BIBLIOTECA WYDRIVER V1.1

EXEMPLO DE USO

Tutorial de utilização da Biblioteca WyDriver para comunicação com painéis de chamada Wyma no sistema operacional Linux.

Rev.0 – 9/10/2017 – Revisão inicial

Índice

1. Introdução

A Biblioteca de Comunicação WyDriver é um arquivo de extensão de aplicativo (JAR) com diversas funções prontas, que embutem e simplificam a implementação de um protocolo de comunicação.

A biblioteca foi concebida usando a tecnologia Java 1.6.0_45, portanto, para sua utilização será necessário possuir no mínimo a mesma versão, além das ferramentas de desenvolvimento. A versão utilizada do Java SE esta disponível em:

https://www.oracle.com/technetwork/java/javase/downloads/java-archive-downloads-javase6-419409.html

Para o desenvolvimento utilizamos o Eclipse Luna que esta disponível em:

https://www.eclipse.org/downloads/packages/release/luna/sr2/eclipse-ide-java-developers

2. Requisitos para iniciar

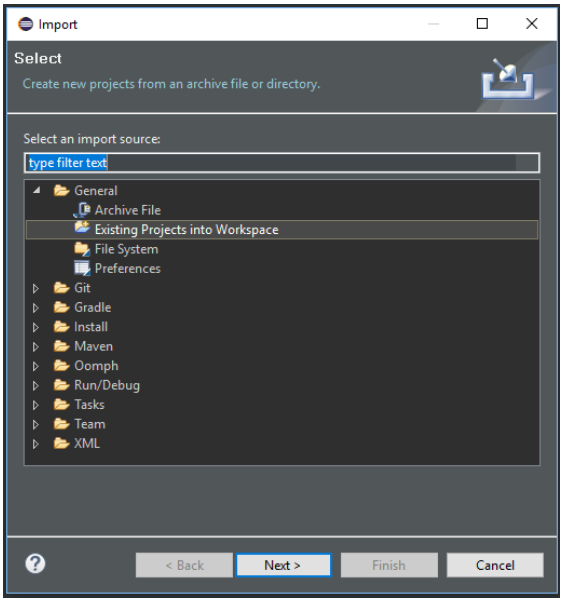
Antes de iniciar o projeto, verifique se recebeu o arquivo do projeto:

project.zip

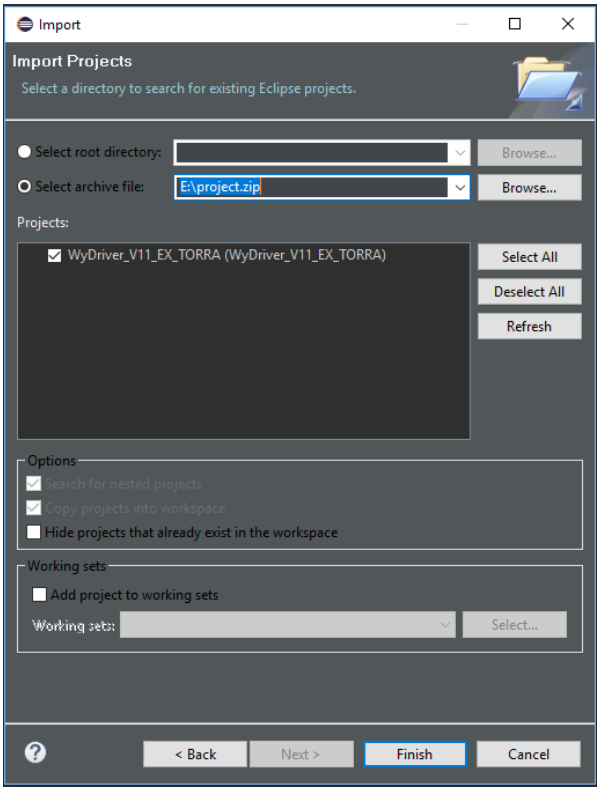
3. Importando o projeto

1 – Abra o Eclipse ou outra IDE que suporte a versão 1.6 do Java.

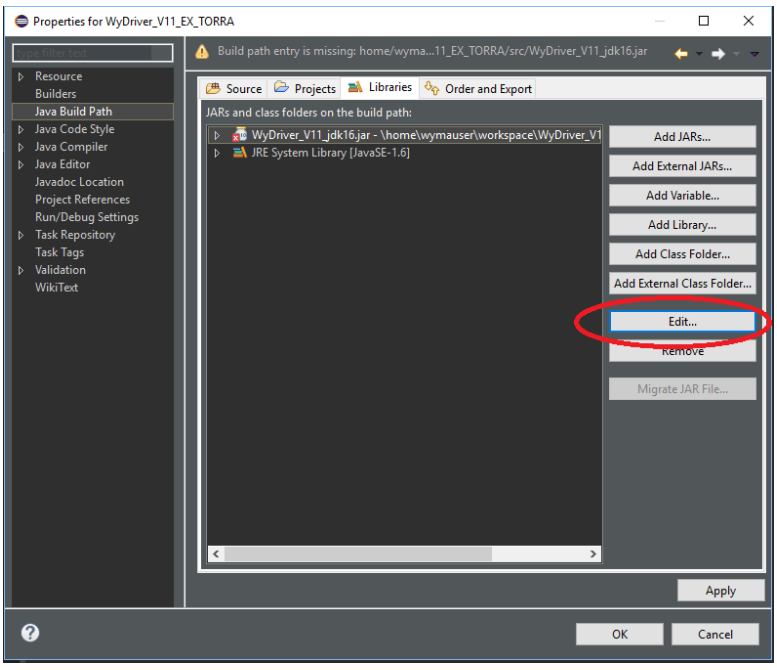
2 – Clique no menu em **“File”** e em seguida em **“Import”**. Escolha a opção **“Existing Project sinto Workspace”**. Clique em **“Next >”**..



