Comunicação entre VTSCADA e CodeSys & Controle de nível de tanque com Factory IO

Henrique Silva Coutinho

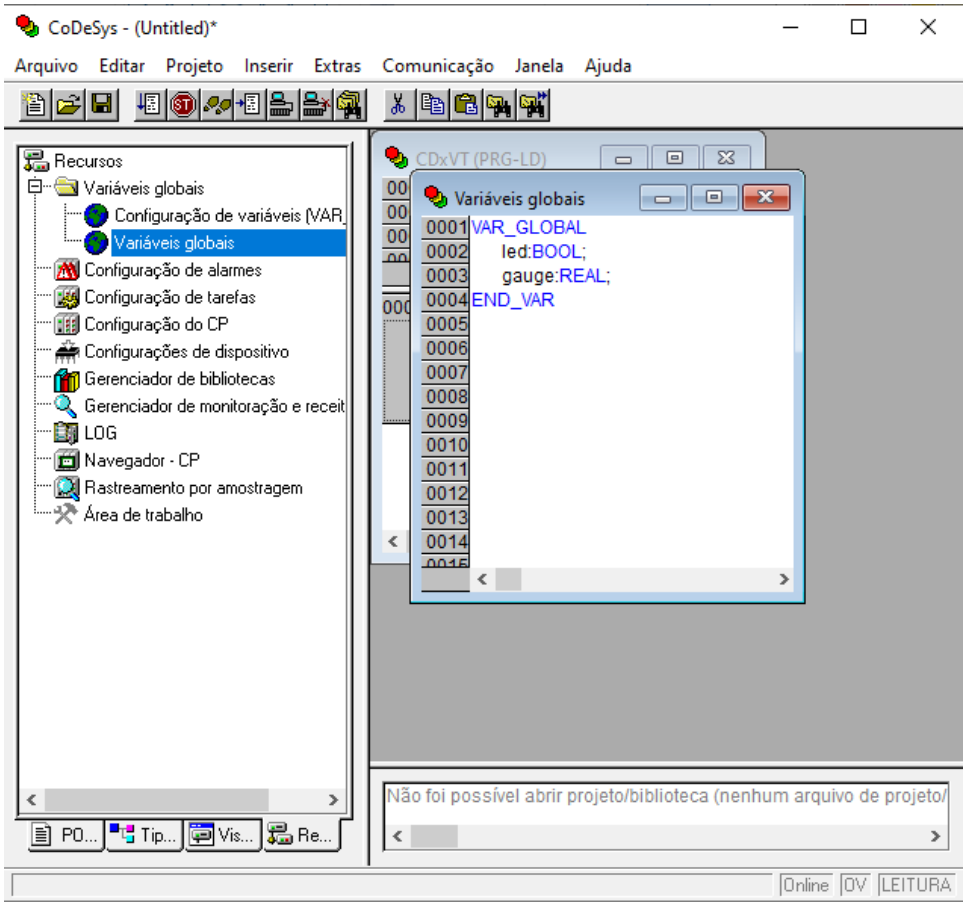
Comunicação entre o VTSCADA E O CODESYS

Neste tutorial aprenderemos:

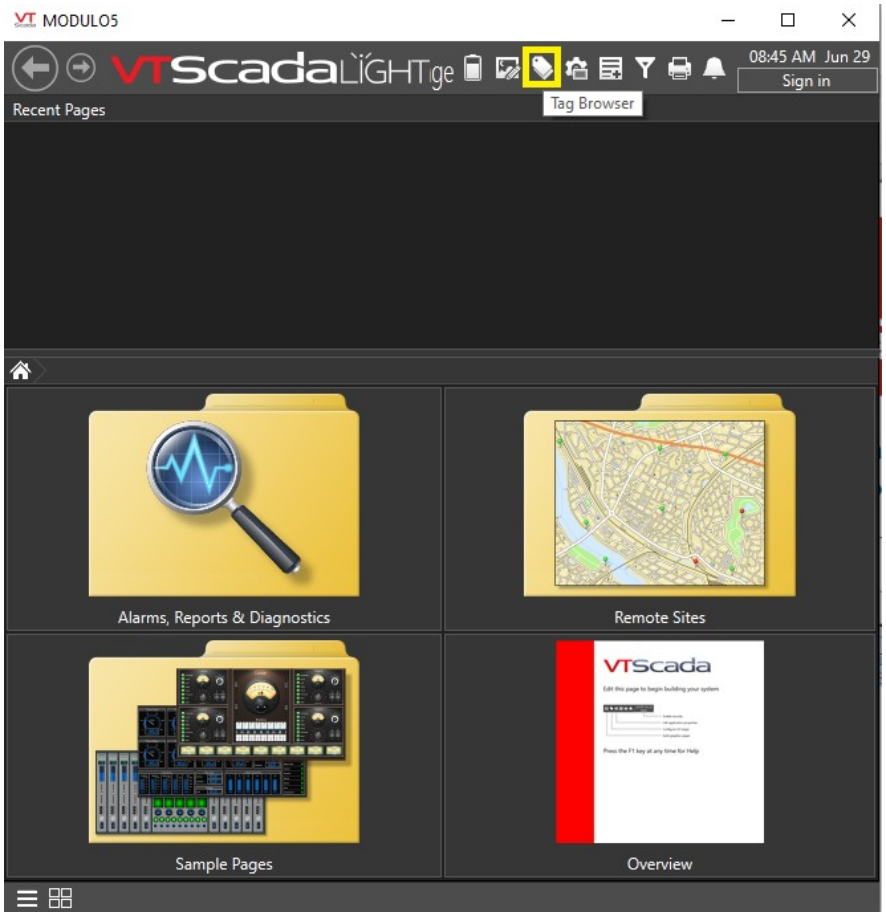
- Como realizar a comunicação entre o VTSCADA e o CODESYS através de uma troca bidirecional;
- Fazer um programa em LADDER no CODESYS comunicar com o supervisório no VTSCADA;
- Implementar uma função de primeira ordem no CODESYS e comunicar com o VTSCADA.

Criando aplicação no CodeSys e no VTSCADA

Com o projeto criado no CodeSys adicionamos duas variáveis globais, para o led e para o gauge.

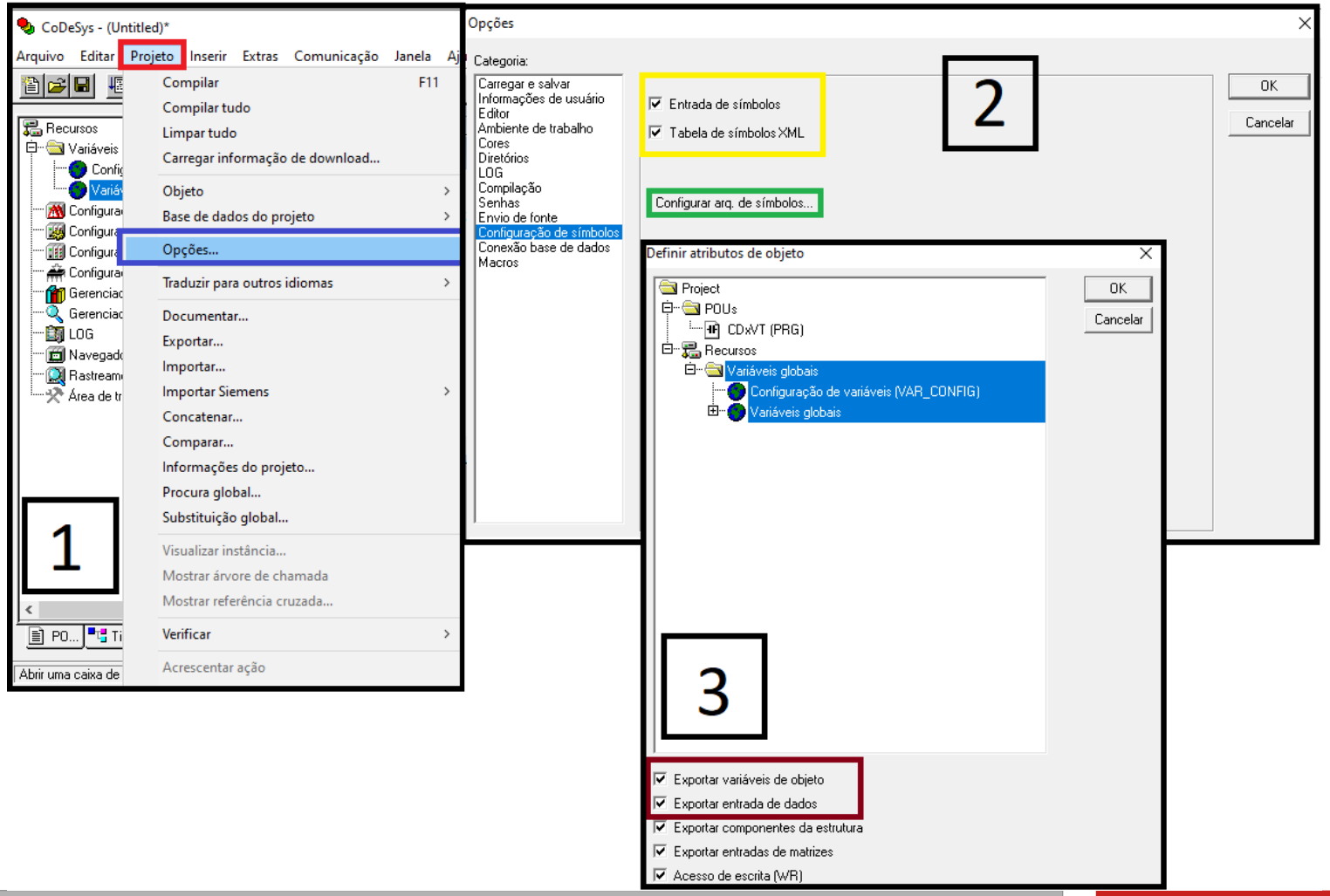


No VTSCADA, com a aplicação criada vamos adicionar uma tag clicando no Tag Browser:



Criando aplicação no CodeSys e no VTSCADA

Devemos então estabelecer quais programações e variáveis vamos exportar para o VTSCADA:

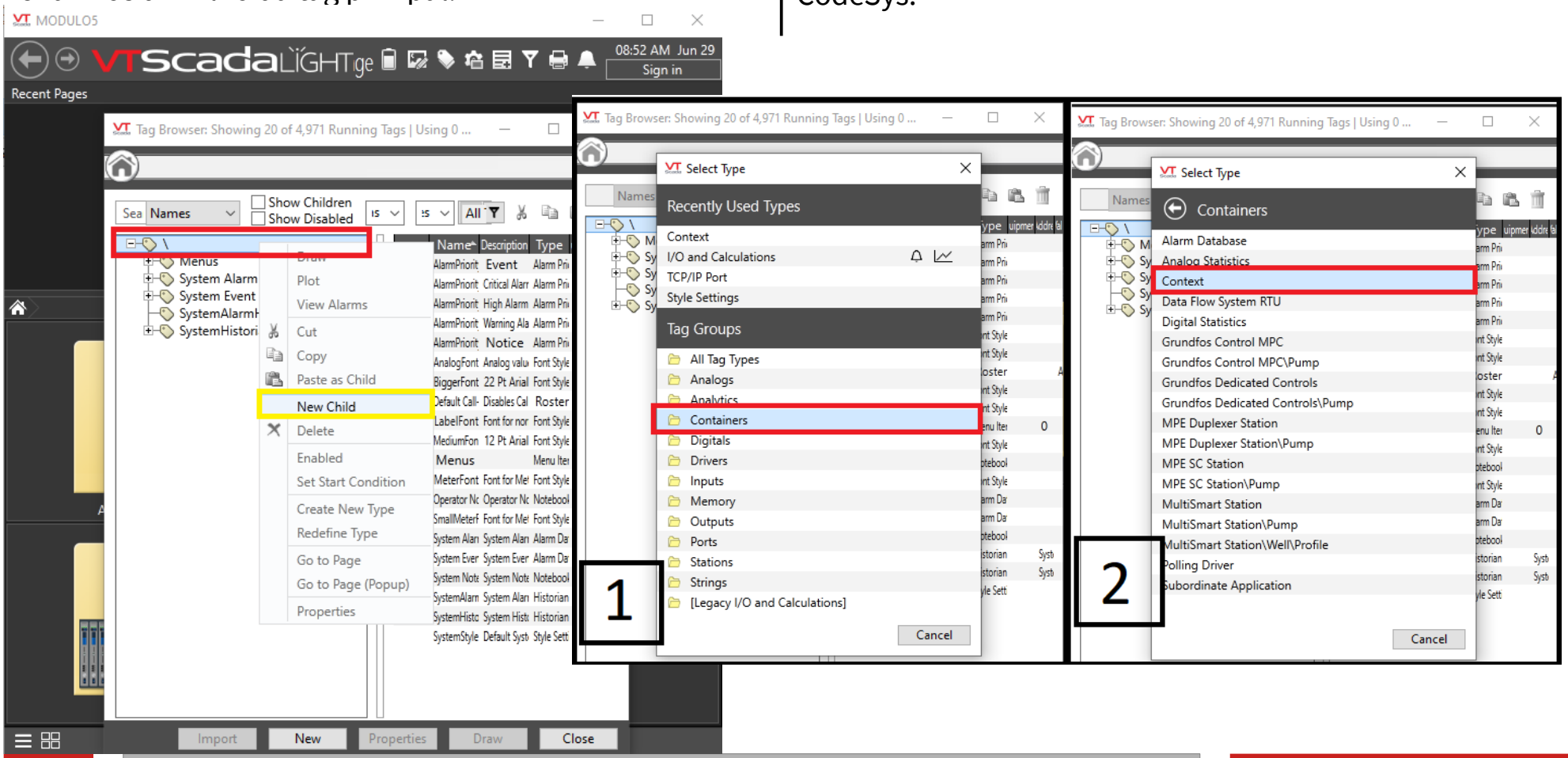


Configuração do CodeSys para exportar. Seguindo as ordens de tela 1,2 e 3 e marcando as opções em colorido.

