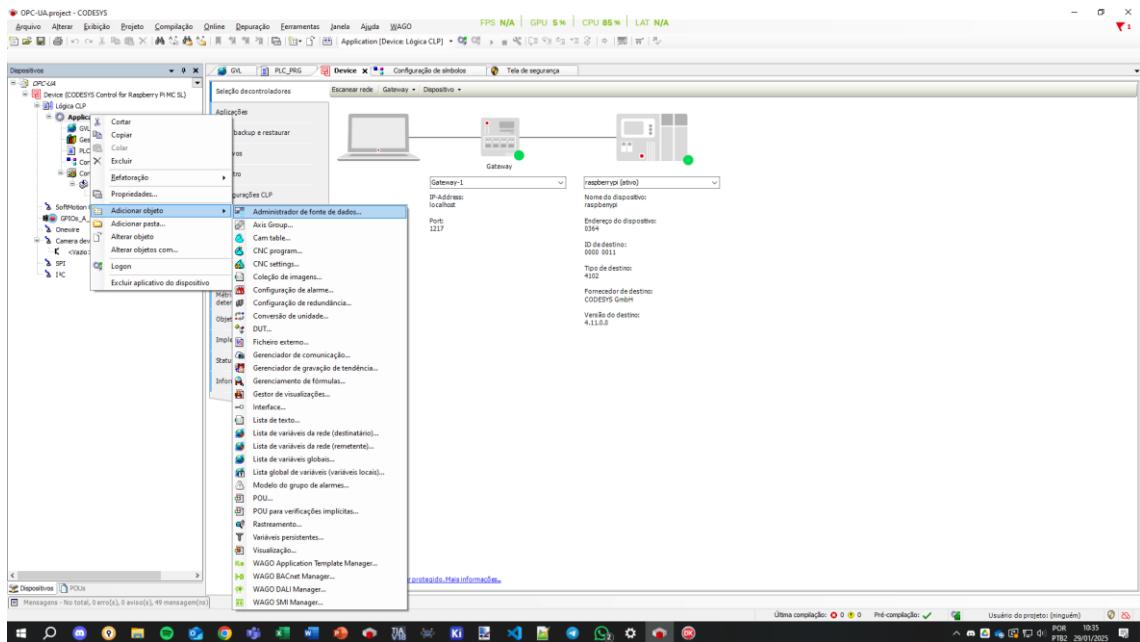
```

1 from opcua import Server
2 from opcua.ua import VariantType, NodeId
3 import time
4 import random
5
6 server = Server()
7 server.set_endpoint("opc.tcp://0.0.0.0:4840/freeopcua/server/")
8 namespace_idx = server.register_namespace("Namespace4")
9 objects_node = server.get_objects_node()
10 server_folder = objects_node.add_object(NodeId(1000, namespace_idx), "ServerVariables")
11
12 variables = {
13     "Define Peso": server_folder.add_variable(NodeId(1, namespace_idx), "Define Peso", round(random.uniform(0.1, 100.0), 2), varianttype=VariantType.Float),
14     "Contador": server_folder.add_variable(NodeId(2, namespace_idx), "Contador", random.randint(0, 1000), varianttype=VariantType.Int32),
15     "Btx": server_folder.add_variable(NodeId(3, namespace_idx), "Btx", random.choice([True, False]), varianttype=VariantType.Boolean),
16     "BTx": server_folder.add_variable(NodeId(4, namespace_idx), "BTx", random.choice([True, False]), varianttype=VariantType.Boolean)
17 }
18
19 for variable in variables.values():
20     variable.set_writable()
21
22 server.start()
23 print("Servidor OPC UA iniciado em opc.tcp://0.0.0.0:4840/freeopcua/server/")
24
25 try:
26     while True:
27         time.sleep(1)
28
29 except KeyboardInterrupt:
30     print("\nEncerrando o servidor OPC UA...")
31 server.stop()
32 print("Servidor encerrado.")

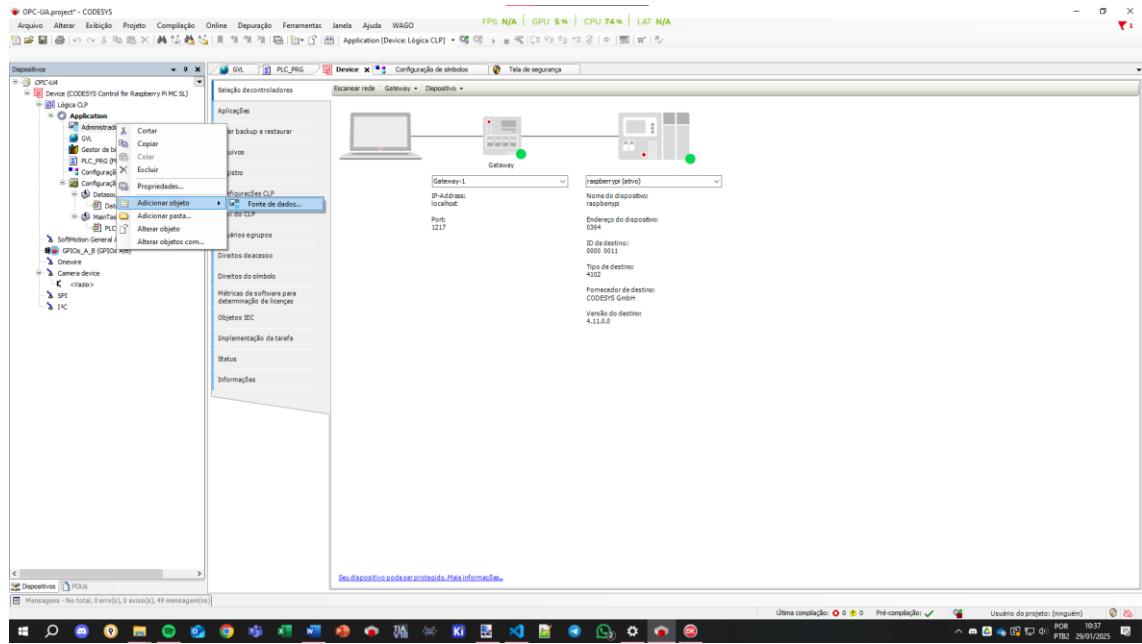
```

Rodando esse servidor, o próximo passo é fazer a conexão pelo Codesys.