3 – Selecione a opção **“Select archive file:”**, clique em **“Browse..”** e busque o arquivo da biblioteca **“project.zip”**. Clique em **“Finish”**.



4 – Ajuste o caminho do arquivo de biblioteca dentro do projeto, clique no menu **“Project”** em seguida na opção **“Properties”** , dentro da caixa de diálogo clique em **“Java Build Path”**, localize o arquivo **“WyDriver_V11_jdk16.jar”** , selecione este arquivo e em seguida clique no botão **“Edit”**. Ajuste o caminho deste arquivo, o local correto é dentro da pasta do projeto: **“WyDriver_V11_EX_TORRA\src”**



5 – O projeto estará pronto para execução.

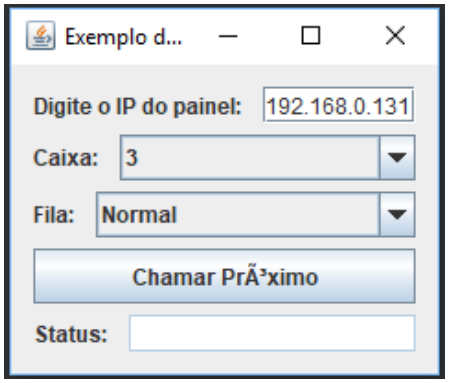
4. Software Aplicativo exemplo

O software aplicativo exemplo em Java que preparamos deve ser usado para testar a chamada do painel e verificar se toda a infraestrutura de rede esta funcionando. O uso do software é bem simples:

1º Execute o aplicativo no Eclipse, clicando no menu **”Run”** e em seguida no item **“Run”**.

2º Ajuste o campo IP com o endereço IP do painel, preencha também qual o número do caixa e o tipo de fila.

3º Com o botão de chamada você chama o cliente mostrando o número no painel.



5. Biblioteca WyDriver

Classe: Socket

Responsável por enviar o pacote de dados para o painel.

Métodos:

Nome:
Socket

Descrição:
Método construtor.

Nome:
Send

Descrição:
Envia o pacote para o painel mostrar o próximo número de caixa.

Cabeçalho:
bool Send(String IP, Packet pack)

Parâmetros:

IP – Informa o IP do painel para onde será enviado o pacote.
pack – Informa o objeto da classe Packet.

Nome:
getStatus

Descrição:

Coleta o resultado da comunicação com o painel.

Cabeçalho:

bool getStatus()

Classe: Packet

Responsável por montar o pacote de dados.

Métodos:

Nome:
Packet

Descrição:

Construtor da classe recebe todas as informações necessárias para montar o pacote de dados.

Cabeçalho:

Packet(byte rClass, byte rId, char cmdType, byte cmdGroup, byte cmdNum, int cmdIdx, int lenData, byte[] data)

Parâmetros:

rClass – Número da classe remota (255)
rId – Número do Id remoto (255)
cmdType – Tipo de requisição ('S')
cmdGroup – Grupo de comandos (51)
cmdNum – Número do comando (1)
cmdIdx – Index do comando (1)
lenData – Comprimento dos dados (12)
data – Dados (TIPO, 0, CAIXA, 1 , 0 , 0 , 0 , 0 , 0 , 2 , 0)

Nome:
getPacket

Descrição:

Recupera o pacote de dados montado.

Cabeçalho:

Byte[] getPacket()

6. Código exemplo

Faremos a análise do código abrindo o Eclipse e expandindo o projeto **“WyDriver_V11_EX_TORRA”** em **“Project Explorer”**.

Localize o arquivo **“Send.java”**, vamos analisar como é feito o envio dos valores para o display.

1) ***Socket sock = new Socket ();***
   Packet pack;
   No código acima criamos dois objetos um chamado “sock” da classe “Socket” e outro chamado “pack” da classe “Packet”.
   a. O objeto pack é responsável pela **preparação** do pacote de dados em um formato que o painel entende.
   b. O objeto sock é responsável pelo **envio** do pacote de dados para o painel através do IP informado.
2) **pack = new Packet((byte) 255, (byte) 255, 'S', (byte) 51, (byte) 1, 1, 12, new byte[]{(byte)(*LINE*.getSelectedIndex()+1),0,(byte)(*BOX*.getSelectedIndex()+1),1, 0,0,0,0,0,0,2,0 });**
   No código acima fornecemos todas as informações para montar o pacote de dados para o painel, neste tipo de painel dedicado somente utilizamos os parâmetros em vermelho, respectivamente são o número da linha e o número do caixa, os demais são valores fixos não devem ser alterados.
3) **sock.Send(IP.getText(), pack.getPacket());**
   No código acima utilizamos o método Send da classe Socket para enviar o pacote para o painel, passando como parâmetro o IP do painel e o pacote montado anteriormente.
4) **if (sock.getStatus()==Const.FRM_OK)**
   No código acima obtemos a resposta da transmissão lendo um atributo disponível na classe Socket.