Universidad de Deusto

Máster en Automatización, Electronica y Control Industrial

Profesor:

Julio Ocejo

Asignatura:

Redes de Comunicación Industrial

Aluno:

Alcides Fernandes

Manual:

Desarrollo de una aplicación de comunicación mediante Modbus/TCP

RESUME:

Este manual contiene información sobre una aplicación cliente modbus TCP / IP desarrollada en C # WPF, la aplicación permite al cliente enviar solicitudes de lectura y escribir datos a diferentes servidores modbus, pero la prueba se implementó solo para solicitudes con 5 funciones (1,2, 3, 4 y 16).

A modo de ejemplo práctico, se agregaron a la aplicación dos Simuladores para robot UR3 (copellia y otro desarrollado desde cero) las dos aplicaciones hacen lo mismo, mueven las articulaciones del robot a través de las respuestas de valores enviados por el modbus UR3 del robot servidor instalado en el laboratorio 006. Otro añadido es el simulador de robot KUKA KRC6, aunque no utilicé el protocolo modbus, lo implementé por el simple hecho de que el protocolo Modbus tiene casi la misma característica en cuanto a comunicación (preguntas y respuestas).

Modo Usuário

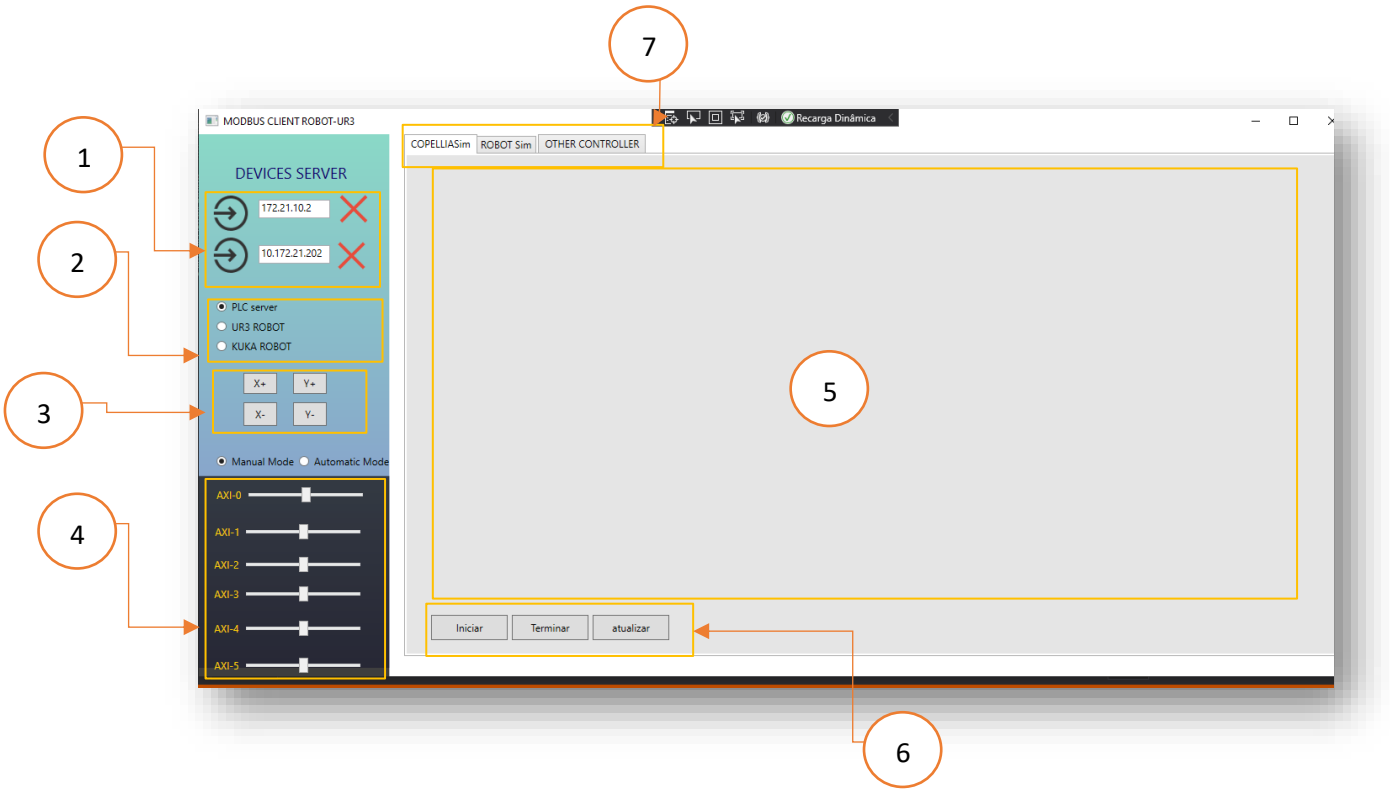

Figura 1_Pantalha inicial

1-Conenexión Server