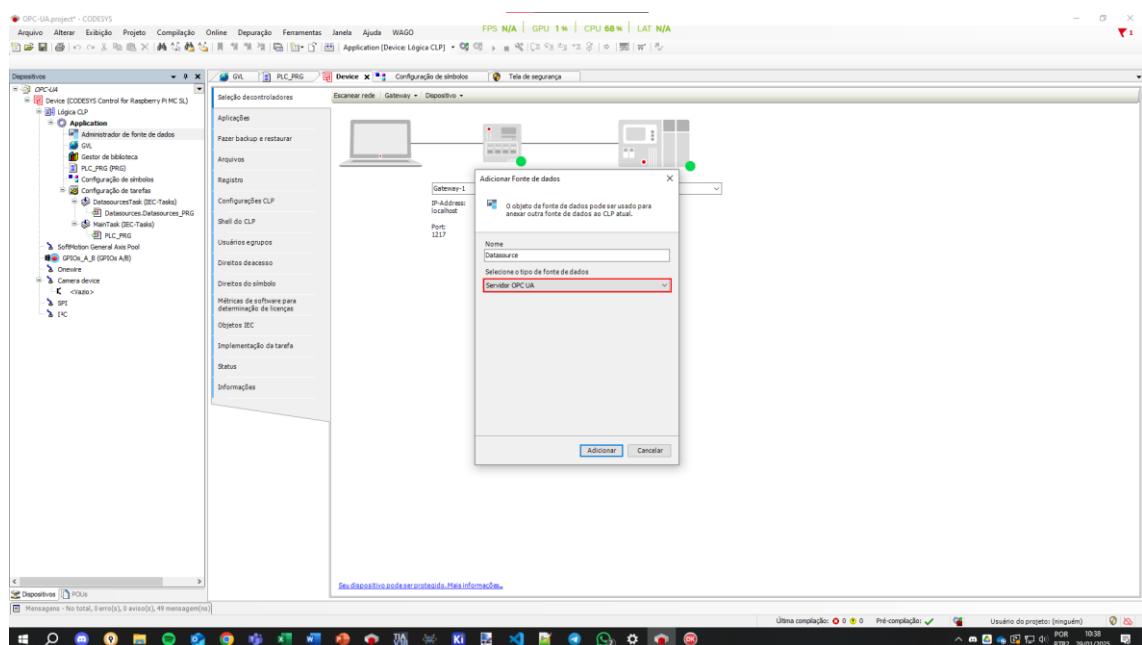
Para o cliente OPC UA, adiciona-se um novo objeto à **Application**, dessa vez, um **Administrador de fonte de dados...** como mostrado na figura abaixo.



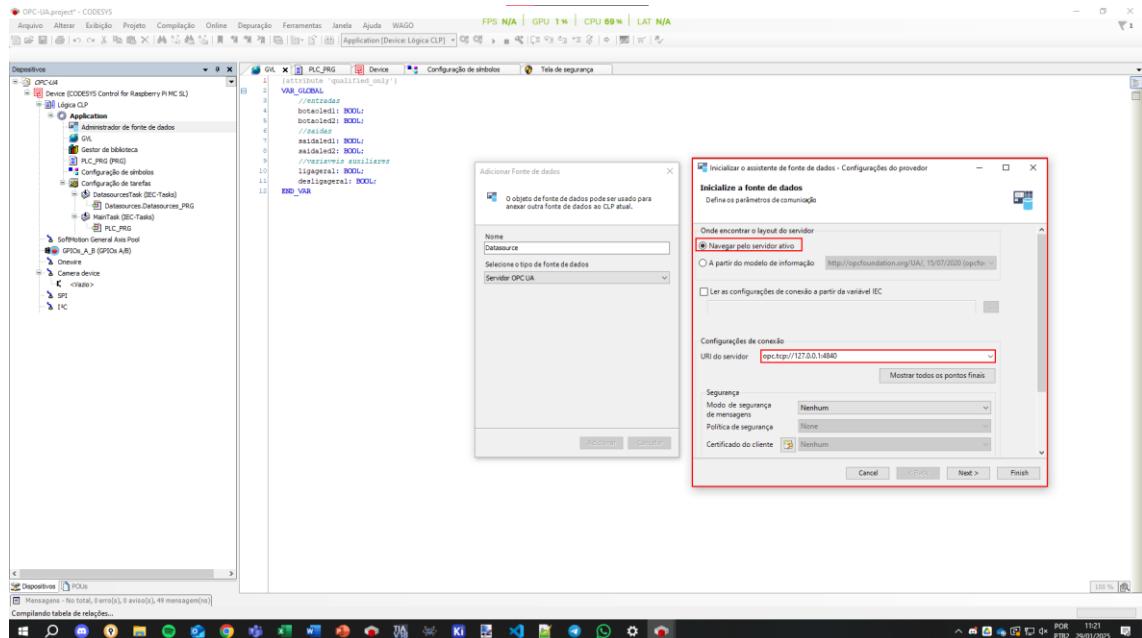
Agora em **Administrador de fonte de dados...**, adicione uma **Fonte de dados**, como mostrado na figura abaixo.



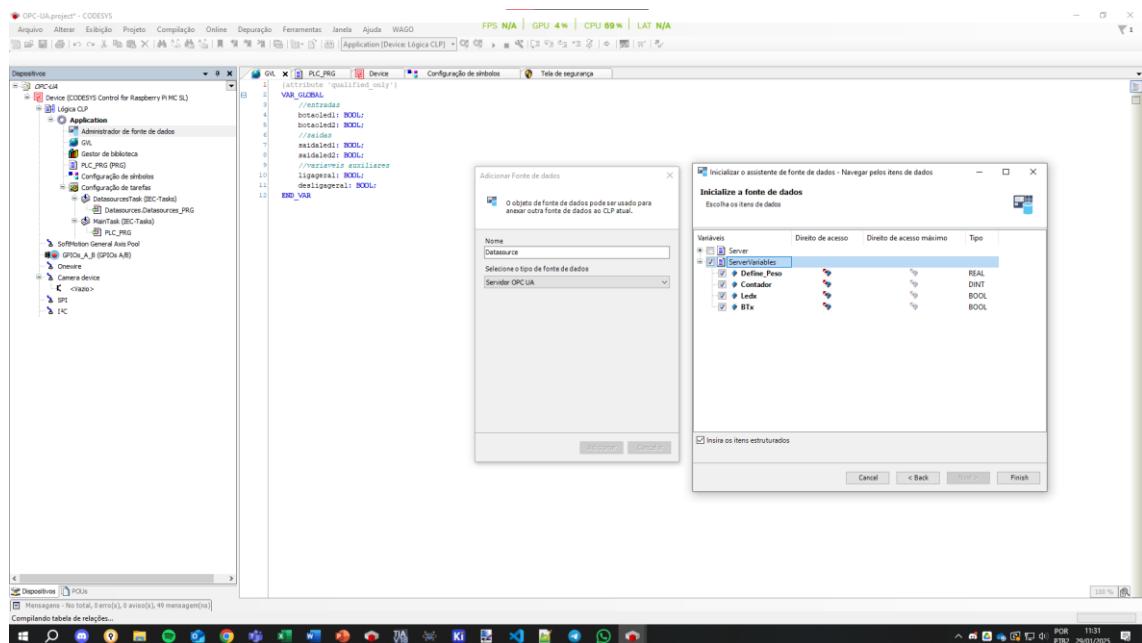
O tipo da **Fonte de dados**, deve ser **Servidor OPC UA**.



