DISEÑO DE PRÁCTICAS DE CONTROL VIRTUALES CON MATLAB Y FACTORY I/O

Michelena Grandío, Álvaro[1]; Casteleiro-Roca, José Luis [2]; Jove Pérez, Esteban [3]; Quintián Pardo, Héctor [4]; Zayas-Gato, Francisco [5]; Calvo-Rolle, José Luis[6]

[1] Universidade da Coruña, CITIC, 0000-0003-0134-5660
[2] Universidade da Coruña, CTC, 0000-0001-9740-6477
[3] Universidade da Coruña, CITIC, 0000-0002-0625-359X
[4] Universidade da Coruña, CTC, 0000-0002-0268-7999
[5] Universidade da Coruña, CTC, 0000-0002-0994-1961
[6] Universidade da Coruña, CITIC, 0000-0002-2333-8405

RESUMEN

Debido a la evolución de la educación, y en el contexto actual, se hace necesario realizar una adaptación de la planificación educativa hacia un nuevo escenario. Esta nueva planificación conlleva, entre otros, el gran desafío de la adaptación de las prácticas presenciales de laboratorio de la educación superior para que puedan llegar a todo el alumnado. Una alternativa que se está utilizando es el uso de escenarios virtuales, en lugar de las prácticas de laboratorio presenciales, gracias al uso de aplicaciones software. Para el caso concreto de las asignaturas relacionadas en el ámbito de la automática y el control, estos escenarios se pueden lograr con herramientas como el Factory I/O, que gracias al empleo del programa KEPServerEx, se puede interconectar con MatLab/Simulink. En este documento se expondrán unos scripts (programas de MatLab) que se han creado para poder realizar prácticas de control, con unas características similares a plantas de laboratorio reales de las que se disponen en los laboratorios de las universidades. Además, estos scripts se han confeccionado de manera que el alumno pueda cambiar del entorno virtual al real de una manera sencilla.

PALABRAS CLAVE: Factory I/O, MatLab, Laboratorio virtual, Prácticas de control, Docencia no presencial

CITA RECOMENDADA:

Michelena Grandío, Álvaro; Casteleiro-Roca, José Luis; Jove Pérez, Esteban; Quintián Pardo, Héctor; Zayas-Gato, Francisco; Calvo-Rolle, José Luis (2022): Diseño de prácticas de control virtuales con MatLab y Factory I/O. En García Naya, J.A. (ed.) (2022). *Contextos universitarios transformadores: a innovación como eixo vertebrador da docencia. VI Xornadas de Innovación Docente*. Cufie. Universidade da Coruña. A Coruña (pág. 161-171).

DOI capítulo: https://doi.org/10.17979/spudc.000016.161

DOI libro: https://doi.org/10.17979/spudc.000016

ABSTRACT

Due to the evolution of education, and in the current context, it is necessary to adapt educational planning to a new scenario. This new planning entails, among others, the great challenge of adapting face-to-face laboratory practices in higher education so that they can reach all students. An alternative that is being used is the use of virtual scenarios, instead of face-to-face laboratory practices, thanks to the use of software applications. For the specific case of subjects related to automation and control, these scenarios can be achieved with tools such as Factory I/O, which, thanks to the use of the KEPServerEx program, can be interconnected with MatLab/Simulink. This document will expose some scripts (MatLab programs) that have been created to be able to carry out control practices, with similar characteristics to real laboratory plants that are available in University laboratories. In addition, these scripts have been made in such a way that the student can change from the virtual environment to the real one in a simple way.

KEY WORDS: Factory I/O, MatLab, Virtual laboratory, Control practices, Non-face-to-face teaching

1. INTRODUCCIÓN

En trabajos previos de los autores como los que se pueden ver en [1-3], se pone de relieve que las materias con peor rendimiento académico en las titulaciones universitarias son aquellas en las que el número de sesiones prácticas es escaso, o susceptible de ser incrementado significativamente. Es por ello que una posible vía de actuación para tratar de mejorar el porcentaje de aprobados en los estudiantes aborde el aumento del número de sesiones prácticas. Estas sesiones fortalecen los conocimientos adquiridos durante las sesiones teóricas, tratando así de mejorar el rendimiento de los estudiantes.

