Capítulo 6

Conexión por OPC

Se ha dedicado este capítulo a la conexión por OPC (*OLE for Process Control*) debido a su gran importancia dentro de este proyecto. OPC nos permite que programas muy diferentes entre sí compartan datos sin ningún tipo de restricción. En este proyecto se ha utilizado para realizar una plataforma en la que se interconectan el software Unity y Matlab que permite aplicar estrategias de control desarrolladas en Unity a modelos de procesos diseñados en Simulink. De esta manera conseguimos total independencia de la planta física, evitando desplazamientos y posibles indisponibilidades de la planta.

La conexión por OPC ya se ha utilizado en otros proyectos para realizar una conexión similar pero en sentido inverso, controlaban desde Matlab la planta. Además también se ha utilizado para diseñar SCADAs en otros programas como Vijeo Citect.

En este capítulo se explicará cómo se ha realizado la conexión y cómo se pasan los datos entre los diferentes programas.

6.1. OLE para Control de Procesos

6.1.1. Motivos para el desarrollo de la especificación OPC.

La arquitectura informática para la industria de proceso incluye los siguientes niveles:

- Gestión de campo: información sobre los dispositivos de instrumentación (estado, constitución, configuración, etc.).
- Gestión de proceso: datos sobre el proceso productivo adquiridos y procesados por sistemas SCADA y DCS.
- Gestión de negocio: integración de la información de planta en los sistemas que gestionan los aspectos financieros de la fabricación.

Se trata de que en la industria se puedan utilizar herramientas estándar (paquetes SCADA, bases de datos, hojas de cálculo) para construir un sistema que recoja sus necesidades de mejora de la productividad.

Para ello es necesario desarrollar una arquitectura de comunicaciones abierta y efectiva que se centre en el acceso a los datos, no en los tipos de datos.

Hay muchas aplicaciones cliente que requieren datos de dispositivos y acceden a ellos desarrollando controladores o drivers de forma independiente. Esto implica:

- Duplicación de esfuerzos: todos los programas necesitan un driver para un determinado hardware.
- Falta de consistencia entre drivers: hay características del hardware no soportadas por todos los drivers.
- Cambios en el hardware: hacen que los drivers queden obsoletos.
- Conflictos de acceso: generalmente, dos programas no pueden acceder simultáneamente al mismo dispositivo puesto que poseen drivers independientes.

Los fabricantes de hardware no pueden desarrollar un driver eficiente utilizable por todos los clientes debido a las diferencias de protocoles entre clientes.

OPC proporciona un mecanismo para extraer datos de una fuente y comunicarlos a cualquier aplicación cliente de manera estándar. Los fabricantes de hardware pueden desarrollar servidores optimizados para recoger datos de sus dispositivos. Dando al servidor un interfaz OPC que permite a cualquier cliente acceder a dichos dispositivos.

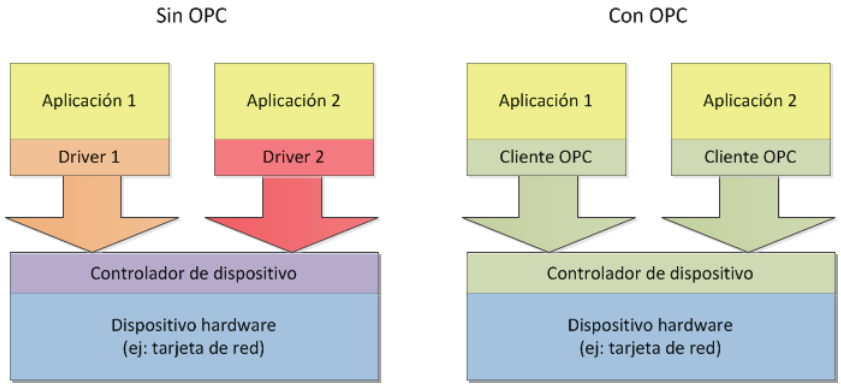

Figura 6.1.1: Diagrama ejemplo aplicaciones sin y con OPC

6.1.2. Origen de la especificación OPC

OPC se basa en la tecnología OLE/COM (Object Linking and Embedding / Component Object Model) de Microsoft.