Após clicar em **adicionar**, abrirá essa nova janela, onde deve ser configurado o IP do servidor OPC UA, nesse caso, o servidor Python utilizado possui o seguinte endereço: **opc.tcp://127.0.0.1:4840**. Após configurar esse endereço, então clique em **NEXT**, esse procedimento irá fazer uma tentativa de conexão com o servidor, e se for bem-sucedida, irá aparecer os dados do servidor.



Na imagem abaixo, mostra o servidor e as variáveis após realizar a conexão, basta selecionar as variáveis do servidor e clicar em **Finish**.



Feito isso, o Codesys agora consegue ler e escrever em um servidor OPC UA.

9 - Apêndice

9.1 Dicas da Raspberry PI (RPI)

A nossa RPI é a versão Raspberry PI V3 B (RPI).

É necessário um cartão SD mini para rodar a RPI.

Toda as informações pode ser obtida diretamente do site : [Raspberry Pi](#)

9.2 Instalação do sistema operacional

Fazer o download no windows do Raspberry Pi imager ([Raspberry Pi OS – Raspberry Pi](#)).

Para a RPI 3 B deve ser instalado o OS de 32 bits.

Após gravar a imagem no cartão, retirar ele e colocar na raspberry.

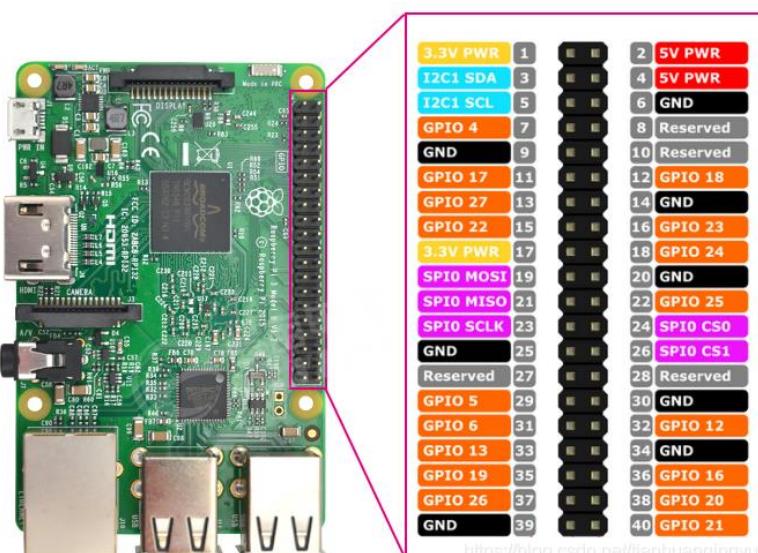
9.3 Configurando os GPIOs

Segundo os fóruns do Codesys, o Codesys suporta leitura e escrita nos GPIOs das seguintes placas:

- Raspberry PI 3 B
- Raspberry PI 3+
- Raspberry PI 4

Ainda não vi ninguém falando que conseguiu comunicar na Raspberry PI 5.

Como na UFU temos a Raspberry PI 3 B Rev 2, foi ela que eu testei e consegui fazer funcionar e vou abordar ela como RPI3B. A pinagem de GPIO da placa são estes da figura abaixo.



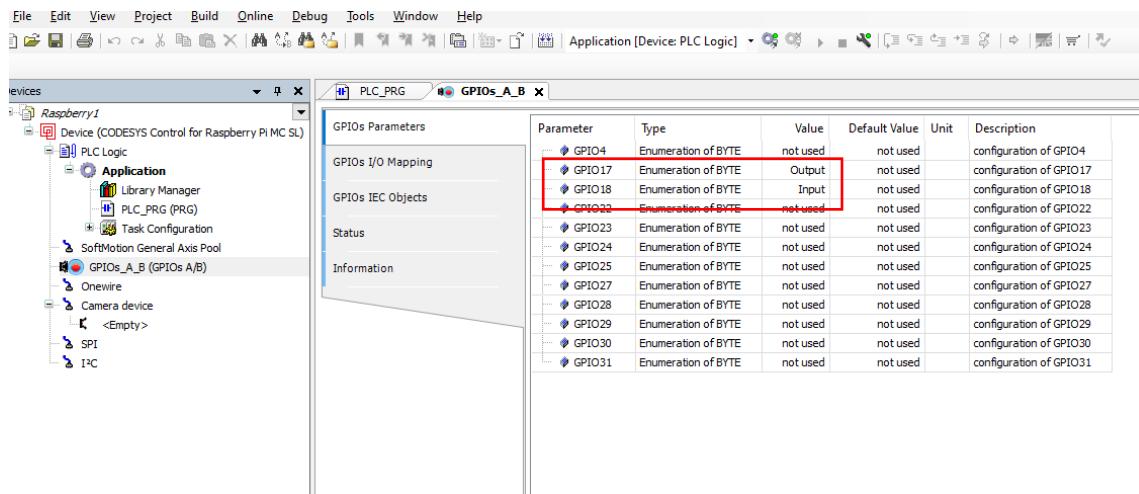
Outra pinagem incluindo o one-wire :https://pinout.xyz/pinout/1_wire

Para o Codesys, já está mapeado os seguintes GPIOs (somente 12 que podem ser escolhidos como entrada ou saída, porém isso vai depender da placa). A tabela abaixo mostra o que foi possível testar usando a RASP3BR2.