Criando aplicação no CodeSys e no VTSCADA

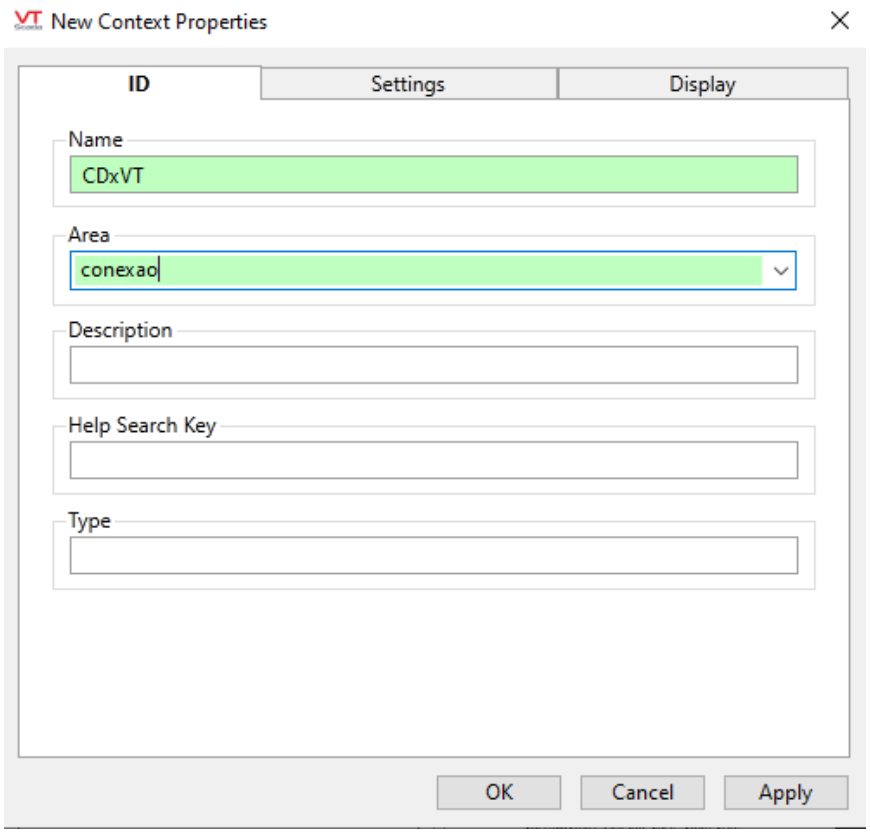
No Tag Browser, clicamos com o botão direito do mouse na área em vermelho e clicamos em New Child para criarmos um filho da tag prinipal:

Clicamos então nos botões em vermelho (Containers e em seguida Context) para criarmos a conexão com o CodeSys:

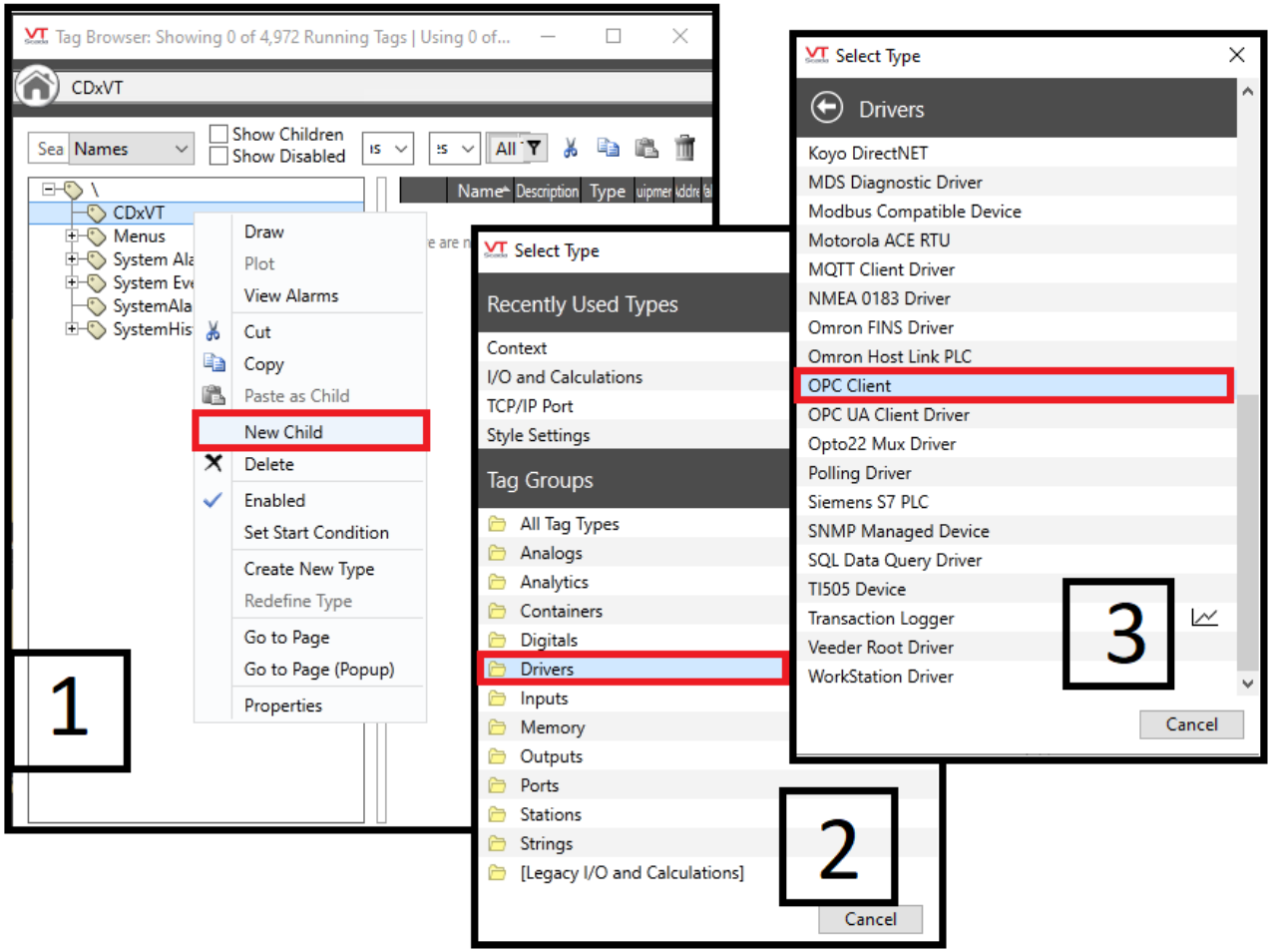


Criando aplicação no CodeSys e no VTSCADA

Damos um nome e escrevemos a área:



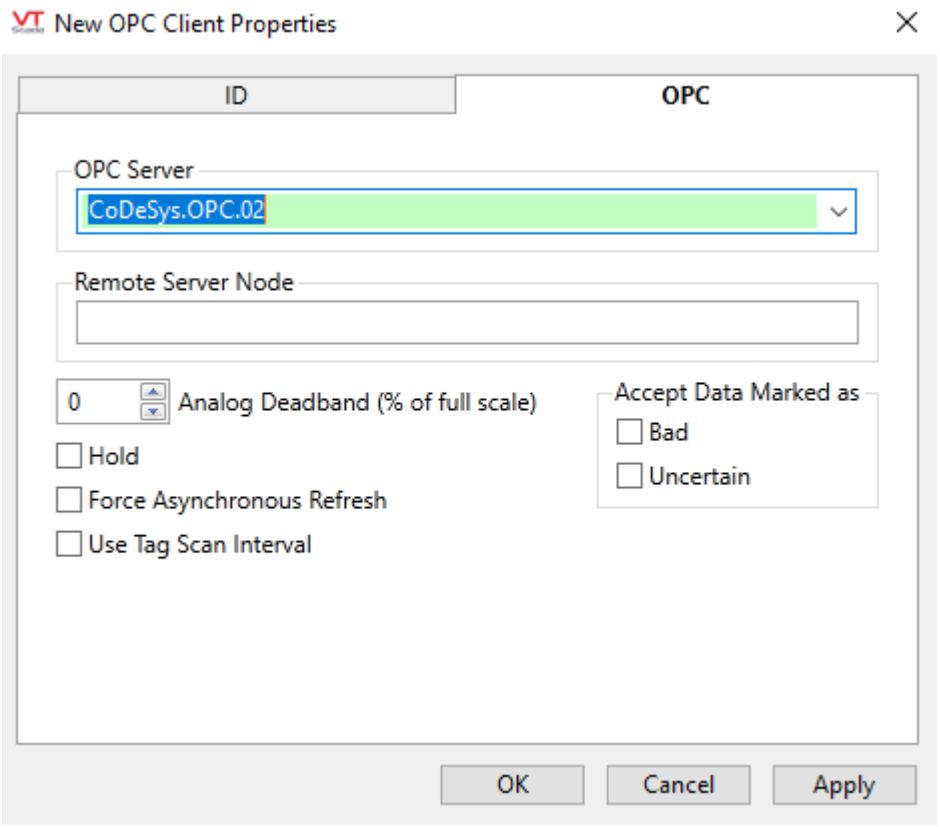
Criando aplicação no CodeSys e no VTSCADA



Novamente seguimos a ordem enumerada para conectarmos ao OPC do CodeSys

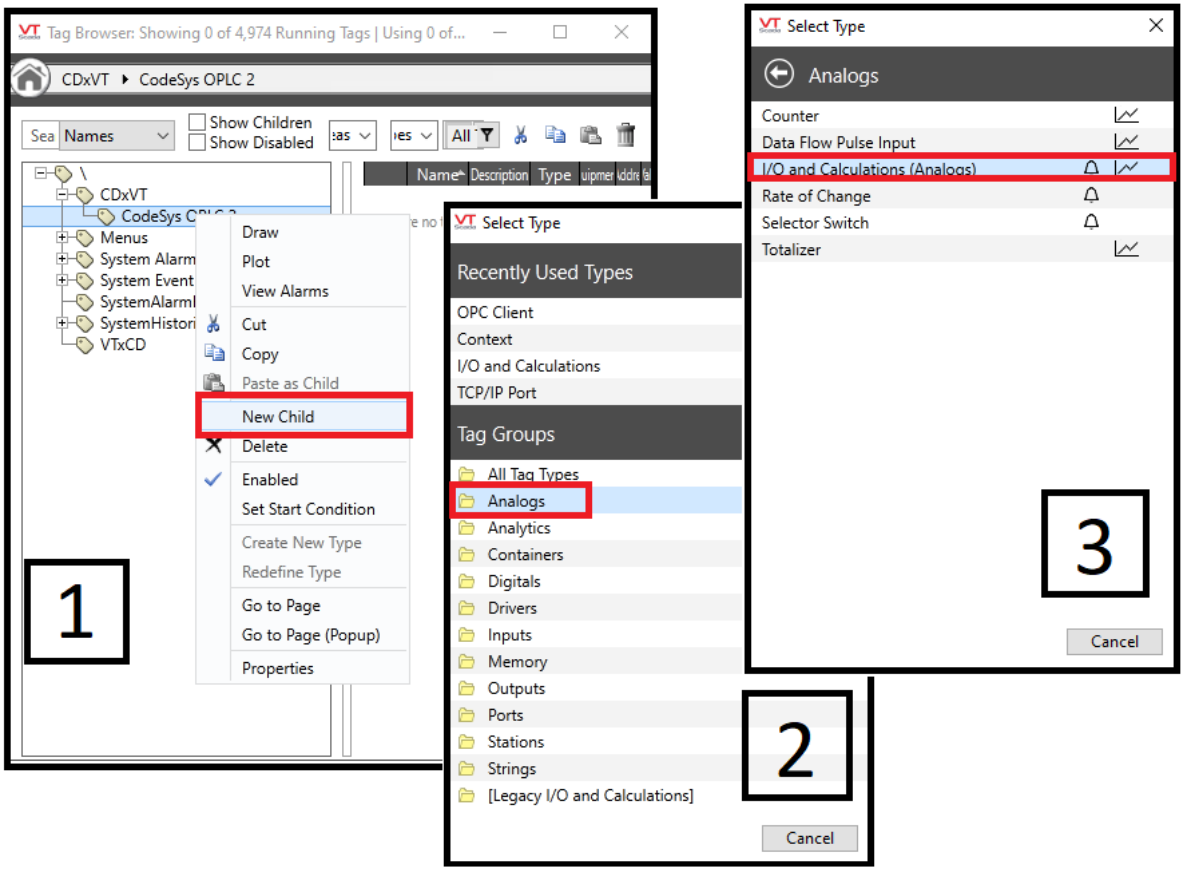
Criando aplicação no CodeSys e no VTSCADA

Selecione o OPC Server do CodeSys como abaixo:



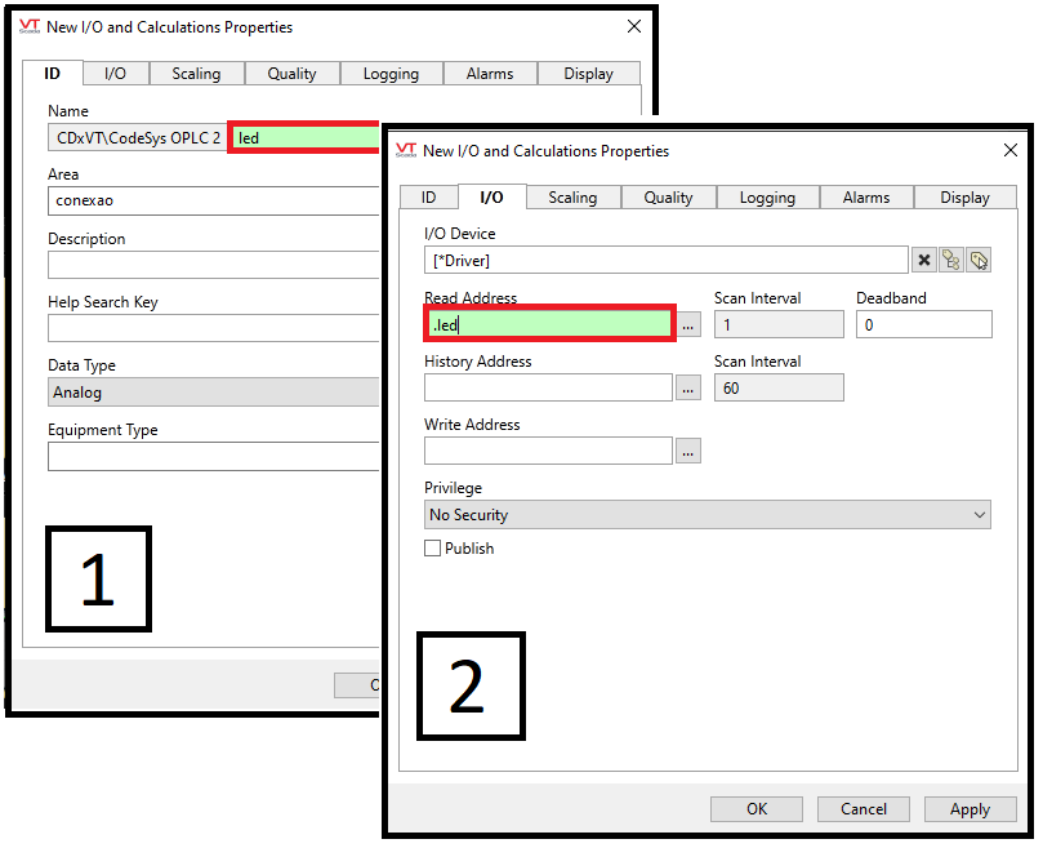
Criando aplicação no CodeSys e no VTSCADA

Agora devemos criar as tags para nossas variáveis: Para ambas varáveis seguimos os passos a seguir:



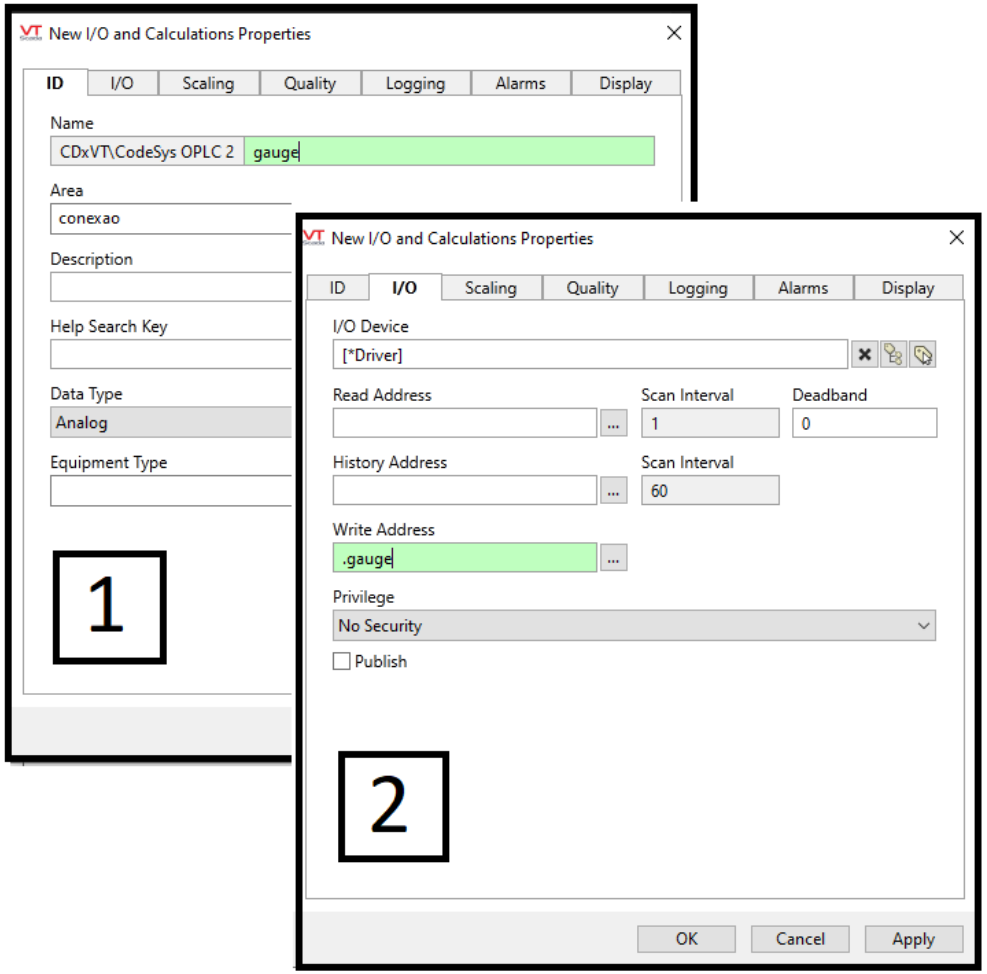
Criando aplicação no CodeSys e no VTSCADA

Para o led:



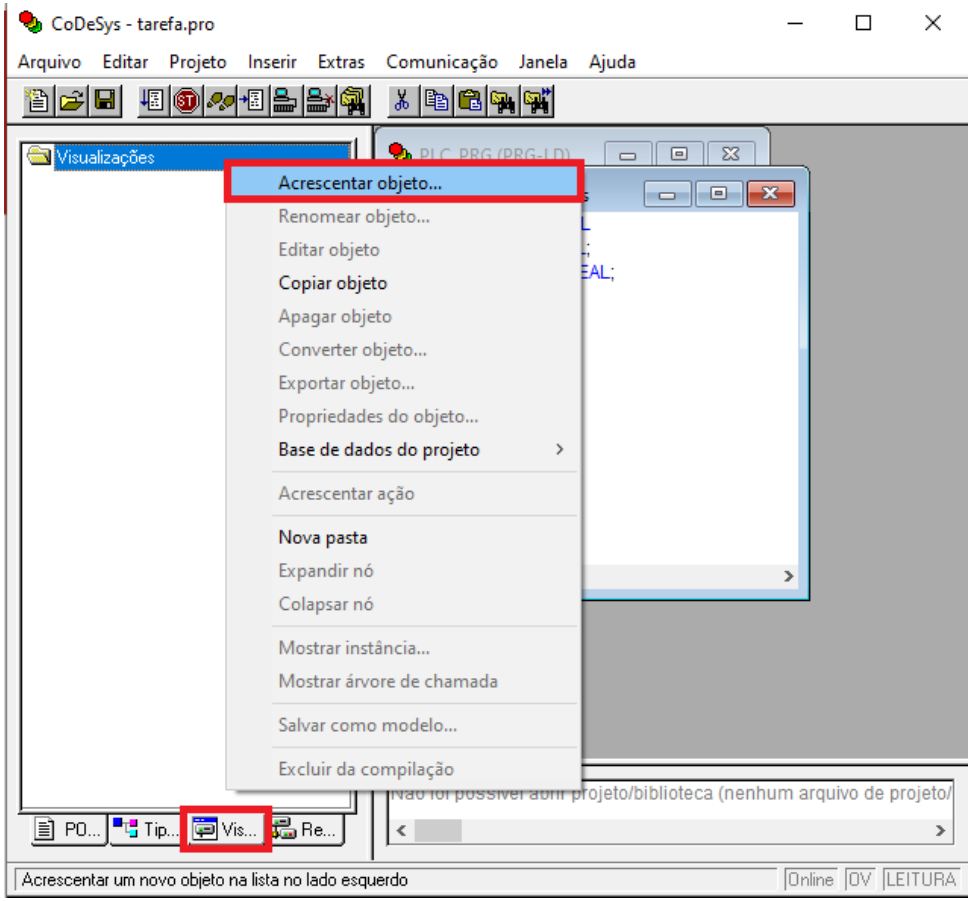
Criando aplicação no CodeSys e no VTSCADA

Para o gauge



Configurando a IHM

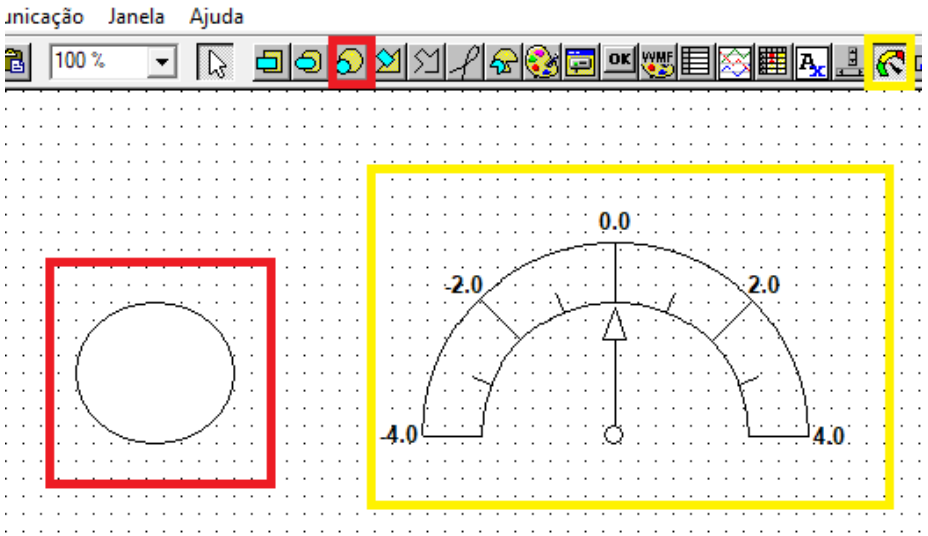
No CodeSys criamos uma nova visualização:



No VTSCADA vamos ao Idle Studio:

Configurando a IHM

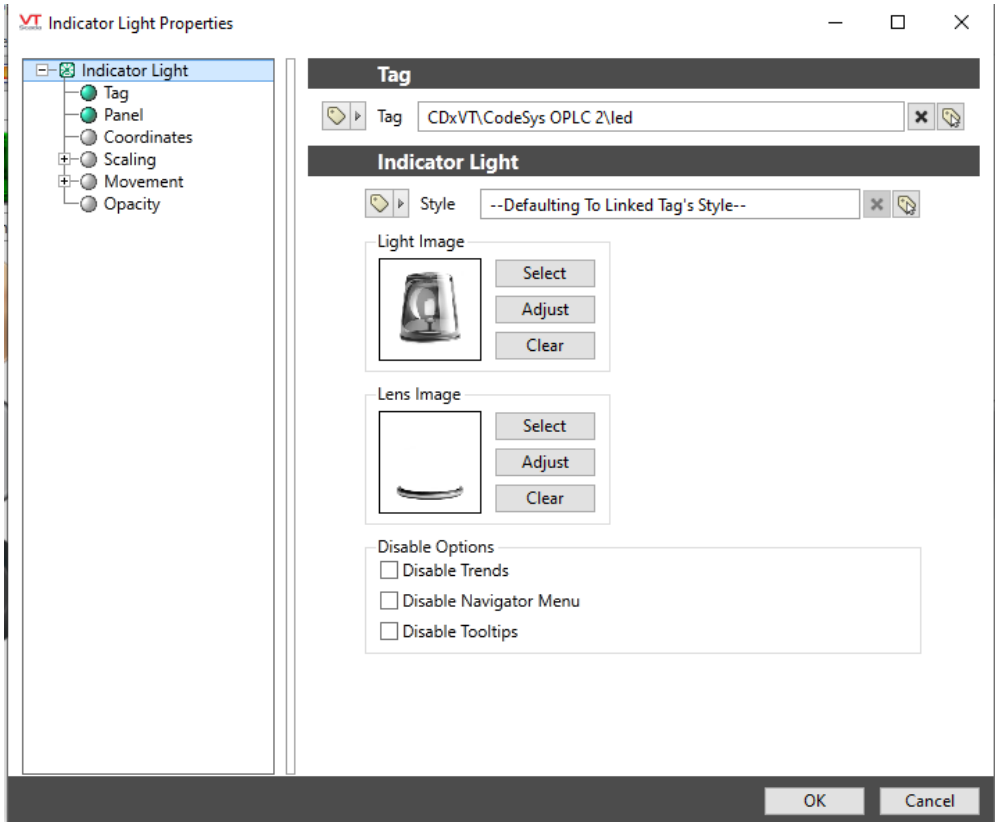
Na visualização colocamos algo para representar um botão(nesse exemplo um circulo) e um gauge :