Siguiendo esta vía de actuación, las escuelas de ingeniería han avanzado en el diseño de nuevas plantas de laboratorio, con la idea de ampliar la cantidad de puestos disponibles y permitir a los alumnos realizar prácticas con gran variedad de sistemas [4]. Sin embargo, hoy en día, y gracias a aplicaciones informáticas, se ha conseguido obtener mejores resultados utilizando plantas de laboratorio virtuales [5-7]. En este nuevo planteamiento, desaparece tanto la limitación de acceso a las instalaciones físicas, como la limitación horaria, además de que el número y tipo de plantas de laboratorio disponible puede incrementarse notablemente.

En este trabajo se van a explicar los scripts desarrollados para interactuar desde MatLab con una planta virtual de laboratorio de control de nivel. El programa MatLab, en este caso, será el que realice un lazo de control para regular el nivel deseado en el depósito virtual. El programa Factory I/O será el que cree el escenario virtual que contendrá la planta de control de nivel. Por último, el KEPServerEX 6 será el que se encargue de realizar la interconexión de uno a otro a través de la configuración de un servidor OPC Virtual.

Un servidor OPC, *OLE (Object Linking and Embedding) for Process Control*, es un dispositivo que se encarga de recopilar datos de dispositivos reales y hacer de enlace con diferentes programas (entre otras cosas). Los programas software trabajan comunicándose con el servidor OPC, y éste es el que se encarga de enviar y recibir los datos con los equipos hardware reales. En el caso aquí contemplado, no existen dispositivos reales, ya que se trata de una planta virtual de laboratorio, pero el principio de funcionamiento sería el mismo para el control de una planta real a través de un servidor OPC (Figura 1).

La configuración de los tres programas para poder realizar la correcta comunicación entre ellos se puede encontrar en [8], donde los autores describen las características fundamentales y las configuraciones necesarias de los tres programas.

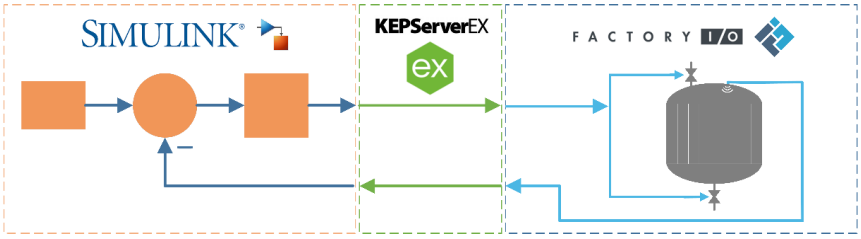

Figura 1. Esquema de interconexión

Después de esta introducción se describen los scripts implementados, donde se hace una pequeña introducción a la topología de control utilizada por los autores. Finalmente se presentarán las conclusiones del trabajo.

2. DESCRICIÓN DE LOS SCRIPS

Los scripts que se han desarrollado en este trabajo se basan en una topología de control previa desarrollada por los autores. Esta topología se puede ver en la Figura 2, donde se muestran las llamadas a los scripts programados en un programa de control.

```
% Ejemplo de sistema de control

% Activación de la tarjeta de adquisición de datos
DAQ_Start

% Definición de las variables básicas
...

% Bucle de control
for i=1:n
    ...

    % Actualización de los valores en la planta
    DAQ_Write(U,Valvula_vaciado);
    PV=DAQ_Read;

    % Cálculo de la señal de control según el controlador utilizado
    ...

    % Temporización del bucle
    ...
end

% Desactivación de la tarjeta de adquisición de datos
DAQ_Stop

...
```

Figura 2. Topología de un programa de control

Inicialmente, antes de entrar en el bucle de control propiamente dicho, se activaría la tarjeta de adquisición de datos con la script *DAQ_Start*. Al finalizar el programa de control, el script utilizado para desactivar la tarjeta de adquisición de datos sería el *DAQ_Stop*. Dentro del bucle de control se pueden ver los otros dos scripts programados: el *DAQ_Write*, para enviar datos a la tarjeta de adquisición de datos y actualizar el valor de sus salidas; y el *DAQ_Read*, para recibir datos de la tarjeta de adquisición y poder conocer lo valores de sus entradas.