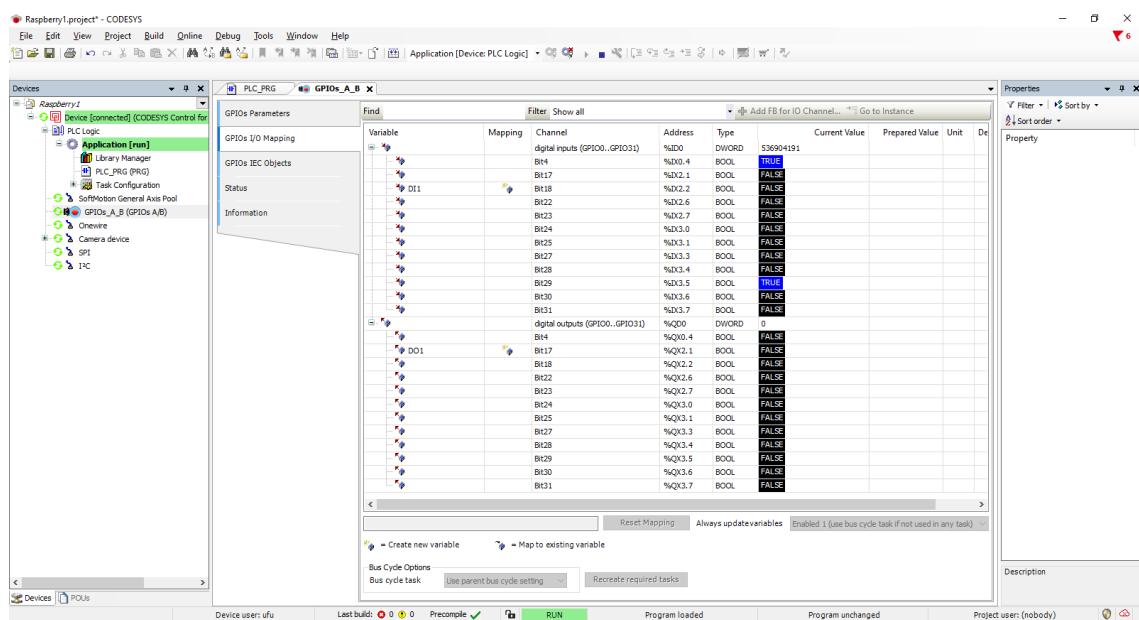
RASP3BR2.PIN	RASP3BR2.GPIO	Descrição
7	GPIO4	Ele está associado a um clock. Eu vi que precisa desabilitar ele primeiro. Não consegui fazer isso ainda.
11	GPIO17	OK
12	GPIO18	OK
13	GPIO27	OK

15	GPIO22	Ok
16	GPIO23	OK
18	GPIO24	OK
22	GPIO25	OK
	GPIO28	Estes GPIOs de 28 a 31 não existem na placa RASP3BR2
	GPIO29	
	GPIO30	
	GPIO31	

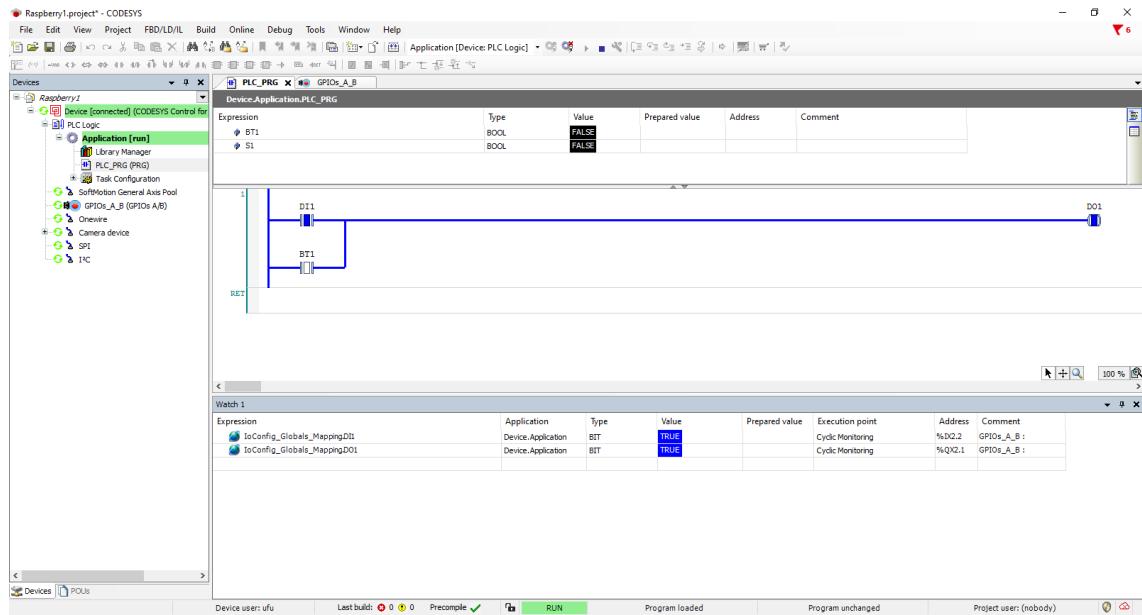
Após ligado fisicamente os IOs, agora deve ser feito o mapeamento na logica do GPIO como entrada ou saída. Então, para cada GPIO definir se ele eh input ou output conforme figura abaixo.



Após o mapeamento deve ser tageado os bits correspondentes em GPIO IO Mapping. Este tag é o que sera usado na lógica.



E então fazer a logica com os tags escolhidos



9.4 Resolvendo Problema : Caso a Raspberry não acione as saídas digitais

Dependendo da configuração do Linux da Raspberry, pode ser que o Codesys não consiga acessar corretamente os GPIOs. Neste caso é necessário desabilitar o modulo de Remote IO do GPIO. Para isso seguir os seguintes passos:

1. Acessar a raspberry através do Linux (com monitor, teclado e mouse, usuário e senha "ufu").
2. Abrir um terminal
3. Digitar “**>sudo raspi-config**” (neste caso vai abrir uma configuração de hardware como na bios do windows).
4. Ir em “Interface Options”
5. Ir em “Remote GPIO”
6. Colocar a opção “**No**” para “**Remote IO available**”
7. Após esta configuração é necessário resetar a raspberry. E neste caso agora será possível acessar os GPIOs como output no Codesys.