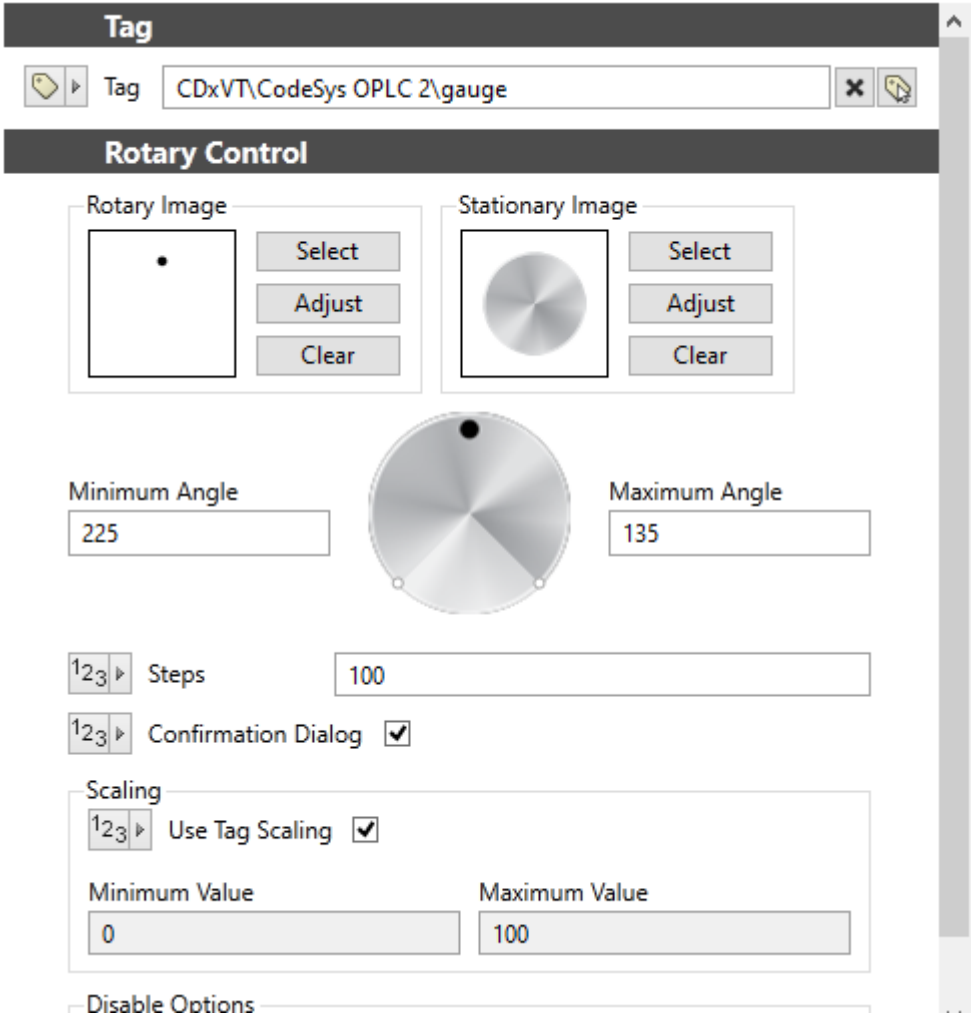


No Idle Studio adicionamos um led e um slide:

Configurando a IHM

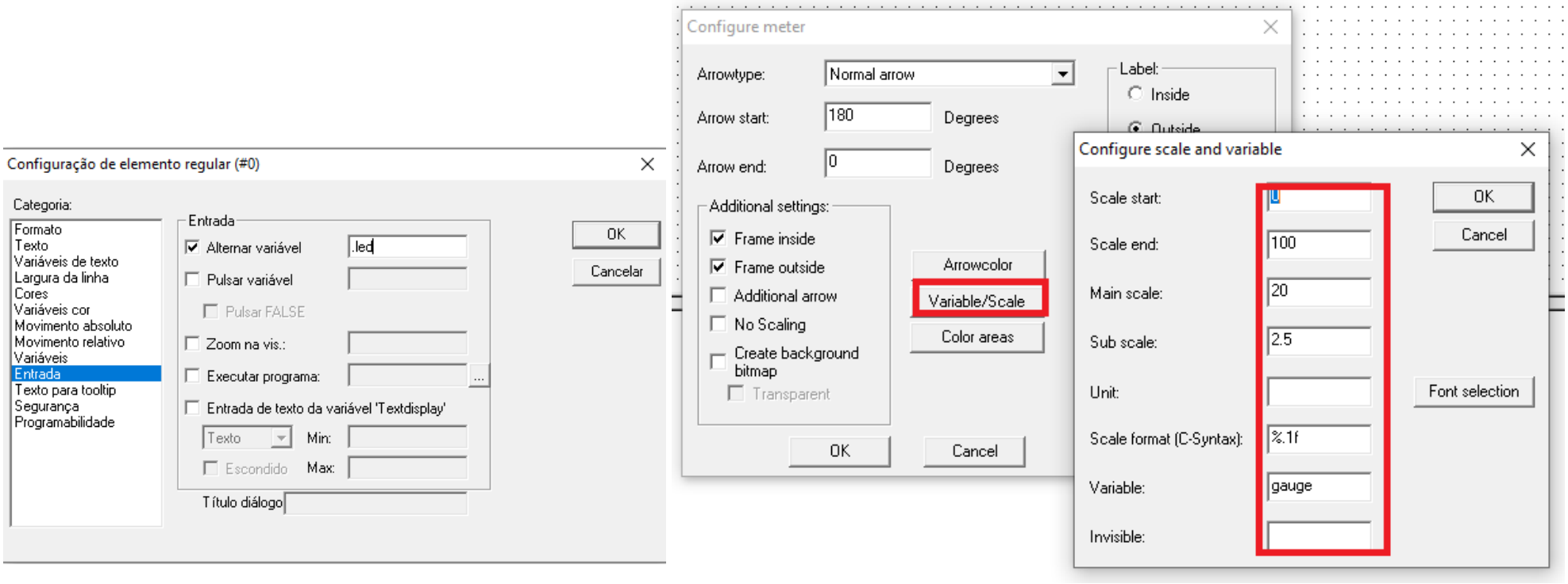
Configurando o led e o slide:





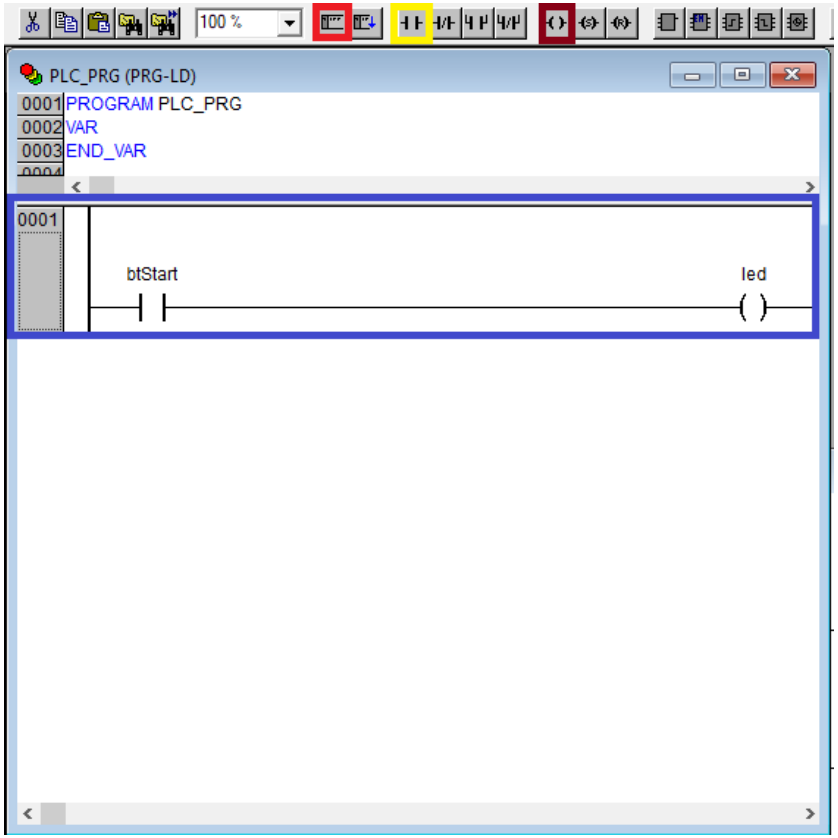
Configurando a IHM

Configurando o botão e o gauge:



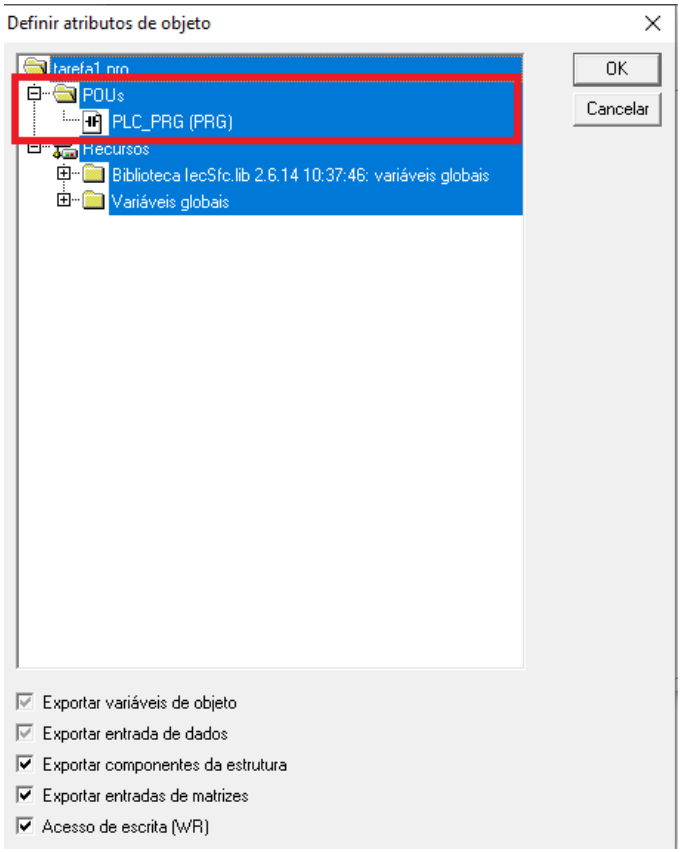
Comunicação programa LADDER

Para realizar a comunicação entre um programa LADDER no CodeSys com o VTSCADA é simples. Primeiramente façamos um porgrama LADDER como no primeiro tutorial. (foi adicionado um variável btStart ao nosso programa anterior, desta forma temos que proceder de forma igual para fazer a ligação da variável com o VTSCADA).



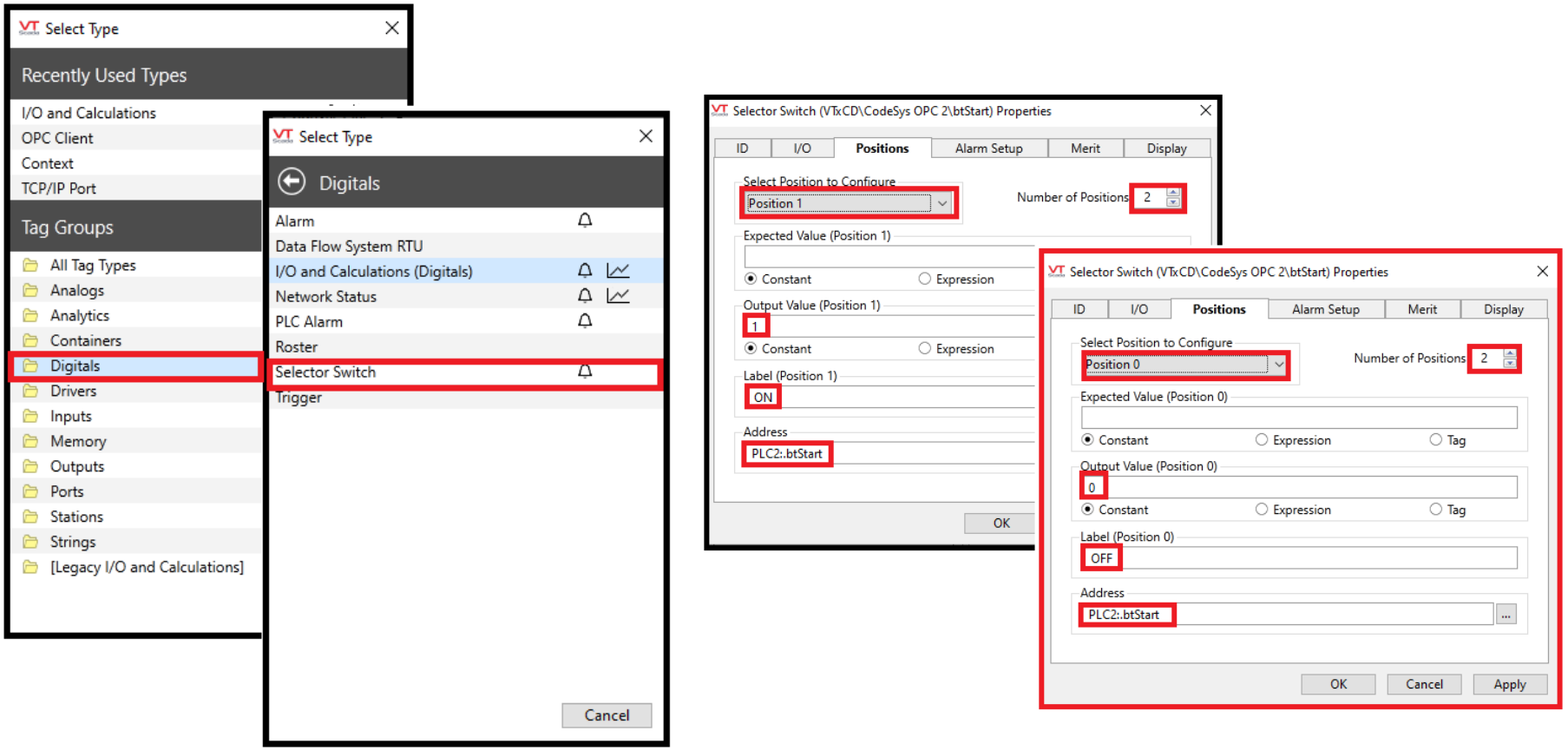
Comunicação programa LADDER

Quando formos exportar, basta selecionar também o programa LADDER no POU para exportar junto com as variáveis:



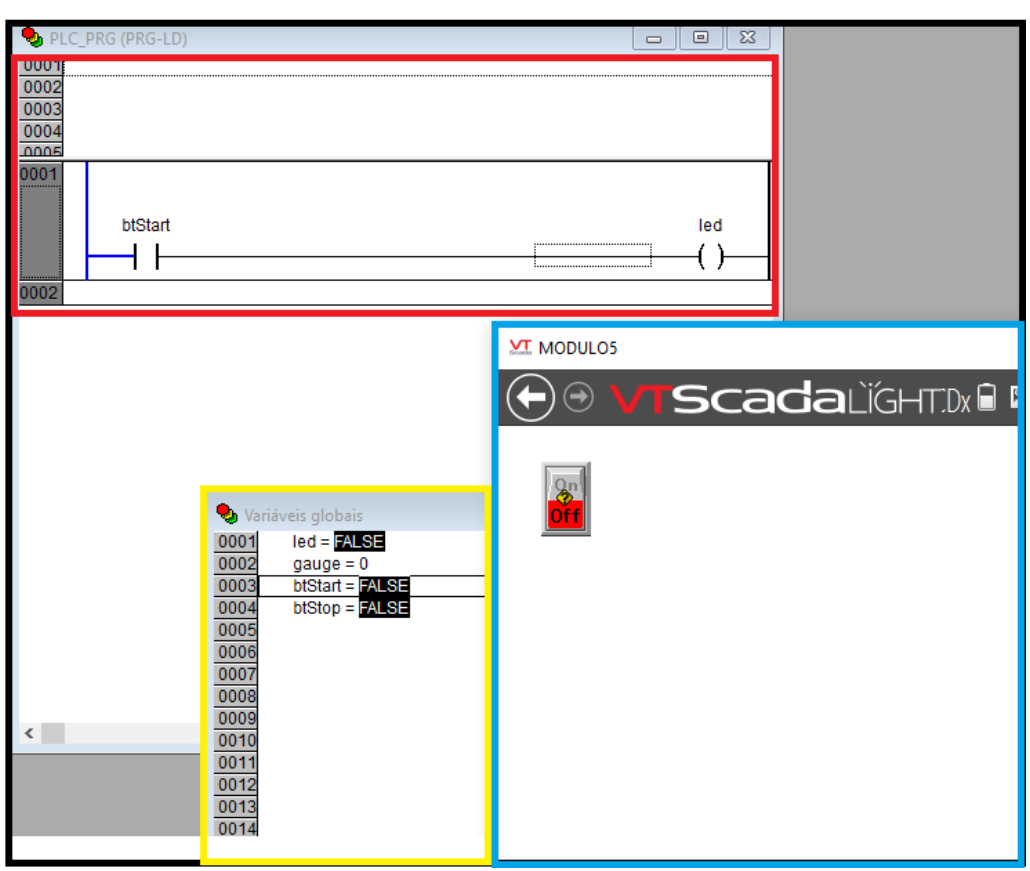
Comunicação programa LADDER

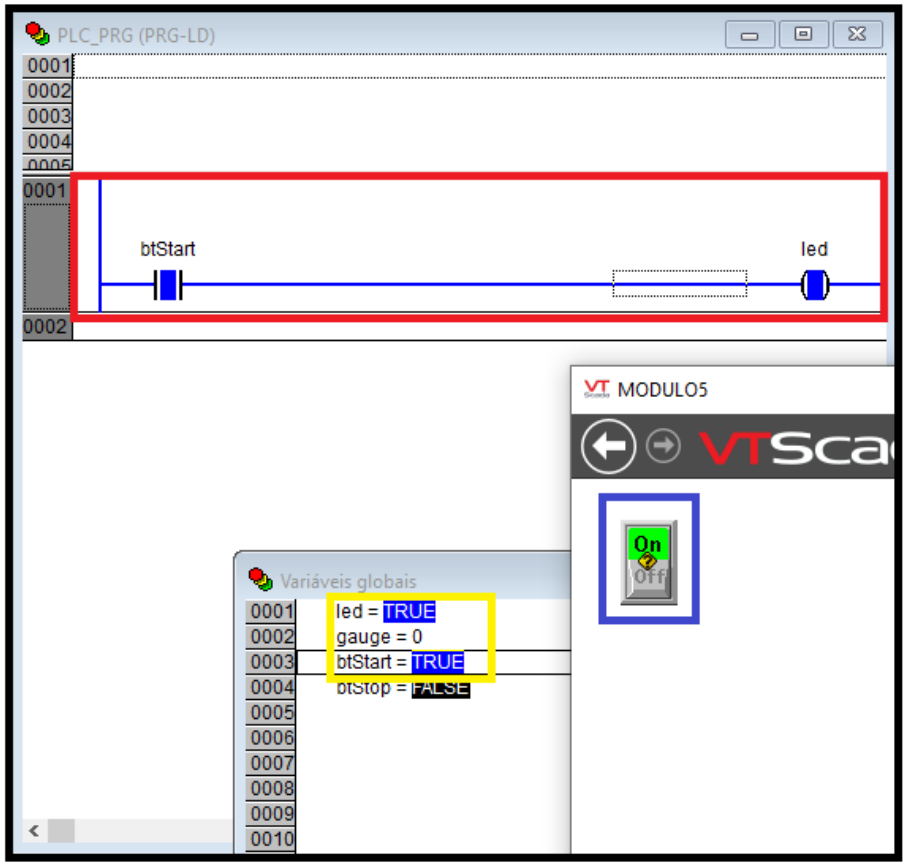
Fazemos a tag para nova variável (desta vez digital e selector switch) e um botão para ela:



Comunicação programa LADDER

O programa funcionará normalmente:





FT primeira ordem

Agora iremos implementar um sistema de primeira ordem no nosso programa LADDER. Utilizaremos como base o programa anterior para mais facilidade:

Um sistema de primeira ordem tem a seguinte função de transferência:

$$Y(s)/X(s) = k.1/(s+a)$$

Aplicando a função Z inversa para sistema discretos temos que:

$$Y(t) = K.((1-p)y\_n\_1(t) + p.U(t))$$
$$\text{Com } p = deltaT/(1/a + deltaT)$$

FT primeira ordem

Vamos utilizar tudo que aprendemos até aqui para essa parte, por isso o passo a passa de criar a IHM no VTSCADA, além de declaração de variável globais e outros serão omitados.

Começamos criando o bloco funcional com o sistema de primeira ordem:

Primeira_Ordem (FB-ST)

```
0001 FUNCTION_BLOCK Primeira_Ordem
0002 VAR_INPUT
0003     TauFiltro:REAL;
0004     Ganho:REAL;
0005     TS:REAL;
0006     u:REAL;
0007 END_VAR
0008 VAR_OUTPUT
0009     Y:REAL;
0010 END_VAR
0011 VAR
0012     y_n:REAL;
0013     y_n_1:REAL;
0014     a:REAL;
0015 END_VAR
```

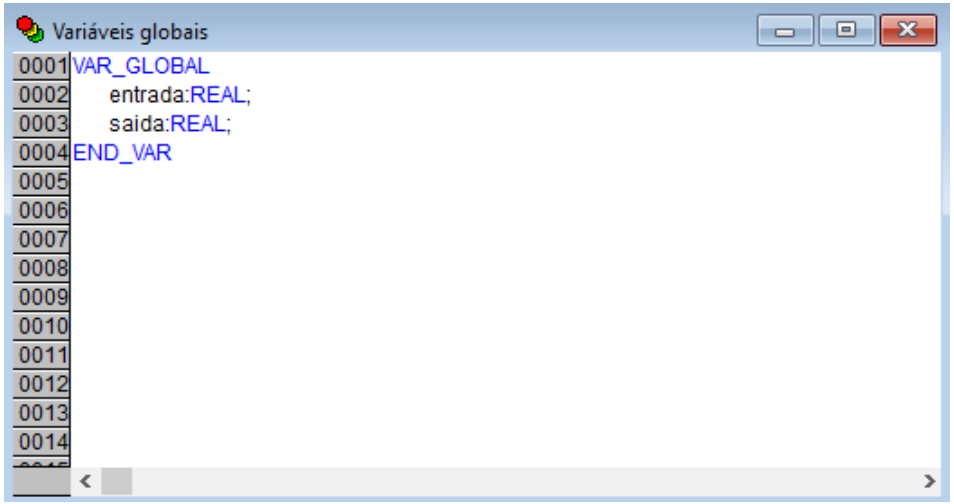
```
0001 a:= TS/(TauFiltro + TS);
0002 y_n := (1-a)*y_n_1 + a*u;
0003 y_n_1 := y_n;
0004 Y := ganho*y_n;
0005
0006
```

Lembrando que nosso sistema é discreto, logo precisamos sempre armazenar os valores de saída.

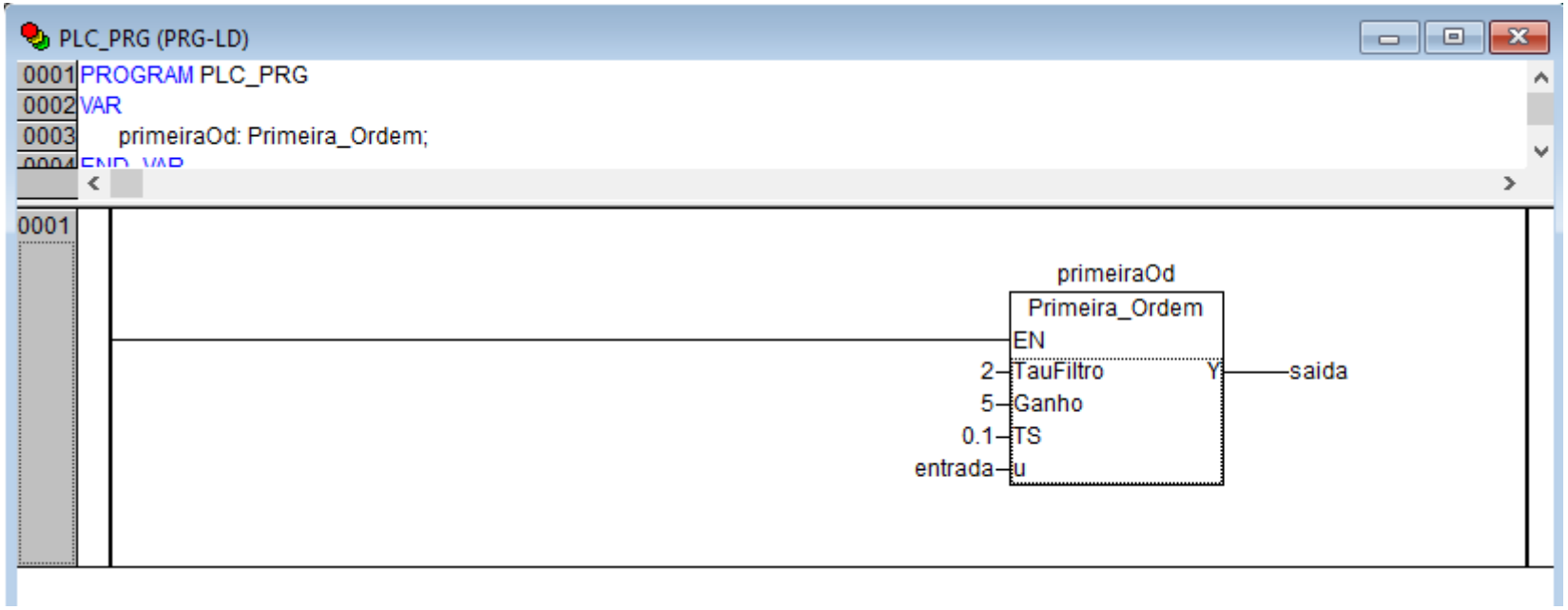
Usamos a função do nosso sistema no tempo como obtido no slide anterior.

FT primeira ordem

Criamos as variáveis globais e a IHM no VTSCADA:
Um bloco para inserir o valor de entrada e um gauge para visualizarmos a saida.