Contiene los campo para agregar los IP del servidor al que pretende conectarse, botónes para conectarse y desconectarse (inicialmente oculto) y un icono que indica el estado de la conexión. Para robot UR3 e otros PLC se conectam pela puerta 502 (Modbus) y para o robot Kuka se conecta en la puerta 7000.

2- Seleción del tipo de simulador

3-Mover diferentes vistas del Modelo 3D;

4-Controlo Manual de las articulaciónes del Robot;

5-Canvas reservado para el simulador Copellia;

6-Botones para inicializar e cierrar la aplicacion Copellia;

7-TabControl de los Simuladores.

OTROS CONTROLADORES O SERVIDORES MODBUS

Es un tab que contén 5 funciones:

1-Leer Salidas Discretas: lo usuario deve anãdir la primeira salida y el numero de salida deseada, nota que la primeira salida no deve ser menor que uno. La respuesta es mostrado en una tabla de datagrid.

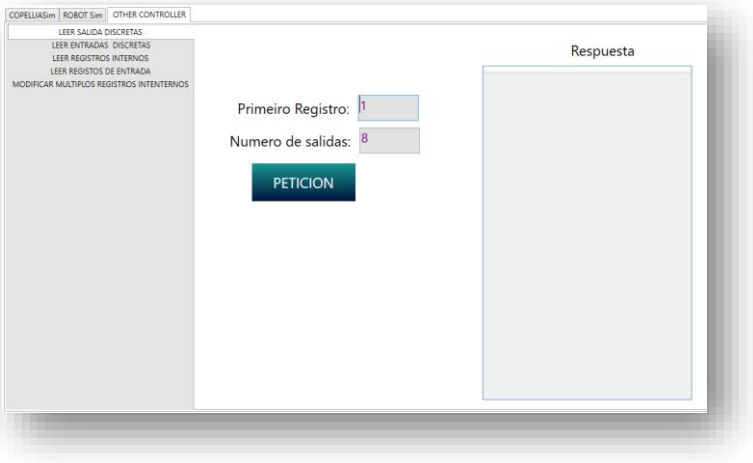

Figura 2-funcione 1

2-Leer Entrada Discretas: lo usuario deve anãdir la primeira entrada y el numero de entrada deseada, nota que la primeira entrada no deve ser menor que uno. La respuesta es mostrado en una tabla de datagrid.

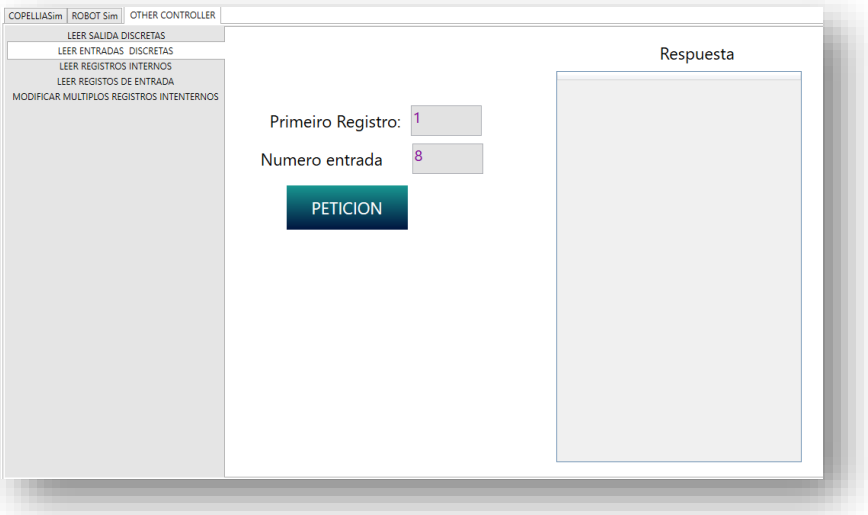

Figura 3-funcione 2

2-Leer Registros internos: igual a los casos anteriores.