9.5 Usando SPI ou I2C no Codesys com a Raspberry

O CODESYS Control for Raspberry Pi MC SL tem suporte a SPI e I2C nativo a partir das versões mais recentes, mas com importantes limitações. Abaixo explico o que é possível fazer, como é feito, e como integrar com C quando necessário:

CODESYS oferece blocos de função para SPI (e também para I2C), mas apenas como parte da biblioteca GPIO (General Purpose I/O), que é limitada e não suporta protocolos complexos nativamente.

Configurar a SPI na Rasp PI

Para habilitar as interfaces **SPI** ou **I2C** na Raspberry Pi 3 Modelo B, siga os passos abaixo (válido para Raspbian / Raspberry Pi OS):

1. Acesse o menu de configuração

No terminal, execute:

```
bash  
CopiarEditar  
sudo raspi-config
```

2. Vá até o menu de interfaces

Navegue por:

```
plaintext  
CopiarEditar  
5 Interfacing Options → P4 SPI  
→ P5 I2C
```

Selecione **SPI**, pressione **Enter**, escolha **Yes** para habilitar.

Faça o mesmo para **I2C**.

3. Reinicie a Raspberry Pi

```
bash  
CopiarEditar  
sudo reboot
```

4. Verifique se foi habilitado

SPI: deve aparecer `/dev/spidev0.0` e `/dev/spidev0.1`

```
bash  
CopiarEditar  
ls /dev/spidev*
```

I2C: deve aparecer /dev/i2c-1

```
bash
CopiarEditar
ls /dev/i2c*
```

5. (Opcional) Instalar ferramentas para I2C

```
bash
CopiarEditar
sudo apt-get install -y i2c-tools
i2cdetect -y 1
```

Observação

A I2C padrão é a /dev/i2c-1 (bus principal).

A SPI padrão é a /dev/spidev0.0 (CS0) e /dev/spidev0.1 (CS1).

Ambas usam GPIOs padrão:

I2C: GPIO 2 (SDA), GPIO 3 (SCL)

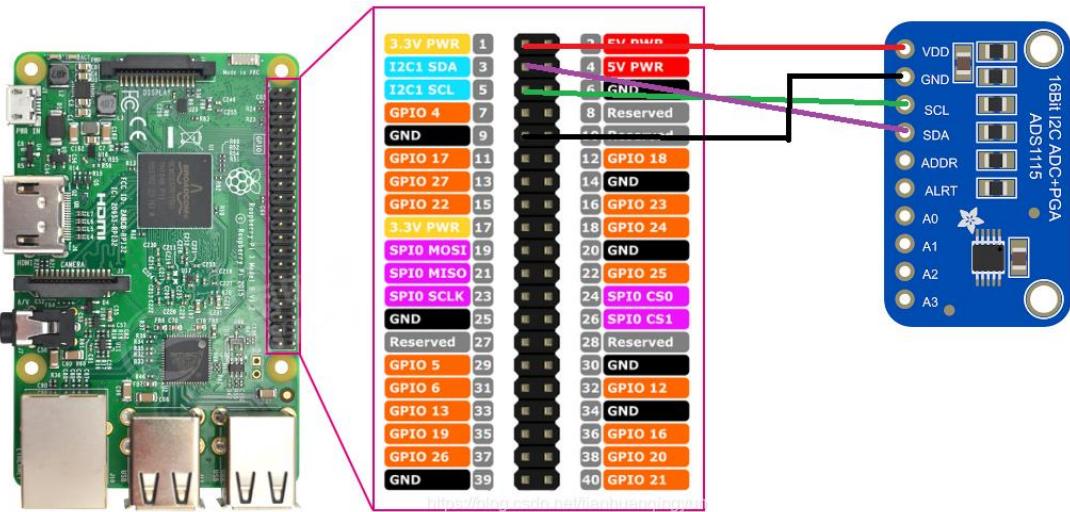
SPI: GPIO 10 (MOSI), 9 (MISO), 11 (SCLK), 8 (CE0), 7 (CE1)

Configurar no CODESYS

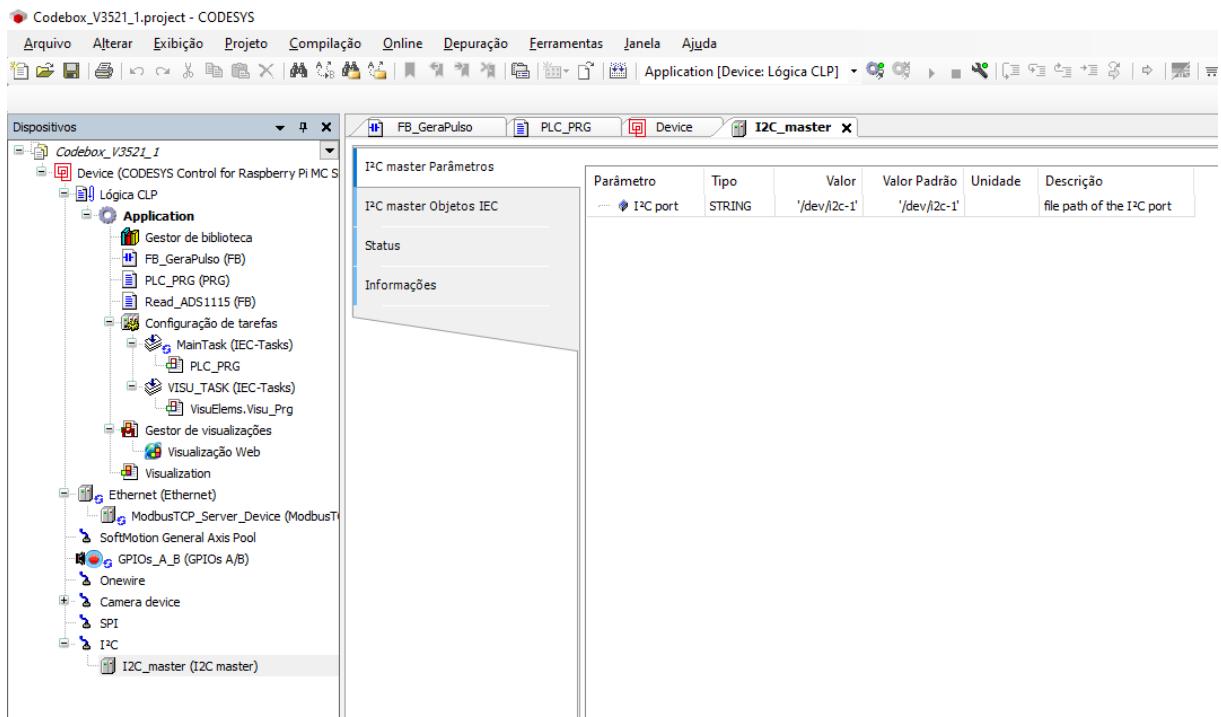
1. Instale e ative o pacote "CODESYS Control for Raspberry Pi MC SL".
2. Vá em Device > Add Device > Raspberry Pi SPI Master.
3. Acesse via blocos do tipo:
 - o SysGPIO.SPIRead()
 - o SysGPIO.SPIWrite()
 - o SysGPIO.SPIReadWrite()
4. Limitantes:
 5. Não há suporte direto a protocolos como SPI com framing, checksum, ou timing preciso.
 6. É necessário saber **exatamente o protocolo do dispositivo externo**, incluindo quantos bytes mandar, tempo entre bytes, etc.

