CONFIGURAÇÃO DO SCADA WINLOG

INTRODUÇÃO

Pretende-se neste documento fazer uma curta introdução ao SCADA WinLog, bem como discutir alguns aspetos relacionados com a respetiva configuração.

SCADA WINLOG

O WinLog disponibiliza um ambiente de desenvolvimento que permite criar aplicações SCADA (simples) para supervisionar e monitorizar processos industriais.

O WinLog (3.02) pode ser obtido no seguinte [link][1]. Trata-se se uma versão LITE mas totalmente funcional do SCADA WinlogPro, que tem como única limitação o número máximo de **tags** (*gates*) que se podem criar, neste caso 24 e cujo tempo de execução da aplicação está limitado a 15 minutos (não há limite para o tempo de desenvolvimento). Para o projeto que irá desenvolver estes valores são suficientes.

O WinLog executa num ambiente do tipo WinXX [xx=7..10]. Para ambientes de base outro tipo (ex. Mac OS) recomenda-se a sua instalação numa máquina virtual.

Embora a utilização do WinLog seja bastante intuitiva, recomenda-se a seguinte abordagem:

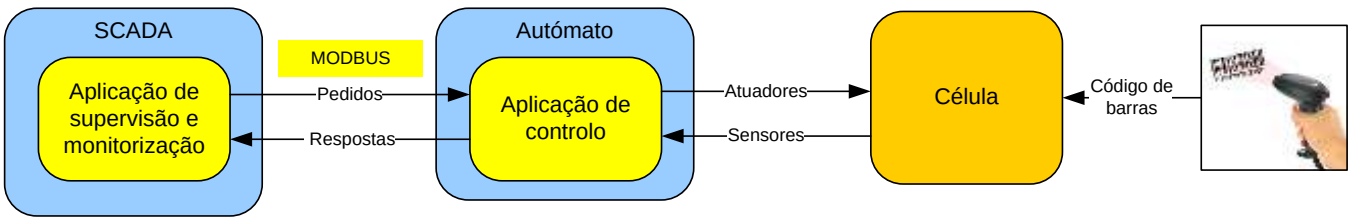
- Instalar a aplicação.
- Ler o documento Project Manager disponível no seguinte [link]. Este documento permite ter uma visão global das funcionalidades do SCADA.

Se necessário pode obtida informação complementar em:

- Manuais online (Help)
- Outros manuais disponíveis no seguinte [link]. Os manuais mais relevantes para este trabalho são:
  - User Manual - Application Builder
  - User Manual - Gate Builder
  - User Manual - Template Builder
  - User Manual - Code Builder
- Tutoriais na Web (Youtube: search WinLog)
- Site do fabricante [link]

COMUNICAÇÃO ENTRE O WINLOG E O AUTÓMATO

A aplicação de supervisão irá trocar dados com aplicação de controlo (i.e. autómato) utilizando o protocolo de comunicações Modbus/TCP. Este último aspeto é um requisito imposto ao trabalho.



Em conctreto, o SCADA Winlog vai enviar pedidos para o autómato, através do protocolo Modbus/TCP, para ler ou escrever em posições de memória do autómato (ver adiante), e assim, de forma indireta, supervisionar e monitorizar a célula.

[1] O acesso ao site ftp://labs-automacao.fe.up.pt necessita de uma ligação VPN se for realizado fora da Feupnet.

PROTOCOLO MODBUS

O Modbus é um protocolo de comunicações muito utilizado para a troca de dados entre equipamentos industriais e que opera segundo um modelo Cliente-Servidor (ou Mestre-Escravo). O cliente (um equipamento) envia pedidos para um servidor (outro equipamento), que por sua vez responde a esses pedidos.

O protocolo é muito simples e permite essencialmente ler ou escrever em dois tipos de dados: **bits** e **registos** (cada registo tem 16 bits). Os bits podem ser apenas de leitura, denominados **Discrete Inputs** ou de leitura e escrita, denominados **Coils**. Por sua vez os registos podem também ser apenas de leitura, denominados **Input Registers** ou de leitura e escrita, denominados **Holding Registers**.

Cada bit ou registo possui um endereço numérico único na gama **0…9999**. Para ler ou escrever num bit ou registo é preciso invocar (i.e. 'chamar') uma **função** específica. Cada função é identificada por um **número** (inteiro).

Tipicamente uma função indica qual o tipo de dados a que ser pretende aceder (bit ou registo), qual o endereço dos dados e qual o tipo de operação que se pretende realizar sobre os dados: ler ou escrever (neste último caso é necessário também enviar os valores a escrever). A função é invocada pelo cliente junto do servidor. As respostas enviadas pelo servidor são essencialmente de dois tipos: o pedido foi realizado com sucesso (no caso de um pedido de leitura implica retornar os valores solicitados); o pedido não pode ser realizado (i.e. porque ocorreu um erro).