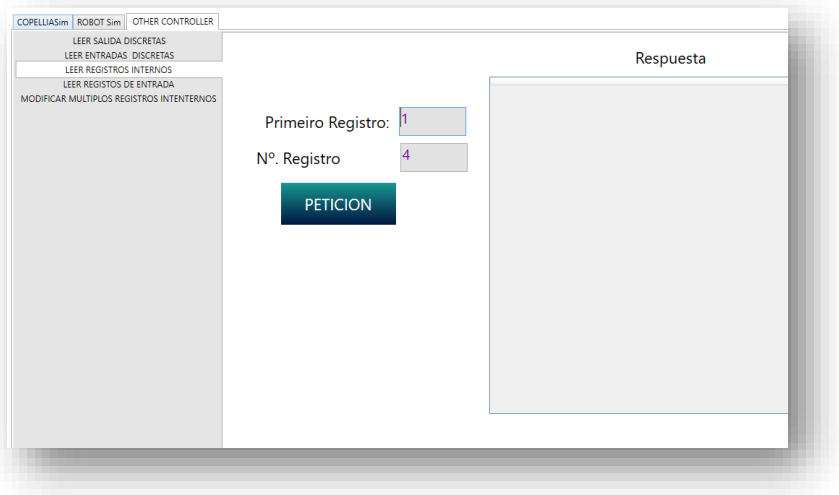

Figura 4-Funcione 3

2-Leer Registros de Entrada: igual a los casos anteriores.

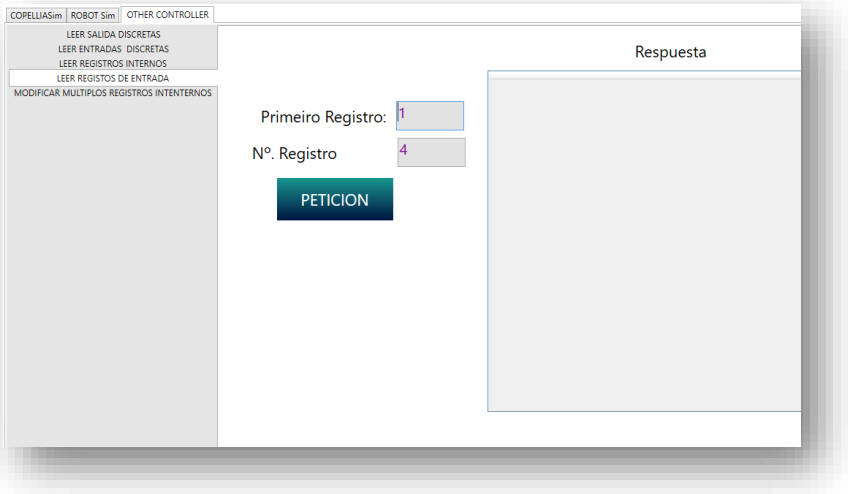

Figura 5 Funcione 4

2-Modificar Multiplos Registros: diferente de los casos anteriores, el usuario deve aãdir el primero registro y el numero de registro que si desea modificar, depues añadir en el Datagrid los valor de cada registo, la resuesta será el primero registro y el numero de registro modificado.