En la Figura 3 se puede ver, a modo de ejemplo, la estructura de los scripts programados; en este caso se trata del *DAQ_Start*, pero la estructura de programación es la misma para los 4 scripts. Se muestra esta estructura para enfatizar el hecho de que la programación del bucle de control es independiente del tipo de sistema que se esté controlando. En este caso, el script incluye el código necesario tanto para comunicarse con un servidor OPC simulado (es el caso en el que se centra este trabajo), como para el uso de tarjetas de adquisición de National Instruments, tarjetas de adquisición basadas en Arduino, o incluso para realizar pruebas con sistemas simulados mediante su función de transferencia.

```
1   function []=DAQ_Start()
2   % This code is used when you want to use an OPC Simulator
3   %{ ...
32
33  % This code is used when you want to use a real NI_DAQ
34  %{ ...
56
57  % This code is used when you want to use a real Arduino_DAQ
58  %{ ...
71
72  % This code is used when you want to simulate a system
73  %{ ...
103 end
```

Figura 3. Ejemplo de estructura interna de un script DAQ_*

En cada aplicación, el alumno deberá utilizar el código correspondiente, descomentando la sección de código que desea ejecutar. De esta manera, por ejemplo, los alumnos podrían trabajar inicialmente con sistemas simulados mediante funciones de transferencia, posteriormente pasar a plantas simuladas a través de un OPC, y finalmente probar sus programas de control en las plantas reales del laboratorio. En los tres supuestos el programa

de control es similar, y lo único que tendrían que cambiar es la parte del código correspondientes en los scripts de control.

2.1 DAQ_START

Para la comunicación entre los diferentes scripts, se utilizan variables globales, de manera que lo primero que se hace en los scripts es declarar esas variables globales (Figura 4, líneas 3-5). En el caso concreto del *DAQ_Start*, se puede ver como en la línea 9 se crea el cliente OPC, que en este caso está en el propio ordenador al tratarse de un servidor OPC simulado. Después, en la línea 15, se realiza la conexión con el servidor, y en la línea 20 se añade el grupo correspondiente que contiene los ítems de las señales que se van a usar. En las líneas 23-25 de la Figura 4 se puede ver como se asocian los valores de los ítems del servidor OPC con las variables globales declaras al inicio del script. Finalmente, en la línea 29, se cierra la válvula de descarga para iniciar la práctica correspondiente siempre desde el mismo estado.

```
1  function []=DAQ_Start()
2  % This code is used when you want to use an OPC Simulator
3  global Level
4  global Charge
5  global Discharge
6
7  % Create an OPC DA client for a local server (previously create in
8  % KEPServerEX6
9  OPC_Simulator=opcda('localhost','Kepware.KEPServerEx.V6');
10
11 % Connect to de OPC virtual server. If this instruction fails ('No register
12 % class' menssage appear) it is neccesary to run the 'opcregister'
13 % instruction. This should be happen only the first time that OPC
14 % instructions are used.
15 connect(OPC_Simulator);
16
17 % The rest of the configuration is especific to the used that the OPC has
18 % It is neccesary to create a group to identify the items that are
19 % connected to the OPC
20 Level_Control=addgroup(OPC_Simulator,'Factory.OPC');
21 % Then, the items are create throw the variables that will we used in
22 % DAQ_Read and DAQ_Write scripts
23 Level=additem(Level_Control,'Factory.OPC.Sim_Level');
24 Charge=additem(Level_Control,'Factory.OPC.Sim_Charge');
25 Discharge=additem(Level_Control,'Factory.OPC.Sim_DisCharge');
26
27 % The Discharge valve will be closed previously, as the DAQ_Stop put it in
28 % a open state to empty the plant
29 write(Discharge,0);
```