No caso do WinLog o processo de configuração dos pedidos Modbus é muito simples, sendo representado por uma mnemónica do tipo: **F:EEEE**, em que **F** é o número da função e **EEEE** o endereço do bit ou registo. No caso de **F** estão definidos os seguintes valores: **1** (para ler ou escrever em bits); **2** (para ler bits); **3** (para ler ou escrever em registos); **4** (para ler registos).

Assim, a representação **4:0012** significa que se pretende ler o registo nº 12, enquanto que a representação **1:0002** significa que se pretende ler ou escrever no bit nº 2.

O envio dos pedidos (e a receção de respostas) pode ser realizada de duas formas:

- Através de uma interface de comunicação série (porta série RS-232 ou RS-485), denominando-se a respetiva implementação Modbus Série (tem duas variantes Modbus ASCII e Modbus RTU).
- Através da Internet (TCP/IP), denominando-se a respetiva implementação Modbus/TCP. Esta é a opção que vai ser utilizada neste trabalho porque o autómato dispõe de um servidor Modbus/TCP integrado que é automaticamente ativado quando se configura a respetiva carta Ethernet.

Pode encontrar no seguinte [link] um documento com uma descrição mais detalhada do funcionamento do protocolo Modbus. A utilização do Modbus/TCP no Winlog está coberta no seguinte manual [link] (ou no help online).

CONFIGURAÇÃO DO PROJETO

Assume-se que nesta fase já leu o manual **Project Manager**.

Por forma a simplificar o desenvolvimento da aplicação, fornece-se um projeto já pré-configurado que pode ser obtido no seguinte [link].

Grave o projeto numa pasta. Depois de abrir o WinLog importe o projeto que descarregou (**Project-> Import**). Em concreto, foram definidos neste projeto:

- O dispositivo autómato (**Device 1**) com que a aplicação irá comunicar.
- O canal de comunicação (**Channel 1**) entre a aplicação e o autómato (protocolo Modbus/TCP). A aplicação SCADA será o cliente e o autómato o servidor.

Deverá configurar os restantes elementos do projeto, nomeadamente os elementos relacionados com as gates, alarmes e históricos.

CONFIGURAÇÃO DAS TAGS (GATES)

Para que a aplicação SCADA possa trocar dados com o autómato é necessário ter em conta os seguintes aspetos:

- **Servidor Modbus do autómato**. O servidor Modbus do autómato permite apenas ler ou escrever em registos. Estes registos correspondem às posições de memória **%MW** do autómato. Assim, quando, por exemplo, se está a ler o registo **10** do servidor, está-se a ler a posição de memória **%MW10** do autómato. De forma análoga, quando se pretende escrever no registo **20** está-se a escrever na posição de memória %**MW20** do autómato.
- As tags (**gates**) às quais correspondem variáveis no autómato só podem ser do tipo **3:xxxx** ou **4:xxxx** (i.e. registos). Assim se se pretender criar uma tag de leitura correspondente à posição de memória **%MW10** do autómato, esta tag deve ser configurada no WinLog com o endereço **4:0010**. De forma análoga, quando se pretende criar uma tag de escrita correspondente à posição de memória **%MW20** autómato, esta tag deve ser configurada no WinLog com o endereço **3:0020**.
- **Tipos de dados associados às tags**. De forma geral as tags criadas no WinLog devem ser do tipo **Numeric** com um tipo de dados **U_INT** (unsigned int).
- **Número de tags (gates) no WinLog**. Como o número de tags (gates) que se podem criar para comunicar com o autómato está limitado a 24 é necessário alguma algum cuidado quanto à forma como estas são definidas, caso contrário, e tendo em conta as características deste trabalho, rapidamente pode ficar sem tags disponíveis[2]. Para evitar este problema propõe-se a utilização de um processo de **compactação**, descrito de seguida:

Exemplo: admita que tem 4 variáveis do tipo BOOL no autómato (denominadas V1, V2, V3 e V4) que pretende aceder no SCADA. Em vez de criar 4 tags do tipo **Digital** (uma para cada variável), pode criar uma única tag do tipo **Numeric** (registo de 16 bits) em que cada bit do registo representa a variável que pretende aceder. Por exemplo, pode criar uma %MW com a seguinte organização e associa-la a uma tag:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
_	_	_	_	_	_	_	_	_	_	_	_	V4	V3	V2	V1

Com este tipo de abordagem pode "armazenar" em cada tag (do tipo **Numeric**) até 16 variáveis do tipo BOOL.

Do lado do SCADA é possível aceder a cada um dos bits desta tag através do conceito de **máscara**. Em concreto é possível aplicar à tag um operador do tipo *´At least one bit == value´* para extrair o valor do bit em causa e ignorando os valores dos restantes bits (marcados como **X**). Assim e recorrendo a este exemplo, para detetar se a variável **V3** toma o valor 1 teria que se aplicar a máscara da figura à direita.

Como alternativa, é possível implementar código (componente *Code* do WinLog) que realize uma função equivalente de extração dos bits.

FIM

[2] Este limite aplica-se apenas às tags comunicação. Não há limite definido para as tags internas.