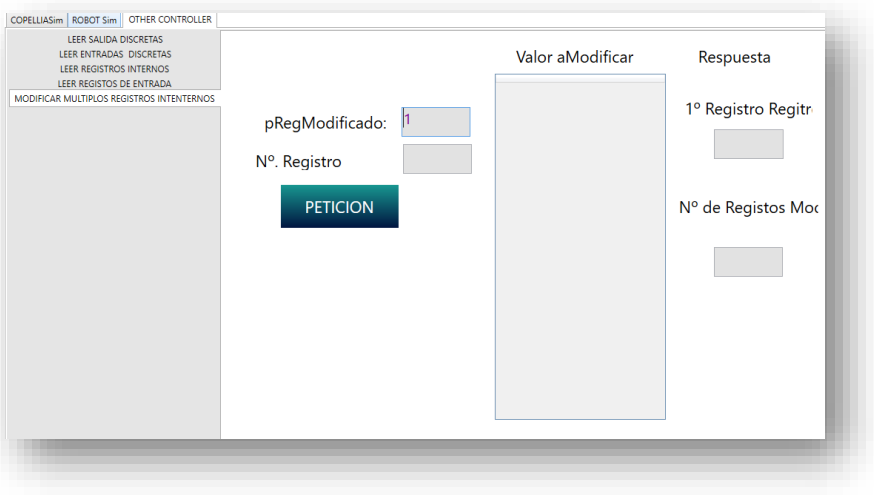

Figura 6 Funcione 16

Simulador Copellia

El boton Iniciar, Terminar y Atualizar permite iniciar la aplicacion copelia e cargar en la ventana de la aplicacion Modbus o cierrar la aplicación.

El simiador es una ciena com dos robot UR3 que pueden moverse de acuerdo con valores que salen de la aplicacion modbus o sea, los Slidbars (AXI0, AXI1…AXI5) defien las posiciones agolares de cada articulacione del robot.

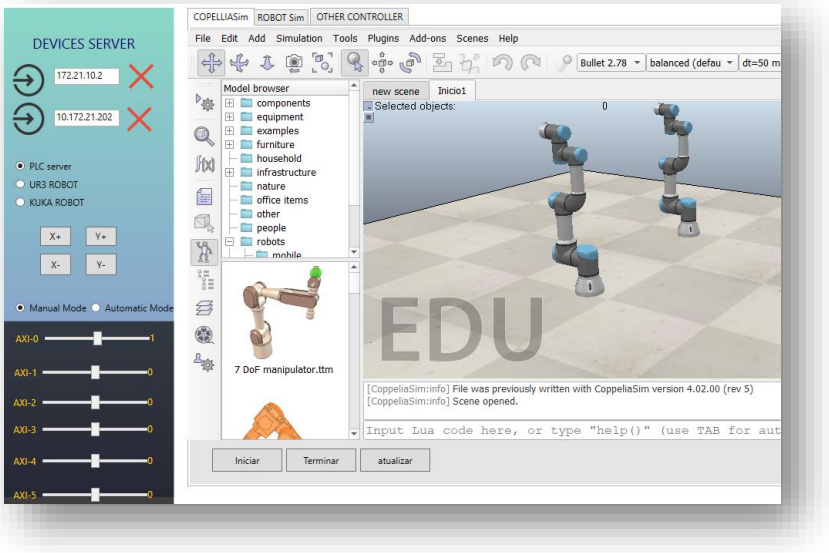

Figura 7-Copellia Sim

Robot Sim

Es un simulador de dos modelos 3D (robot UR3 y Kuka KR6) creado en WPF, con el mismo principio de funcionamiento de copellia.