Esta es la tecnología que permite que componentes de software (escritos en C y C++ por expertos en un sector) sean utilizados por una aplicación (escrita en Delphi o VisualBasic para otro sector).

De esta forma se desarrollaran componentes C y C++ que encapsulen los detalles de acceder a los datos de un dispositivo, de manera que quienes desarrollen aplicaciones empresariales puedan escribir código en VisualBasic que recoja y utilice datos de planta.

El diseño de los interfaces OPC soporta arquitecturas distribuidas en red. El acceso a servidores OPC remotos se hace empleando la tecnología DCOM (Distributed COM) de Microsoft.

6.2. Configuración

Para poder realizar el control de la planta simulada a través del PLC, se necesita una conexión OPC que comunique los programas cliente (Matlab y Unity) con el autómata programable.

Se debe realizar para ello un volcado de aquellas variables de Unity que queramos utilizar en Matlab. El software OFS Factory Server de *Schneider Electric* nos permite agrupar las variables de Unity bajo un nombre de grupo que sera con el que conectemos más tarde desde Matlab una vez identifiquemos el servidor de Schneider.

Antes de nada es necesario configurar el servidor mediante el OFS Configuration Tools.

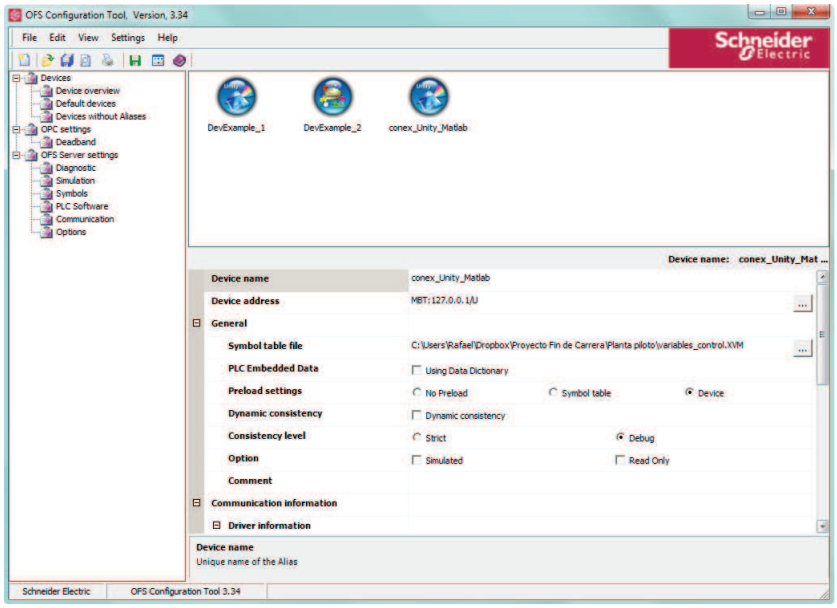

Figura 6.2.1: Pantalla de configuración del OFS Configuration Tool de *Schneider Electric*.

Seleccionamos un nuevo *Device* y lo nombramos como conex_Unity_Matlab. Hay que configurar la IP del autómata, que como ya se indicó variará en función de si es un autómata real o el simulador y seleccionamos el tipo de PLC, en nuestro caso Unity.

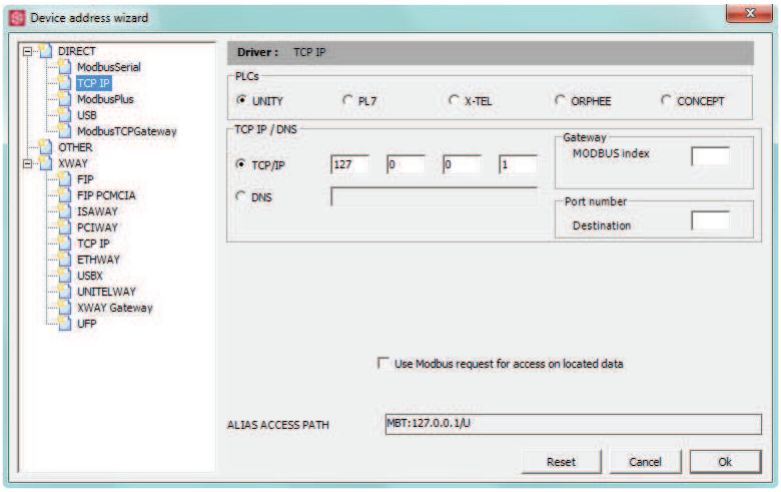

Figura 6.2.2: Pantalla de configuración IP del OFS Configuration Tools.