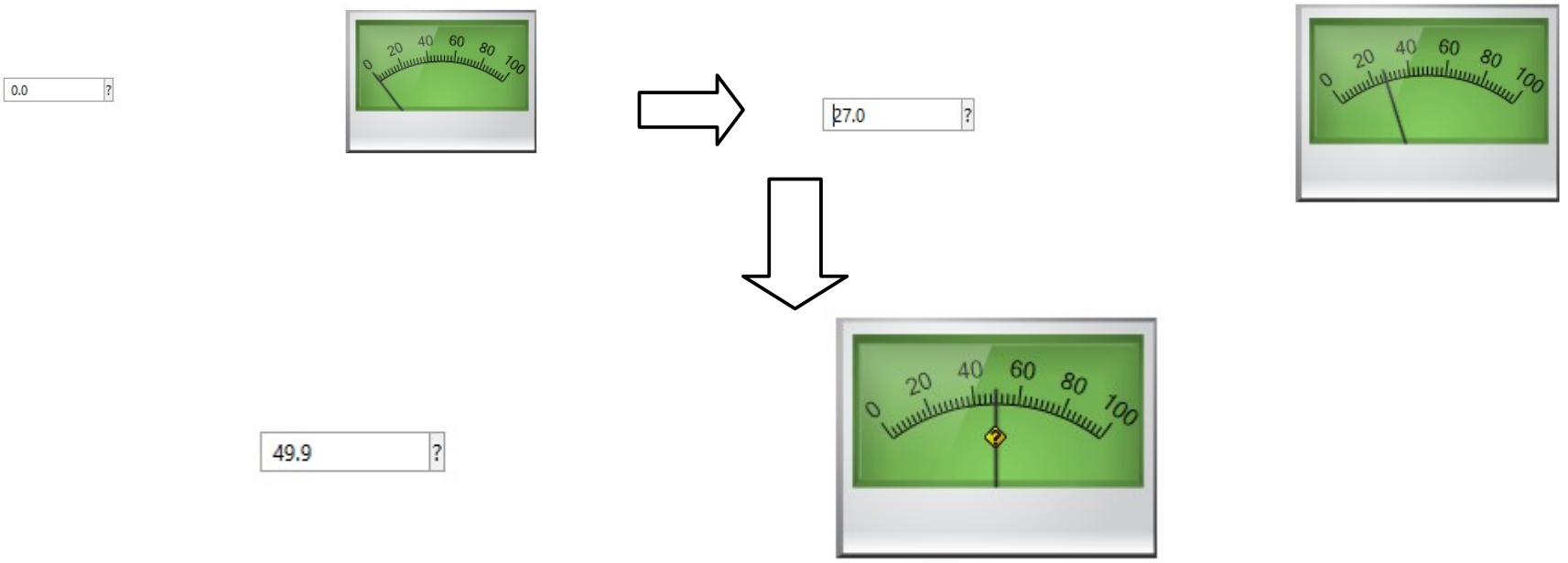


Criamos nosso programa em LADDER com uma instrução do nosso bloco programável:
A entrada é o valor objetivo e a saída e o valor atual da nossa função. O ganho, Tau e DeltaT (TS) são valores definidos pelo usuário.



FT primeira ordem

Depois de fazer todas conexões no OPC, PLC, configurar as saidas do Codesys e criar as tags no VTSCADA, o sistema está pronto: (Com o ganho 5, adicionamos a entrada 10 e a saída tende a chegar ao valor 50 com o tempo:

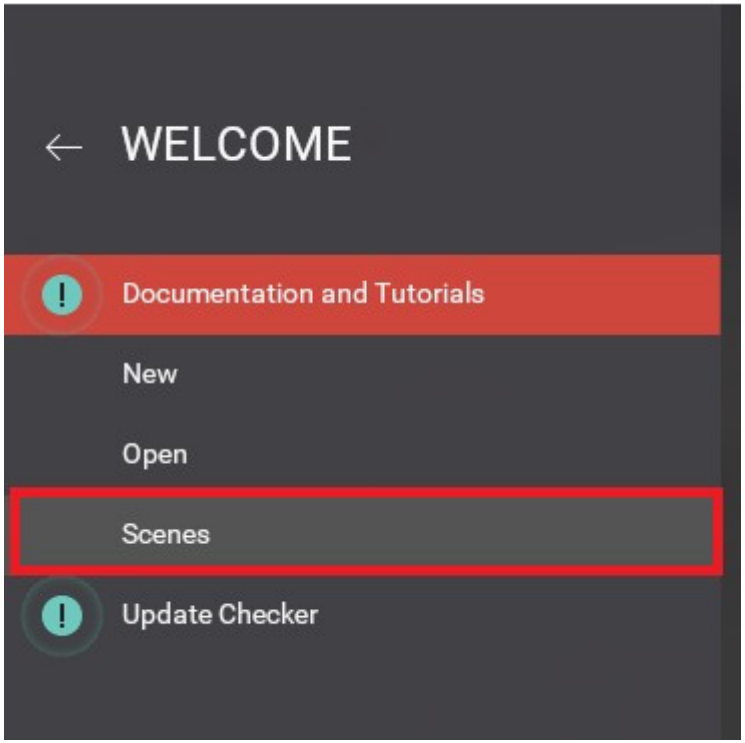


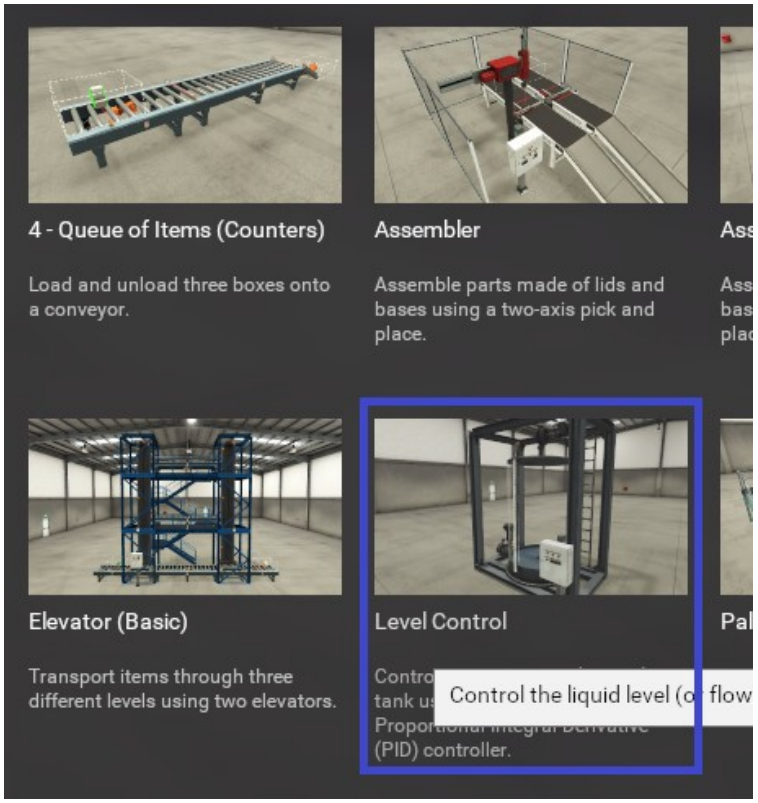
Controle de nível de tanque com Factory IO

Agora vamos fazer um controle de nível. Para isso vamos utilizar uma cena do Factory IO, programar com o CodeSys e gerenciar pelo VTScada.

Fatory IO

Vamos começar selecionando a cena que iremos utilizar no factory IO: Clicamos em Scenes e depois escolhemos a cena Level Control:





Factory IO

A cena em questão será a seguinte:

Um tanque de água com sensores de níveis & válvulas de enchimento e descarga.

Desta forma, vamos programar no CodeSys para podermos gerenciarmos esse tanque a partir de um SetPoint para o volume de água.

CodeSys

Criamos um programa no CodeSys como já ensinado. Para o controle de nível, iremos utilizar um controlador de primeira ordem como vimos no slide 20. Esse controlador será responsável por manter o nível da água no setpoint desejado.

As principais diferenças entre o controlador usado anteriormente e o do controle de nível é que a entrada do controlador é o erro (diferença do valor do setpoint e o nível atual da água) e que necessitamos capturar o DeltaTime real do sistema.

Variáveis CodeSys

Desta forma iremos utilizar as seguintes variáveis:

Variáveis globais

```
0001 VAR_GLOBAL
0002     Descarga: REAL;
0003     Cagra: REAL;
0004     Flow: REAL;
0005     Level: REAL;
0006     PV:REAL;
0007     Reset:BOOL;
0008     ResetLight:BOOL;
0009     SetPoint: REAL;
0010     SP:REAL;
0011     Start:BOOL;
0012     SL:BOOL;
0013     Stop:BOOL;
0014     StopLight:BOOL;
0015     Deltat:REAL;
0016     SaidaBloco: REAL;
0017     erro: REAL;
0018     Ativo:BOOL;
0019 END_VAR
0020
```

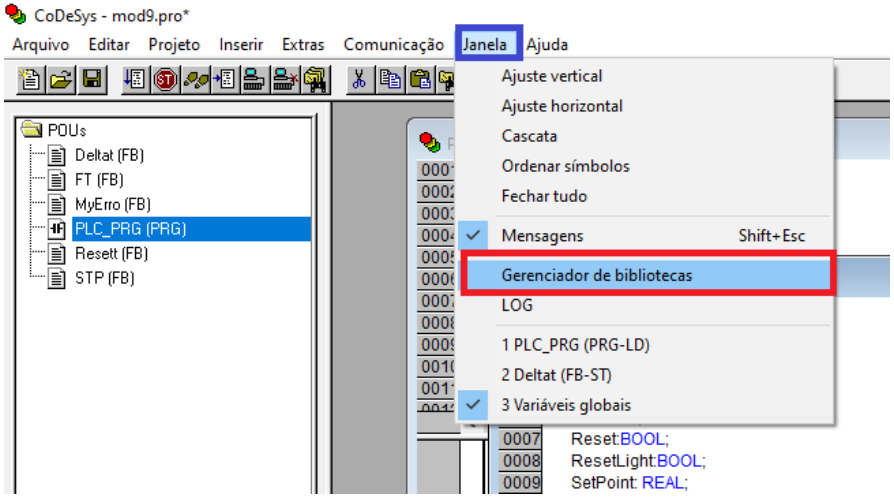
Alguma variáveis são oriundas da nossa cena, como as válvulas de carga e descarga, o sensor de nível, o set point, os botões e outras.

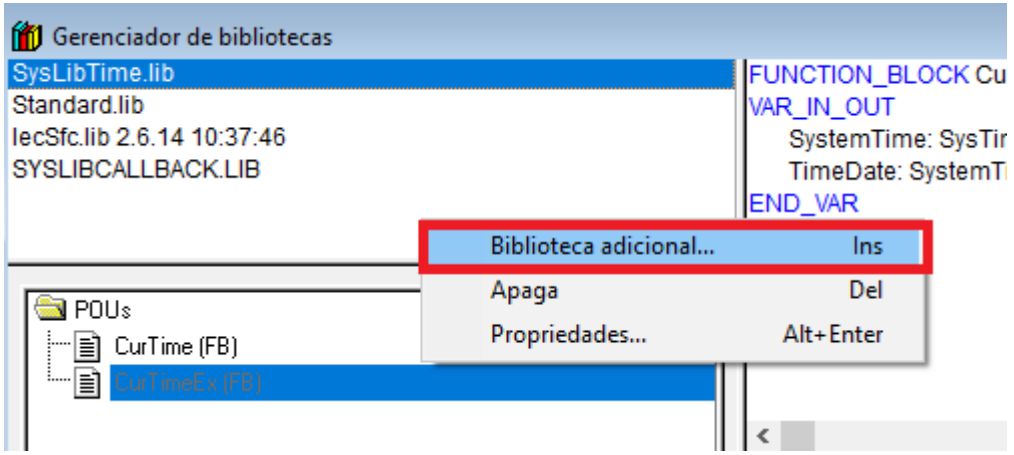
A variável erro será necessária para definimos a diferença entre os valores de nível e setpoint, enquanto a variável ativo será usada para definir se o sistema está ligado ou não.

Deltat será usada para definirmos o tempo entre amostras.