Exemplo de Integração de Sensor ADC no Codesys

Este exemplo mostra a integração do sensor ADS1115 (comunicação I2C) com o Codesys. (modelo nosso da raspberry eh modelo 3 Rev B). A figura abaixo mostra a ligação do sensor com a raspberry (foi usado a porta I2C1 Pino 3 (I2C1.SDA) e Pino 5 (I2C1.SCL)).

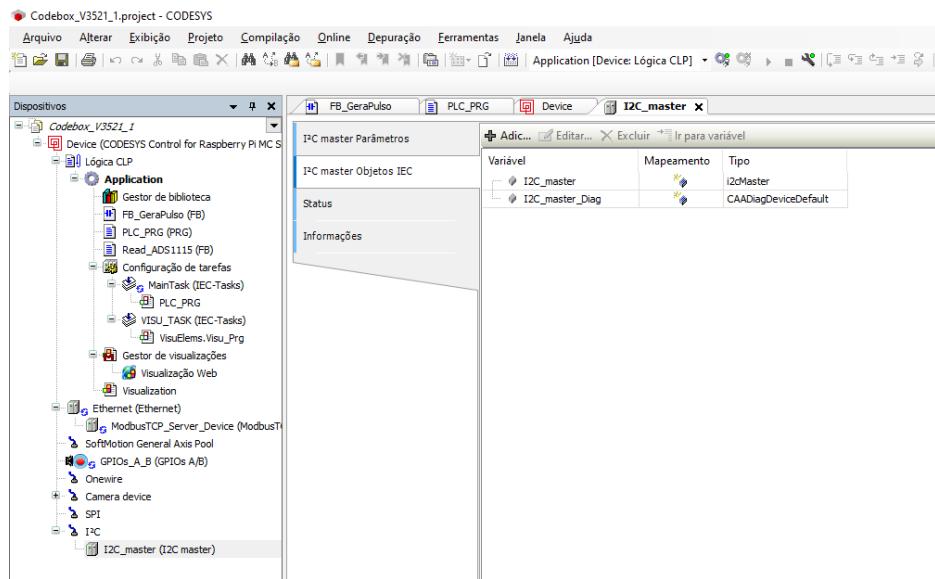


No codesys criar um projeto para raspberry PI MC SL



Criar um device I2C (clicar com o botão direito do mouse e adicionar um novo device). Por default ele já definiu a porta I2C-1 (""/dev/i2c-1").

A figura abaixo mostra que já foi criado um objeto para a instância da porta I2C chamado I2C_Master.



Então, basta usar no código (em ST) o objeto I2C_Master:

- I2C_Master.Operacional (Bool) – indica se o drive está habilitado (não testei para ver se ele consegue desabilitar se necessário)
- I2C_Master.read
- I2C_Master.write.

O ADS1115 é um ADC de 16 bits com registradores configuráveis. Como não foi configurado o pino de endereço, o Endereço I2C: 0x48 (padrão quando o pino ADDR está desconectado). Para ler a conversão de um canal (ex: AIN0), o fluxo padrão é:

1. Escreve-se no registrador de configuração (0x01) para iniciar a conversão. Aqui deve ser selecionado o ganho.
2. Aguarda-se (tipicamente 8 ms).
3. Lê-se 2 bytes do registrador de conversão (0x00).