En *Symbol table* file habrá que poner la ruta hasta el archivo .XVM que contiene las variables exportadas desde Unity. Para exportar estas variables, basta con seleccionar las variables que se desean exportar haciendo clic derecho sobre las variables y seleccionando "Exportación seleccionada". Luego habrá que ponerle un nombre al conjunto de variables exportadas y guardarlas como .XVM. En nuestro caso el nombre de este archivo .XVM es variables_control.XVM.

Por último seleccionamos *Device* en *Preload settings* y *Debug* en *Consistency level* y ya tendremos configurado el servidor.

Con el fichero.m que se muestra en la Figura 6.2.3 llamado conexion_OPC_Matlab.m conseguiremos leer y escribir en las variables del servidor.

```
%Funcion OPC_Matlab%
%creamos el cliente OPC
cliente = opcserverinfo('127.0.0.1');
%buscamos los servidores alcanzables por el PC
allServers = cliente.ServerID';
%seleccionamos el servidor de Schneider asociado al PLC M340
da = opcda('127.0.0.1', 'Schneider-Aut.OFS.2');
%conectamos
connect(da)
%creamos un grupo en el que definir las variables Matlab
grupo0 = addgroup(da);
%Añadimos las variables Matlab al grupo previamente definido,
%cargando en ellas las variables Unity

VARIABLE_A_CONTROLAR_M = additem(grupo0,'conex_Unity_Matlab!VARIABLE_A_CONTROLAR');
REFERENCIA_M = additem(grupo0,'conex_Unity_Matlab!REFERENCIA');
SALIDA_PID_M = additem(grupo0,'conex_Unity_Matlab!SALIDA_PID');
PARO_SIMULACION_M = additem(grupo0,'conex_Unity_Matlab!PARO_SIMULACION');
ERROR_PID_M = additem(grupo0,'conex_Unity_Matlab!ERROR_PID');
INI_CONTROL_M = additem(grupo0,'conex_Unity_Matlab!INI_CONTROL');
INI_SIMULACION_M = additem(grupo0,'conex_Unity_Matlab!INI_SIMULACION');
SALIDA_FUZZY_M = additem(grupo0,'conex_Unity_Matlab!SALIDA_FUZZY');
```

Figura 6.2.3: Código en Matlab para establecer la conexión por OPC.

Mediante los comandos READ y WRITE podemos leer y escribir en las variables exportadas desde Unity a través del servidor de OFS. De este modo podremos leer el valor de la temperatura del reactor y escribir en la señal de actuación el valor que obtenga el controlador de Unity.

Una vez ejecutado el fichero.m aparecerá la ventana del OPC Factory Server que nos indicará que la conexión se ha establecido correctamente y que estamos en disposición de poder leer y escribir sobre las variables.

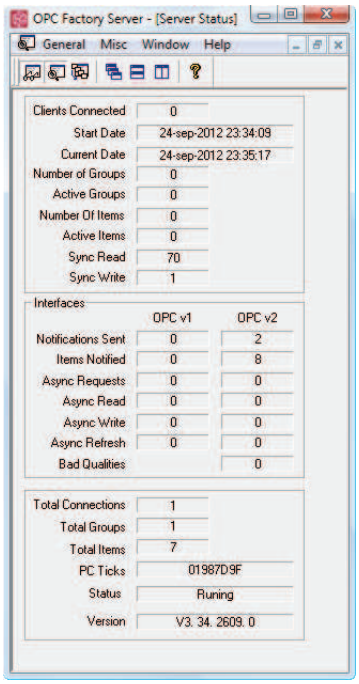

Figura 6.2.4: Ventana del OPC Factory Server.

Durante el transcurso del proyecto se detectaron problemas de incompatibilidad entre la versión V.3.31 y los sistemas operativos Windows Vista y 7. Para solucionar

el problema se optó por conseguir una versión mas moderna la V.3.34 de OFS y el problema desapareció.