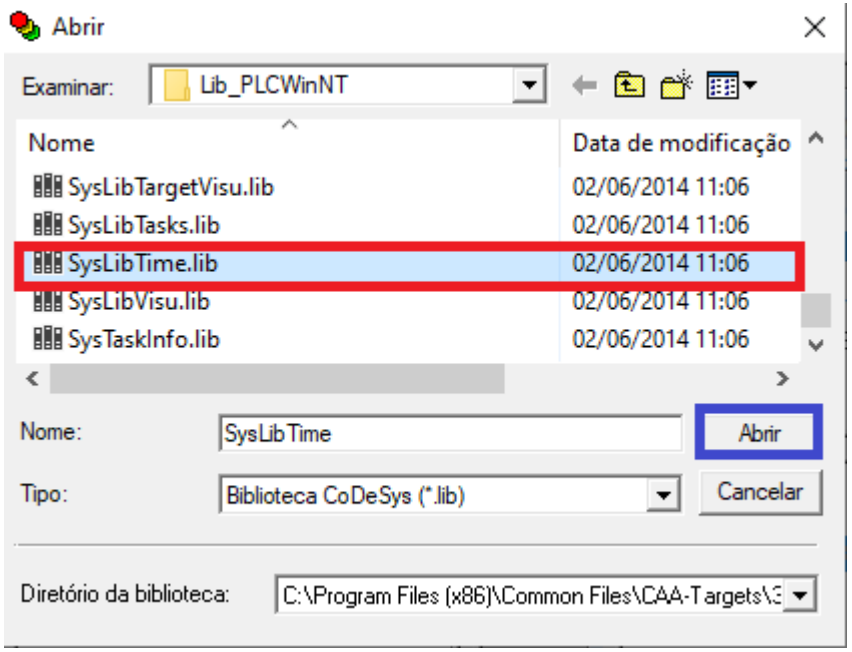
Importando biblioteca

A primeira coisa que iremos fazer é obter o deltaTime, para isso devemos importar a biblioteca systimelib da seguinte maneira:



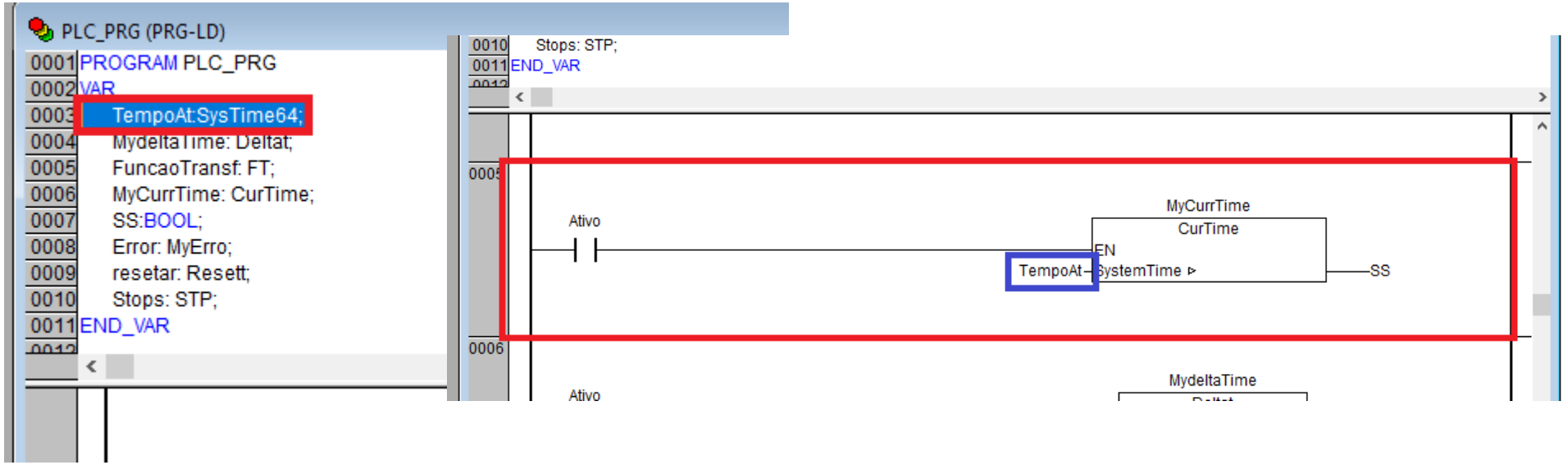


Clicamos em janelas e em seguida gerenciador de bibliotecas. Na nova aba clicamos com botão direito e em biblioteca adicional. Em seguida selecionamos a biblioteca SysLibTime.lib e em abrir. Agora possuímos um bloco funcional capaz de nos da o tempo atual.



Obtendo o tempo

No progama principal criamos uma variável do tipo SysTime64 em seguida criamos uma rede para colocarmos essa variável no novo bloco currTime:



Obtendo o DeltaTime

Agora vamos obter o delta tempo. Criamos um bloco funcional da seguinte maneira:

Deltat (FB-ST)

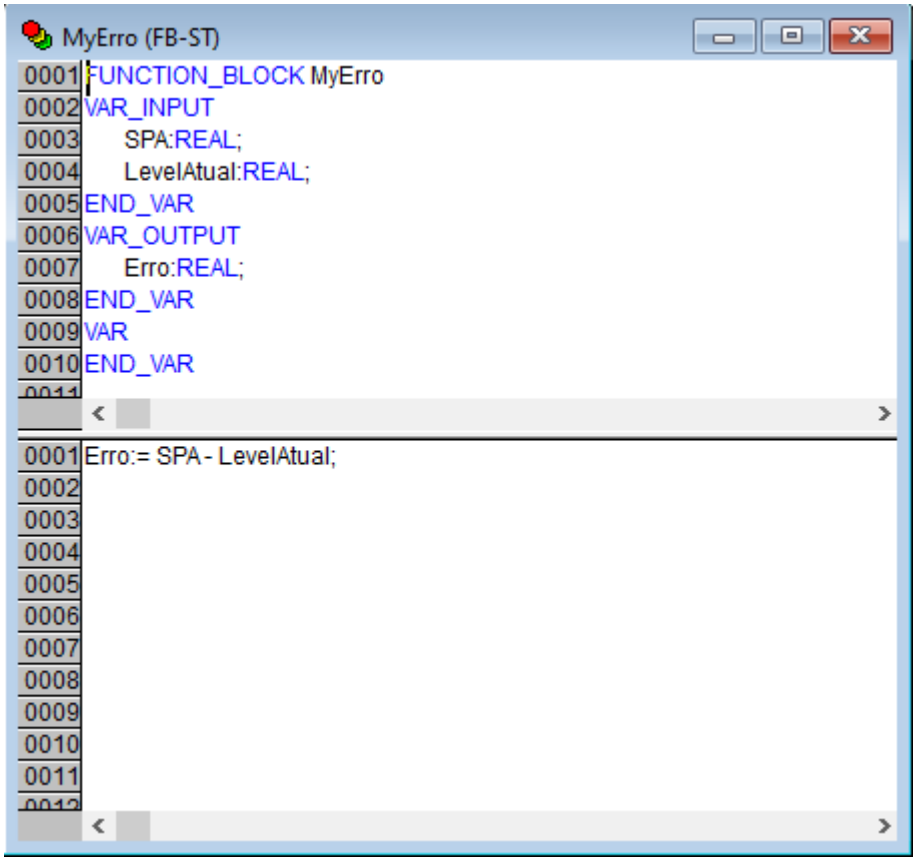
```
0001 FUNCTION_BLOCK Deltat
0002 VAR_INPUT
0003     TempoAtual:SysTime64;
0004 END_VAR
0005 VAR_OUTPUT
0006     DeltaTime:REAL;
0007 END_VAR
0008 VAR
0009     TempoAnterior:SysTime64;
0010 END_VAR
```

```
0001 DeltaTime:=(TempoAtual.ulLow -TempoAnterior.ulLow)/100000.0;
0002 TempoAnterior:=TempoAtual;
0003
0004
0005
0006
0007
0008
0009
0010
0011
```

Pegamos o valor do sistema adquirido pela varável criada anteriormente e fazemos a diferença entre o tempo atual e anterior.

Erro

Agora criamos o bloco funcional do erro:



Controlador

Por fim criamos o bloco do controlador como foi feito em alguns slides anteriores :

FT (FB-ST)

```
0001 FUNCTION_BLOCK FT
0002 VAR_INPUT
0003     TauFiltro:REAL;
0004     Ganho:REAL;
0005     TS:REAL;
0006     u:REAL;
0007 END_VAR
0008 VAR_OUTPUT
0009     Y:REAL;
0010     ValorCarga:REAL;
0011     ValorDescarga:REAL;
0012 END_VAR
0013 VAR
0014     y_n:REAL;
0015     y_n_1:REAL;
0016     a:REAL;
0017 END_VAR
0018
0019
```

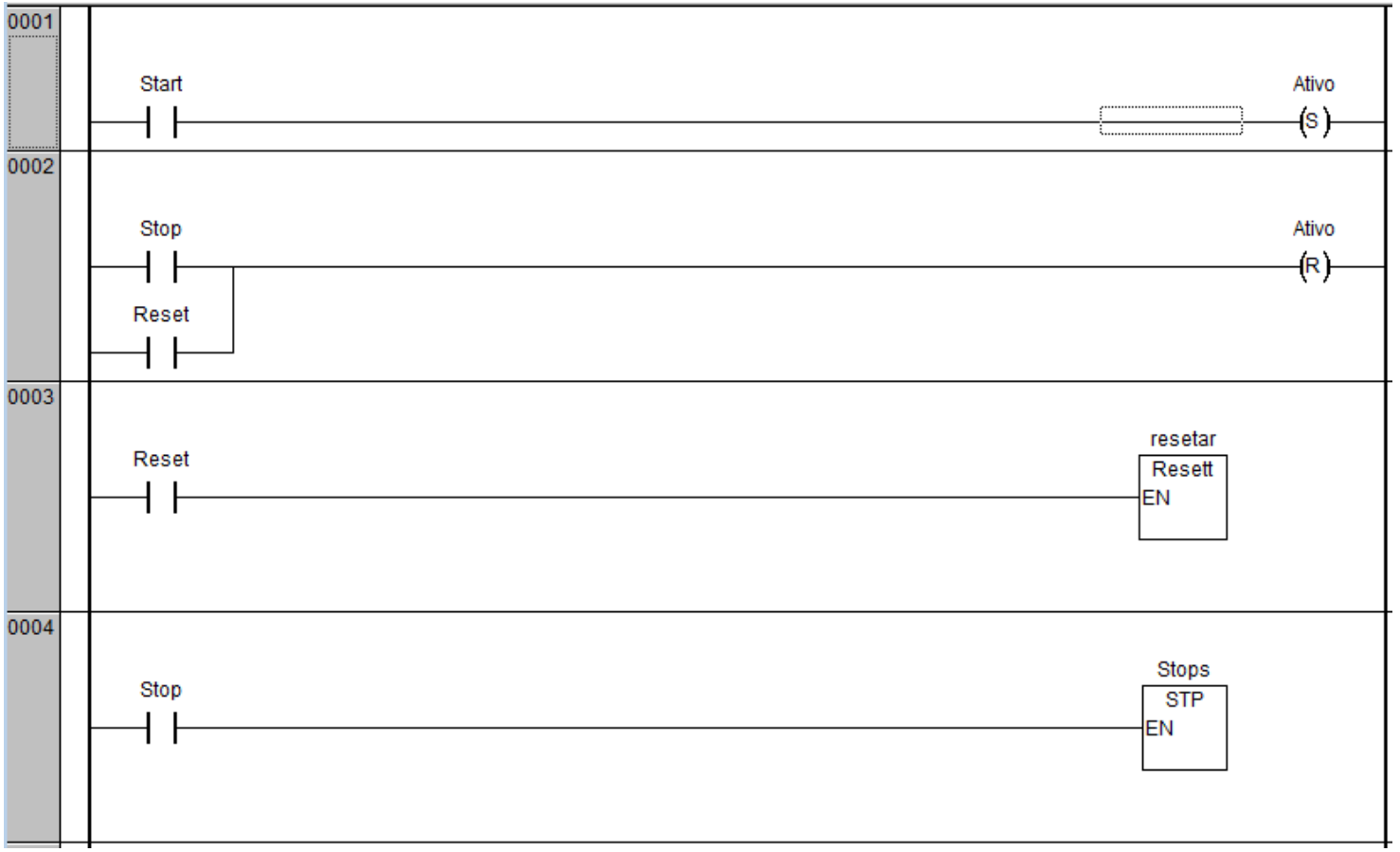
```
0001 a:= TS/(TauFiltro + TS);
0002 y_n := (1-a)*y_n_1 + a*u;
0003 y_n_1 := y_n;
0004 Y := ganho*y_n;
0005 IF (Y>0) THEN
0006     ValorDescarga := 0;
0007     IF(Y<=10) THEN
0008         ValorCarga := Y;
0009     ELSE
0010         ValorCarga:=10;
0011     END_IF
0012 ELSE
0013     ValorCarga := 0;
0014     IF(Y>=-10) THEN
0015         ValorDescarga := -Y;
0016     ELSE
0017         ValorDescarga:= 10;
0018     END_IF
0019 END_IF
0020
```

Controlador

Reparamos que agora devemos implementar a lógica de uso das válvulas. Assim quando o valor de saída for positivo utilizamos a válvula de carga e quando for negativo usamos a válvula de descarga. As mesmas tem limite de de 0 a 10, desta forma, ao ultrapassar esse valor na saída, limitamos a 10 utilizando a condicional IF.

Botões

Adicionamos as lógicas dos botões:



Botões

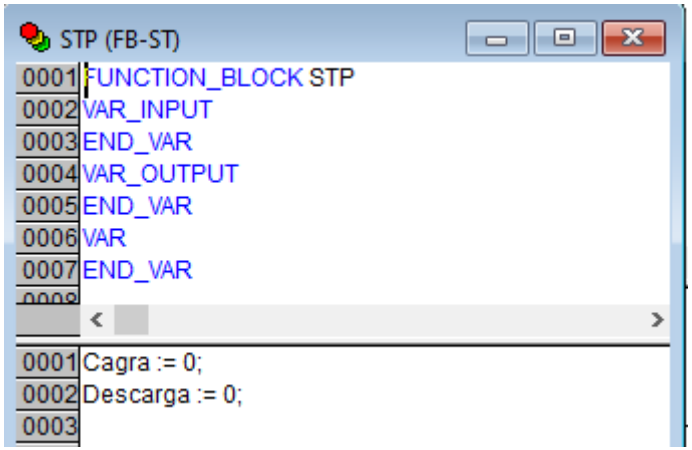
O botão Start altera o valor de ativo para TRUE, mostrando ao programa que o mesmo está ligado.

O botão Stop altera o valor de ativo para FALSE e energiza o bloco stops.

O botão reset altera o valor de ativo para FALSE e energiza o bloco resetar.

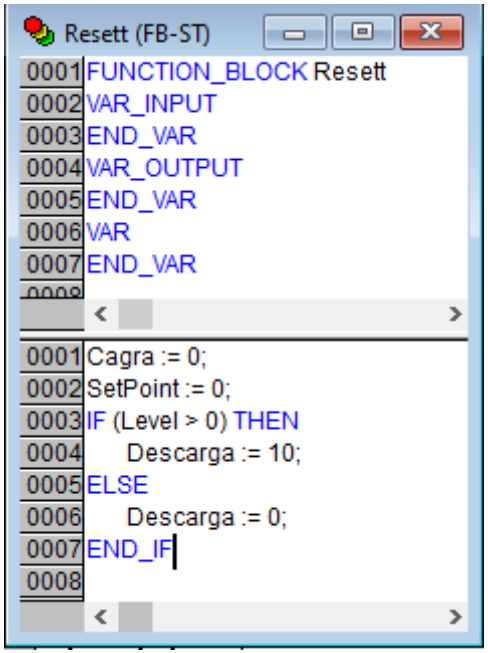
Veremos agora esses blocos.