Neste caso foi criado um function block em ST para ler o valor cujo código está abaixo. Aqui está mostrando o programa principal que está gerando um trem de pulso de 200ms e chama o function block e ele retorna o valor do ADC e o valor convertido em volts.

```

Dispositivos
Codebox_V3521_1
Device (CODESYS Control for Raspberry Pi MCS)
  Lógica CLP
    Application
      FB_GeraPulso (FB)
      PLC_PRG (PRG)
      Read_ADS1115 (FB)
        Configuração de tarefas
          MainTask (IEC-Tasks)
            PLC_PRG
            VISU_TASK (IEC-Tasks)
              VisuElems.Visu_Prg
        Gestor de visualizações
          Visualização Web
        Ethernet (Ethernet)
          ModbusTCP_Server_Device (ModbusT
            ModbusTCP_Server_Device (ModbusT
              ModbusTCP_Server_Device (ModbusT
                ModbusTCP_Server_Device (ModbusT
                  ModbusTCP_Server_Device (ModbusT
                    ModbusTCP_Server_Device (ModbusT
                      ModbusTCP_Server_Device (ModbusT
                        ModbusTCP_Server_Device (ModbusT
                          ModbusTCP_Server_Device (ModbusT
                            ModbusTCP_Server_Device (ModbusT
                              ModbusTCP_Server_Device (ModbusT
                                ModbusTCP_Server_Device (ModbusT
                                  ModbusTCP_Server_Device (ModbusT
                                    ModbusTCP_Server_Device (ModbusT
                                      ModbusTCP_Server_Device (ModbusT
                                        ModbusTCP_Server_Device (ModbusT
                                          ModbusTCP_Server_Device (ModbusT
                                            ModbusTCP_Server_Device (ModbusT
                                              ModbusTCP_Server_Device (ModbusT
                                                ModbusTCP_Server_Device (ModbusT
                                                  ModbusTCP_Server_Device (ModbusT
                                                    ModbusTCP_Server_Device (ModbusT
                                                      ModbusTCP_Server_Device (ModbusT
                                                        ModbusTCP_Server_Device (ModbusT
                                                          ModbusTCP_Server_Device (ModbusT
                                                            ModbusTCP_Server_Device (ModbusT
                                                              ModbusTCP_Server_Device (ModbusT
                                                                ModbusTCP_Server_Device (ModbusT
                                                                  ModbusTCP_Server_Device (ModbusT
                                                                    ModbusTCP_Server_Device (ModbusT
                                                                      ModbusTCP_Server_Device (ModbusT
                                                                        ModbusTCP_Server_Device (ModbusT
                                                                          ModbusTCP_Server_Device (ModbusT
                                                                            ModbusTCP_Server_Device (ModbusT
                                                                              ModbusTCP_Server_Device (ModbusT
                                                                                ModbusTCP_Server_Device (ModbusT
                                                                                  ModbusTCP_Server_Device (ModbusT
                                                                                    ModbusTCP_Server_Device (ModbusT
                                                                                      ModbusTCP_Server_Device (ModbusT
                                                                                      ModbusTCP_Server_Device (ModbusT
                                                                                      ModbusTCP_Server_Device (ModbusT
                                                                                      ModbusTCP_Server_Device (ModbusT
                                                                                      ModbusTCP_Server_Device (ModbusT
                                                                                      ModbusTCP_Server_Device (ModbusT
                                                                                      ModbusTCP_Server_Device (ModbusT
                                                                                      ModbusTCP_Server_Device (ModbusT
                                                                                      ModbusTCP_Server_Device (ModbusT
                                                                                      ModbusTCP_Server_Device (ModbusT
                                                                                      ModbusTCP_Server_Device (ModbusT
                                                                                      ModbusTCP_Server_Device (ModbusT
                                                                                      ModbusTCP_Server_Device (ModbusT
                                                                                      ModbusTCP_Server_Device (ModbusT
                                                                                      ModbusTCP_Server_Device (ModbusT
                                                                                      ModbusTCP_Server_Device (ModbusT
                                                                                      ModbusTCP_Server_Device (ModbusT
                                                                                      ModbusTCP_Server_Device (ModbusT
                                                                                      ModbusTCP_Server_Device (ModbusT
                                                                                      ModbusTCP_Server_Device (ModbusT
                                                                                      ModbusTCP_Server_Device (ModbusT
                                                                                      ModbusTCP_Server_Device (ModbusT
                                                                                      ModbusTCP_Server_Device (ModbusT
                                                                                      ModbusTCP_Server_Device (ModbusT
                                                                                      ModbusTCP_Server_Device (ModbusT
                                                                                      ModbusTCP_Server_Device (ModbusT
                                                                                      ModbusTCP_Server_Device (ModbusT
                                                                                      ModbusTCP_Server_Device (ModbusT
                                                                                      ModbusTCP_Server_Device (ModbusT
                                                                                      ModbusTCP_Server_Device (ModbusT
                                                                                      ModbusTCP_Server_Device (ModbusT
                                                                                      ModbusTCP_Server_Device (ModbusT
                                                                                      ModbusTCP_Server_Device (ModbusT
                                                                                      ModbusTCP_Server_Device (ModbusT
                                                                                      ModbusTCP_Server_Device (ModbusT
                                                                                      ModbusTCP_Server_Device (ModbusT
                                                                                      ModbusTCP_Server_Device (ModbusT
                                                                                      ModbusTCP_Server_Device (ModbusT
                                                                                      ModbusTCP_Server_Device (ModbusT
                                                                                      ModbusTCP_Server_Device (ModbusT
                                                                                      ModbusTCP_Server_Device (ModbusT
                                                                                      ModbusTCP_Server_Device (ModbusT
                                                                                      ModbusTCP_Server_Device (ModbusT
                                                                                      ModbusTCP_Server_Device (ModbusT
                                                                                      ModbusTCP_Server_Device (ModbusT
                                                                                      ModbusTCP_Server_Device (ModbusT
                                                                                      ModbusTCP_Server_Device (ModbusT
                                                                                      ModbusTCP_Server_Device (ModbusT
                                                                                      ModbusTCP_Server_Device (ModbusT
                                                                                      ModbusTCP_Server_Device (ModbusT
                                                                                      ModbusTCP_Server_Device (ModbusT
                                                                                      ModbusTCP_Server_Device (ModbusT
                                                                                      ModbusTCP_Server_Device (ModbusT
                                                                                      ModbusTCP_Server_Device (ModbusT
                                                                                      ModbusTCP_Server_Device (ModbusT
                                                                                      ModbusTCP_Server_Device (ModbusT
                                                                                      ModbusTCP_Server_Device (ModbusT
                                                                                      ModbusTCP_Server_Device (ModbusT
                                                                                      ModbusTCP_Server_Device (ModbusT
                                                                                      ModbusTCP_Server_Device (ModbusT
                                                                                      ModbusTCP_Server_Device (ModbusT
                                                                                      ModbusTCP_Server_Device (ModbusT
                                                                                      ModbusTCP_Server_Device (ModbusT
................................................................

```

Function Block to Read ADS1115

Sessão de Variáveis

FUNCTION_BLOCK Read_ADS1115

VAR_INPUT

PULSO_MS : BOOL;

FIRST_SCAN : BOOL;

END_VAR

VAR_OUTPUT

OUT_RAW : UINT;

OUT_V : REAL;

END_VAR

VAR

adcaddr : BYTE := 16#48;

writeBuf : ARRAY [0..2] OF BYTE; // Endereço do registrador + config

readBuf : ARRAY [0..1] OF BYTE; // Resultado da conversão

status : BOOL;

adcValue : UINT;

RT1 : R_TRIGGER;

rValVolts : REAL;

```
END_VAR
```

```
VAR CONSTANT
```

```
    ADC_LIMIT : UINT := 26525;
```

```
    V_LIMIT : REAL := 3.3;
```

```
END_VAR
```