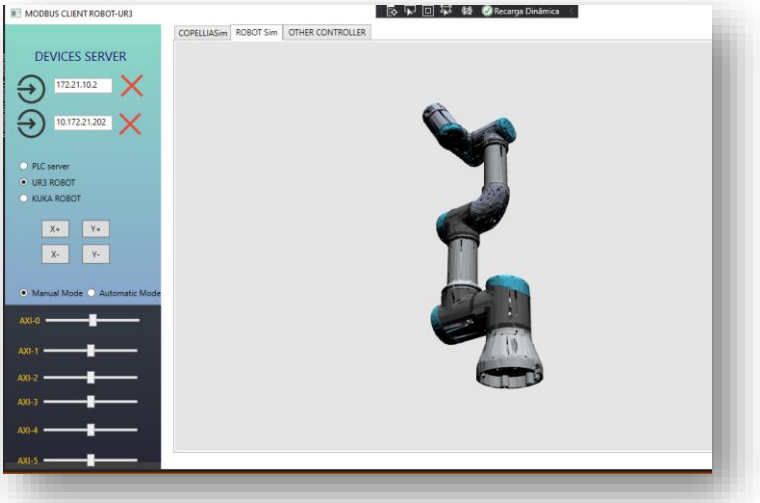

Figura 8 Modelo 3D UR3

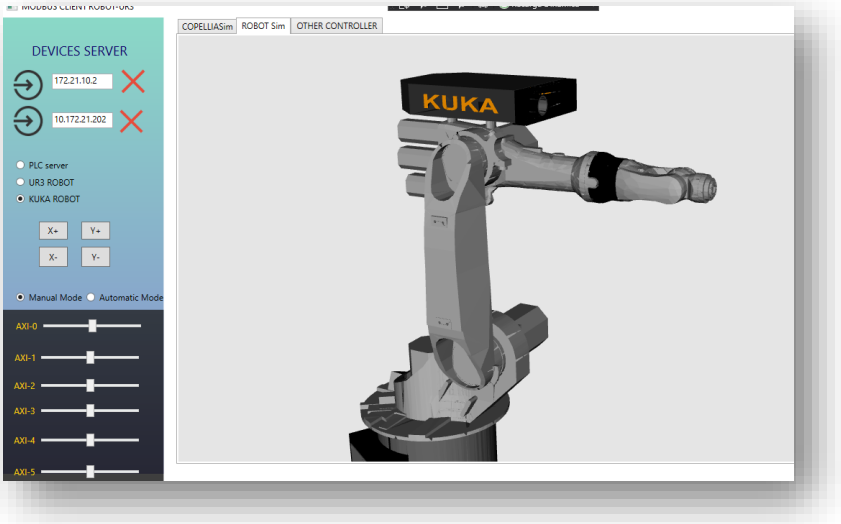

Figura 9 Modelo 3D KR6

Modo Programador

En primero estan las definiciones de libralias, fue añadido el `using` `CoppeliaLib` que tiene los médodos para inciar, cerrar y enviar valores para los robot del coppelia.

`using` `System.Timers` y `using` `System.Windows.Threading`: Para execucion continuo de las peticiones;

```
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;
using System.Collections;
using CoppeliaLib;
using System.Timers;
using System.Windows.Threading;
using System.Globalization;
```

Trama de Peticion de Datos

Fue criando una clase para peticiones de datos, la clase contiene dos método:

1- **Peticion**: com argumiento de numero Mensaje, el Codigo de funcione, el primero registro e el numero de registro que se desea, método retorna un array de bytes que estilzado en a clase Cliente para envio de datos. El método es sencillos, consiste en la construcion de una array de byte de acuedo con la trama de modbus.
2- **Peticion_Multiple:** igual a caso anterior la difereza está en la array que retorna el método, siendo que a anterior retuorna siempre una array de 12 bytes y este retorna el numero de acuerdo con el numedo de registro que se quire modificar.

Simple Petition

La peticiones son hechas por click de un boton de acuerdo no la funcione deseada, un ejemplo esta en la linea 234 – 283.

Primero verificase si cliente está conectado, convertese los campos textos primero e ultimo registro en ushort, declarase una array de bytes_datosque recibe el metodo peticion de la clase Peticion_datos y de segida es enviado la trama por método enviaDatos de la clase Cliente.

```
bytes_datos =(byte[]) peticionRegistro.PETICION(nMensaje, 1,
Convert.ToInt16(primeiro_registro.Text), Convert.ToInt16(Ultimo_Registro.Text));

int res=clie.enviaDatos(bytes_datos,12);
```

se tudo estar bien entonces el método retorna el numero de byte enviado.

Para recibir datos utilizase el método clie.recebeDatos(bytes_respuesta, bytes_respuesta.lenght) y logo segue la lineas de codigo para filtrar la informacion llhegada e dar su devido tratamiento(poner en un datagrid o lo que quiera).

Simple Continua

Permite hacer peticiones continua en cada intervalo de tiempo defenino en la clase timer, utilizase un evento `OnTimedEvent(Object source, ElapsedEventArgs e)` para refrescar los datos de envio e recepciones. Como este timer utiliza utiliza hilo secundário entonces fue necesario añadir un `DispatcherTimer` que correr en el hilo principal para enviar los datos en los objeto da clase principal.

NOTA: la aplicación para los robots utiliza tramas fijas o sea solo para solicitudes de los angles atual delos robots, si desea más cosas basta modificar el primero y el numero de registro que se quiere, pero

sencillohay que tener cuidado si el robot es el kuka porque la trama de peticione cambia casi tudo y la respuesta tiene tamaño diferente.