Bloco Stops



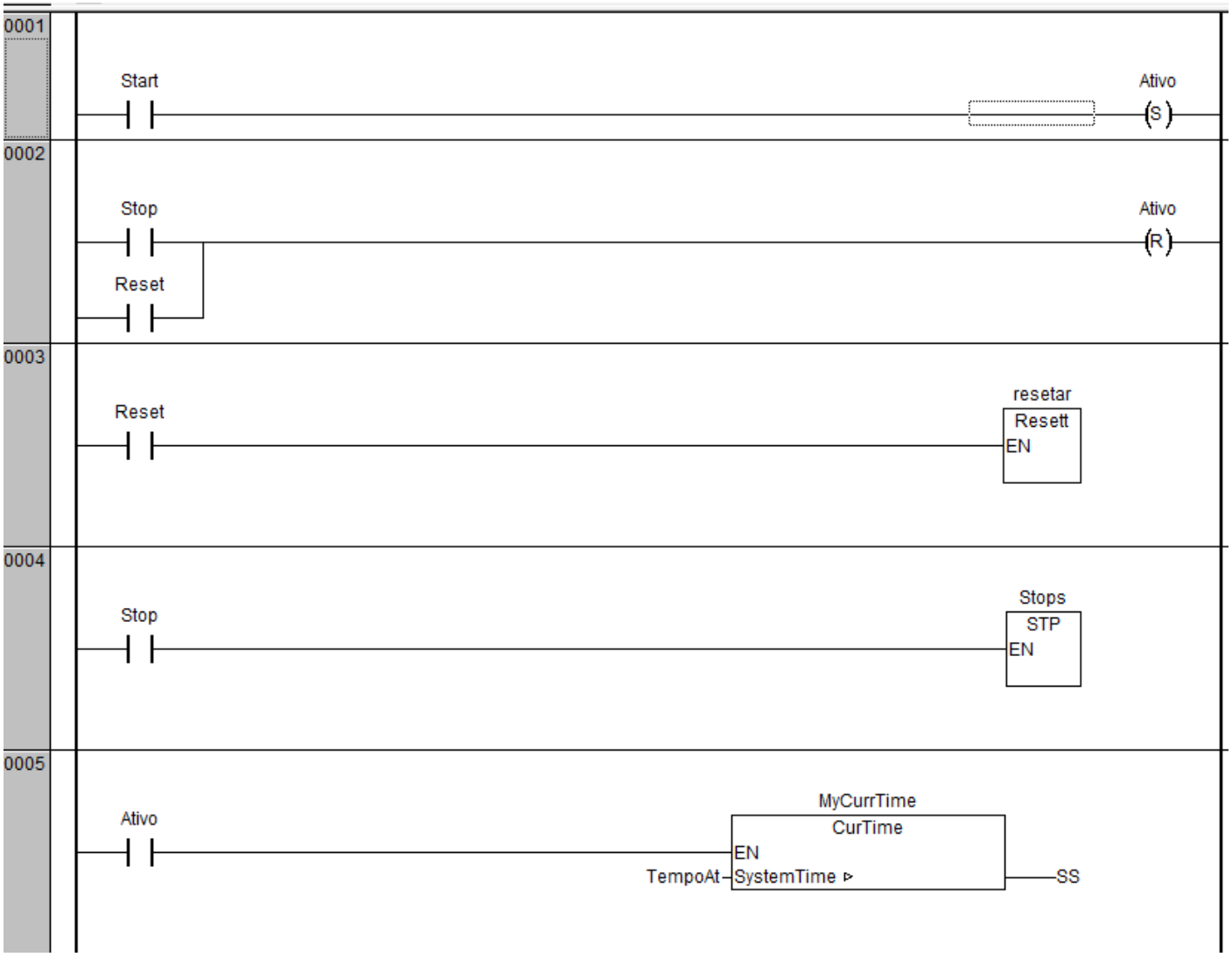
O bloco stops seta os valores das válvulas para 0, parando o enchimento ou descarga do tanque.

Bloco resetar

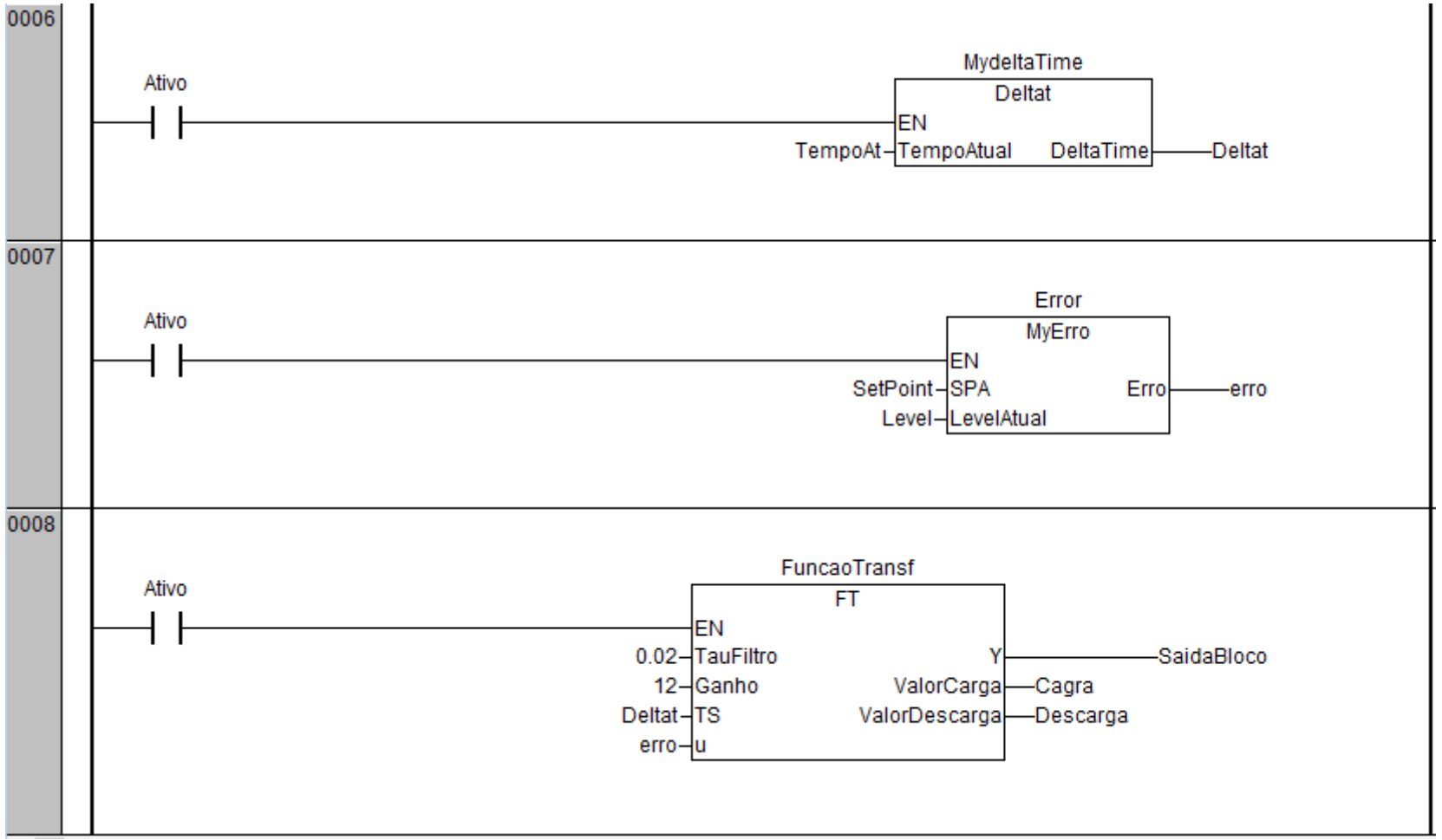


O bloco resetar zera as variáveis e descarrega o tanque completamente.

Programa completo

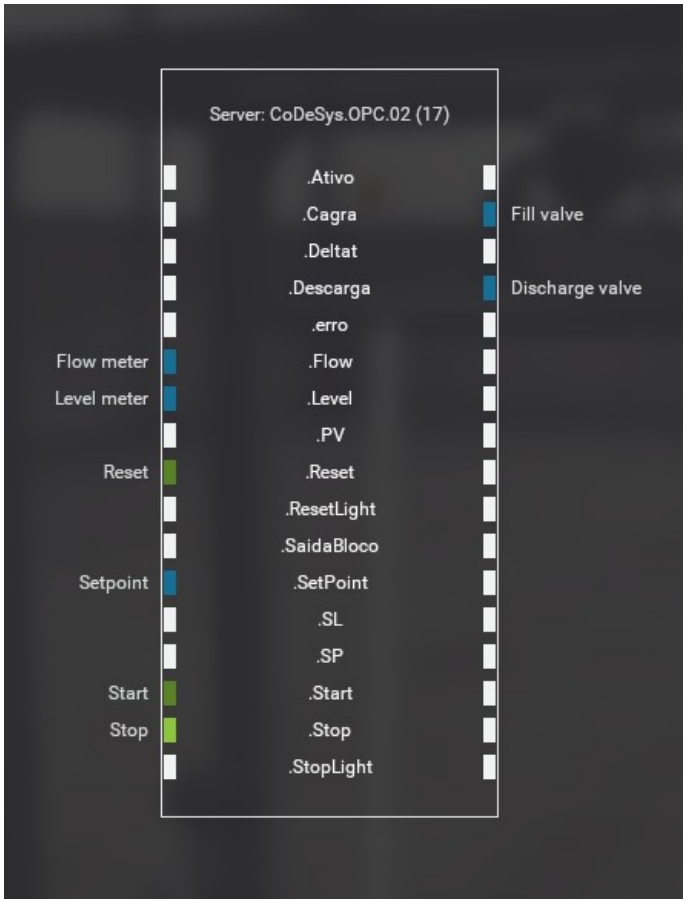


Programa completo



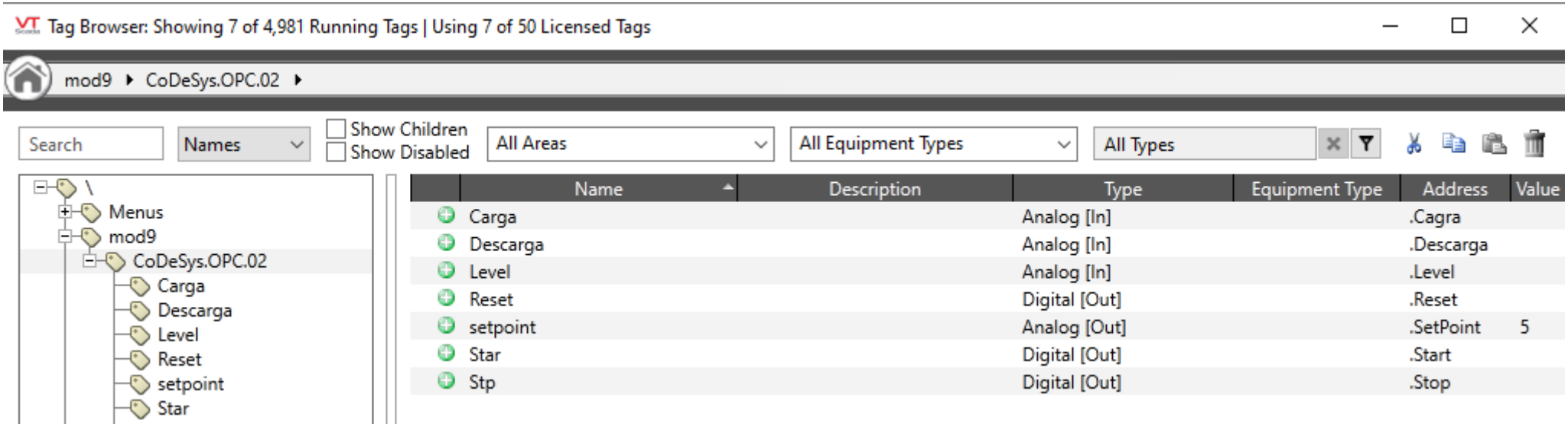
Ligações Factory IO

Agora exportamos as variáveis globais do CodeSys, ativamos o PLC e fazemos a conexões com o facory IO:



VTscada

Agora vamos ao Vtscada criar o leyault do supervisório. Antes disso criamos as tags de container e driver do OPC e em seguida das varáveis que iremos utilizar:



Leyault

Criamos o Leyault com o que desejamos obter de informação e controlar. Linkamos as tags.

Utilizamos os 3 botões, o gauge para o nível, dois visualizadores para os sensores das válvulas e uma entrada de texto para nosso SetPoint.

Terminado

Pronto. Terminamos a configuração de cada um dos nosso softwares para o controle de nível. Lembrando sempre de configurar as conexões de cada software, verificar se o PLC WIN NT está ativo, linkar as variáveis e tagse exportar as variáveis.

Esse é o fim desse tutorial. Nele aprendemos a utilizar o Vtscada e implementar um aplicação com o Factory IO e o CodeSys.

Fim