Figura 4. Código del script DAQ_Start

2.2 DAQ_STOP

Para el caso del script *DAQ_Stop*, lo que se hace es resetear las variables para que la válvula de llenado del depósito quede cerrada y la de descarga abierta (Figura 5, líneas 9 y 10); inicialmente, como se mencionó en el aparatado anterior, se tienen que declarar las variables globales para que todos los scripts trabajes con los mismos datos (líneas 3-5). En el caso de este script, lo único que se hace (en referencia al servidor OPC) es resetearlo para eliminar el objeto creado al ejecutar el script *DAQ_Start* (línea 13). Finalmente, se eliminan las variables globales para liberar la memoria utilizada (líneas 16-18).

```
1  function [] = DAQ_Stop()
2  % This code is used when you want to use an OPC Simulator
3  global Level
4  global Charge
5  global Discharge
6
7  % Firstly it is neccesary to reset the output values (the Discharge valve
8  % will be in a open state to empty the plant)
9  write(Charge,0);
10 write(Discharge,10);
11
12 % Then disconnect the OPC system and delete all OPC objects
13 opcreset
14
15 % The variables of the OPC is deleted
16 clear Level
17 clear Charge
18 clear Discharge
```

Figura 5. Código del script DAQ_Stop

2.3 DAQ_WRITE

En el caso concreto de este script, lo primero será comprobar que los valores que se desea enviar al servidor OPC están en el rango correspondiente. En este caso, hay dos variables involucradas, que el estudiante incluye en la llamada a la función (Figura 6, línea 1) con el nombre *Output_data_0* y *Output_data_1*. La comprobación de que las variables están en rango se hace en las líneas 7-11 y 12-16 respectivamente, y en este caso se asegura que los valores que se enviarán al OPC están en el rango de 0 a 100, que es el más ampliamente utilizado al poderse referir a las variables como porcentajes (porcentaje de apertura, de

luminosidad…). Además, debido a que el OPC trabaja con rangos de 0 a 10, una vez asegurado el rango porcentual, se divide entre 10 cada una de las variables (líneas 19 y 20). Al igual que en los scripts explicados en los apartados previos, las líneas 3 y 4 de la Figura 6 muestra como se declaran las variables globales que se utilizan en las líneas 23 y 24 para actualizar el valor de la válvula de llenado y de vaciado respectivamente.

```
1  function []=DAQ_Write(Output_data_0,Output_data_1)
2  % This code is used when you want to use an OPC Simulator
3  global Charge
4  global Discharge
5
6  % Ensure that the value is inside the operation range
7  if Output_data_0>100
8      Output_data_0=100;
9  elseif Output_data_0<0
10     Output_data_0=0;
11 end
12 if Output_data_1>100
13     Output_data_1=100;
14 elseif Output_data_1<0
15     Output_data_1=0;
16 end
17
18 % As the range of the values is 0@100, the values are scaled to 0@10
19 Output_data_0=Output_data_0/10;
20 Output_data_1=Output_data_1/10;
21
22 % Once the values are in the correct scale, they are "sent" to the OPC
23 write(Charge,Output_data_0);
24 write(Discharge,Output_data_1);
```

Figura 6. Código del script DAQ_Write

2.4 DAQ_READ

Por último, la Figura 7 muestra el código del script utilizado para leer variables de la planta virtual. En este caso, al igual que en el anterior, hay una comunicación entre el usuario y el script a través de la llamada a la función como se puede ver en la línea 1. En este caso, el valor leído se va a devolver en la variable *Input_data*. Como en casos anteriores, el script se inicia con la declaración de la variable global necesaria (línea 3), y lo único que hace es leer su valor (línea 6). Finalmente, y para seguir el mismo criterio mencionado de que las variables estén en valores porcentuales, el valor leído se multiplica por 10 como se muestra en la línea 8.

```
function [Input_data]=DAQ_Read()
% This code is used when you want to use an OPC Simulator
global Level

% The OPC reading is returned in a struct variable
Input_data=read(Level);
% OPC reading range is 0@10, them the measure is scaled to 0@100
Input_data=10*Input_data.Value;
```

Figura 7. Código del script DAQ_Read

4. CONCLUSIONES

Cabe destacar el papel desempeñado por el software KEPServerEX, encargado de establecer la interconexión entre MatLab y Factory I/O a través de la configuración de un servidor OPC Virtual. Además de su principal funcionalidad, el empleo de este tipo de servidor permite que el usuario interiorice la importancia de este estándar de comunicación, estrechamente relacionado con la supervisión y el control de procesos industriales.