Código Fonte

```
(* leitura do ADC via I2C *)
```

```
IF (TRUE) THEN
```

```
    // 1. Configura ADS1115 para AIN0, modo single-shot, ganho 2.048V, 128SPS
```

```
    // Registrador de configuração = 0x01
```

```
    // Dados: MSB = 0xC3, LSB = 0x83 → Config de canal, ganho, modo, etc.
```

```
    writeBuf[0] := 16#01; // Endereço do registrador de configuração
```

```
    writeBuf[1] := 16#C3; // MSB da config
```

```
    writeBuf[2] := 16#83; // LSB da config
```

```
    // Envia a configuração
```

```
    I2C_Master.Write(adcaddr, ADR(writeBuf), 3);
```

```
END_IF
```

```
// Aguarda conversão (8ms típico)
```

```
RT1(CLK:=PULSO_MS);
```

```
IF (RT1.Q) THEN
```

```
    // 2. Aponta para o registrador de conversão (0x00)
```

```
    writeBuf[0] := 16#00;
```

```
    I2C_Master.Write(adcaddr, ADR(writeBuf), 1);
```

```
    // 3. Lê 2 bytes do registrador de conversão
```

```
    I2C_Master.Read(adcaddr, ADR(readBuf), 2);
```

```
END_IF
```

```
// 4. Monta valor lido (16 bits, MSB primeiro)
```

```
adcValue := readBuf[0];
```

```
adcValue := SHL(adcValue, 8) OR readBuf[1]; // conversão direta
```

```
(* convert raw value in float *)
```

```
rValVolts := INT_TO_REAL(UINT_TO_INT(adcValue));
```

```
rValVolts := (rValVolts /ADC_LIMIT) * V_LIMIT;  
(* update the outputs *)  
OUT_RAW := adcValue;  
OUT_V := rValVolts;
```

Integrar Código C com CODESYS na Raspberry Pi

⭐ Sim, é possível integrar código C usando os seguintes métodos:

Método 1 – Comunicação via TCP/UDP sockets

- Escreva um programa em C que roda como **daemon ou serviço** na Raspberry Pi.
- Ele se comunica com o dispositivo via SPI (usando spidev).
- O CODESYS se comunica com esse programa via **TCP socket** usando blocos de função como:
 - SysSocket.SysSockClient
 - SysSocket.SysSockServer
- Comunicação em protocolo binário ou texto.

Método 2 – Arquivos ou memória compartilhada

- O programa em C salva dados em um arquivo ou FIFO (/tmp/spi_data.bin).
- O CODESYS pode ler/escrever esse arquivo usando funções do tipo:
 - SysFile.SysFileOpen, SysFile.SysFileRead

Método 3 – External Library Interface (avançado)

- Compila a função C como **shared library (.so)**.
- Cria uma interface em CODESYS com as funções usando header file (.h).
- Permite chamar diretamente funções em C via CODESYS.

⚠ Este método exige conhecimento em **compilação cruzada**, cuidado com **tipos de dados compatíveis**, e a criação de um **Device Description File (.xml)** se for criar um dispositivo CODESYS customizado.

Fonte:

- CODESYS External C Library documentation
- Exemplo de integração: <https://forge.codesys.com/>

⚠ Conclusão Técnica

Integração	É possível?	Como fazer?	Comentário
SPI via CODESYS	Parcial	Usando blocos da GPIO library (SysGPIO.SPIReadWrite)	Limitado; só funciona para comandos simples.
SPI via programa C	Sim	spidev em C ou Python + comunicação com CODESYS via socket, FIFO ou arquivo	Flexível e confiável.
Integração com C	Sim	shared object (.so) + External Library Interface	Complexo, mas possível.

Integrating C Modules

https://content.helpme-codesys.com/en/CODESYS%20Development%20System/_cds_integrating_c_code.html

Com a integração do código C do CODESYS, módulos desenvolvidos em C podem ser integrados como uma biblioteca em um projeto CODESYS. O módulo é armazenado como um módulo de tempo de execução dinâmico em uma biblioteca, juntamente com a interface IEC implementada nela. Além disso, diversas variantes de um módulo para diferentes sistemas de destino podem ser armazenadas em uma biblioteca.

A biblioteca é exibida ao usuário como qualquer outra biblioteca IEC 61131-3. Quando a biblioteca é usada em uma aplicação, o CODESYS baixa automaticamente o módulo adequado da biblioteca para o controlador e o executa a partir do sistema de tempo de execução CODESYS Control.

Exemplos de aplicação:

Reutilização de código C existente

Integração de código gerado por ferramentas de modelagem, como Matlab®/Simulink®

Integração dinâmica de funções específicas do dispositivo criadas em código C

Todos os módulos dinâmicos de uma aplicação são transferidos e carregados para o sistema de tempo de execução durante o download. O sistema de tempo de execução deve suportar vinculação dinâmica para isso.

Licença para o sistema de tempo de execução

O sistema de tempo de execução requer uma licença que permite o carregamento de módulos C. Sem essa licença, os módulos dinâmicos não podem ser vinculados durante o download e, portanto, o download será abortado.

Os módulos dinâmicos fazem parte do aplicativo de inicialização e são recarregados e ativados quando o controlador é reiniciado. O comando Reset Origin descarrega todos os módulos de código C do aplicativo. Os comandos Reset Cold e Reset Warm não resultam em uma inicialização repetida dos módulos de código C.

O CODESYS não suporta o monitoramento de variáveis em arquivos de código C ou a definição de pontos de interrupção em código-fonte C.

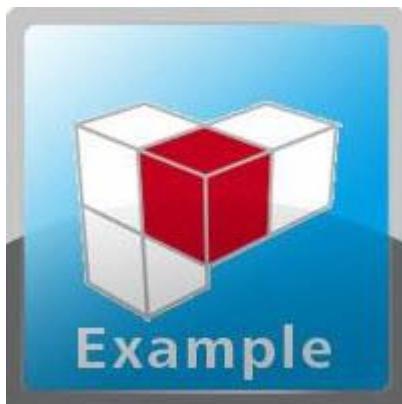
Example: Using a C-Implemented Function

Product: CODESYS Control Extension Package

Example: Using a C-Implemented Function

Product: CODESYS Control Extension Package

The CIntegrationExample.project sample project shows how to use a C-code implemented function in an application.



After the installation of CODESYS Control Extension Package, the sample project is located in the CODESYS Development System installation directory, in the CODESYS Control SL Extension Package\<version>\Examples subdirectory.

System requirements and restrictions

Programming system	CODESYS Development System (version 3.5.18.0 or higher)
Runtime system	CODESYS Control Extension Package (version 4.7.0.0 or higher)