Además, las ventajas que ofrece el empleo de MatLab para el control se han visto acrecentadas gracias al empleo del software Factory I/O. Esta herramienta permite realizar una simulación realista del comportamiento de la planta de control de nivel, pudiendo comprobar el funcionamiento del sistema en tiempo real. De esta manera, la experiencia se ha mostrado más atractiva para el usuario y, por ende, más satisfactoria desde un punto de vista didáctico. Por otra parte, el conocimiento de este software en potenciales lectores, podría favorecer su uso en otras disciplinas típicas de las titulaciones de ingeniería, como podría ser la automatización de procesos.

Se puede ver una explicación del funcionamiento de los scripts en el siguiente enlace:
https://web.microsoftstream.com/video/1aad9d55-d2b0-4def-934d-10fe907c68eb

5. REFERENCIAS

López Vázquez, J. A. & Casteleiro Roca, J. L. & Jove, E. & Zayas Gato, F. & Quintián, H. & Calvo-Rolle, J. L. (2021). Data Collection Description for Evaluation and Analysis of Engineering Students Academic Performance. En *The 11th International Conference on EUropean Transnational Educational (ICEUTE 2020)* (pp. 317-328). Cham. Springer International Publishing

García Ordás, M. T. & López Vázquez, J. A. & Alaiz Moretón, H. & Casteleiro Roca, J. L. & del Blanco, D. Y. M. & Casado Vara, R. & Calvo Rolle, J. L. (2021). Comparative of Clustering Techniques for Academic Advice and Performance Measurement. En *The 11th International Conference on EUropean Transnational Educational (ICEUTE 2020)* (pp. 215-226). Cham. Springer International Publishing

López Vázquez, J. A. & Arce, E. & Fernández Ibáñez, M. I. & Casteleiro Roca, J. L. & Zayas, F. & Suárez García, A. (2022). Longitudinal Study of Grades for the Industrial Electronics and Automation Engineering Degree Programme. En *14th International Conference on Computational Intelligence in Security for Information Systems and 12th International Conference on European Transnational Educational (CISIS 2021 and ICEUTE 2021)* (pp. 295-304). Cham. Springer International Publishing

Michelena, Á. & Zayas Gato, F. & Jove, E. & Casteleiro Roca, J. L. & Quintián, H. & Calvo Rolle, J. L. (2022). Low Cost Three-Phase Motor Speed Control System Design for Educational Laboratory Practices. En *14th International Conference on Computational Intelligence in Security for Information Systems and 12th International Conference on European Transnational Educational (CISIS 2021 and ICEUTE 2021)* (pp. 315-324). Cham. Springer International Publishing

Michelena, Á. & Zayas Gato, F. & Jove, E. & Casteleiro Roca, J. L. & Arce Fariña, E. & Quintián, H. & Calvo Rolle, J. L. (2021). Sistema de prácticas virtuales como alternativa al laboratorio presencial en asignaturas de Ingeniería de Control. En *Xornadas de*

Innovación Docente (5º. 2021. A Coruña) (pp. 249-259). A Coruña. Universidade da Coruña, Servizo de Publicacións

Michelena, Á. & Zayas Gato, F. & Jove, E. & Casteleiro Roca, J. L. & Quintián, H. & Calvo Rolle, J. L. (2021). Implementación virtual de prácticas de asignaturas de control como alternativa a las prácticas de laboratorio presenciales. En *XLII Jornadas de Automática* (pp. 259-268). Castellón. Universidade da Coruña, Servizo de Publicacións

Zayas Gato, F. & Michelena, Á. & Jove, E. & Casteleiro Roca, J. L. & Quintián, H. & Arce Fariña, E. & Calvo Rolle, J. L. (2021). Virtual Implementation of Practical Control Subjects as an Alternative to Face-to-Face Laboratory Lessons. En *Computational Intelligence in Security for Information Systems Conference* (pp. 254-263). Cham. Springer International Publishing

Michelena, Á. & Casteleiro Roca, J. L. & Jove, E. & Zayas Gato, F. & Quintián, H. & Calvo Rolle, J. L. (2022). Creación de laboratorios virtuales para asignaturas de control con Factory I/O® y Simulink®. A Coruña, Universidade da Coruña, Servizo de Publicacións