DESARROLLO DE SOFTWARE DE CONTROL PARA EL KIT EDUCATIVO "COMPACT WORKSTATION" DE FESTO.

V. RODRIGUEZ [1], F. MUR [2], M. CASTRO[2] Y E. SAN CRISTOBAL[2]

[1]*Departamento de Física y Química. Instituto de Enseñanza Secundaria "La Vaguada".*
Zamora. España.
vrm@mac.com
[2]*Departamento de Ingeniería Eléctrica, Electrónica y de Control.*
Escuela Técnica Superior de Ingeniería Industrial. Universidad Nacional de Educación a Distancia.
España.
Fmur@ieec.uned.es
mcastro@ieec.uned.es
elio@ieec.uned.es

En este trabajo se describe el software desarrollado para el control de una planta piloto hidráulica con la que se pretende realizar prácticas de la asignatura de Regulación Automática I, que se imparte actualmente en la UNED. Dicho software se caracteriza por: la facilidad para configurar cualquier sistema de control y por la posibilidad de la realización de prácticas presenciales y a distancia. Adicionalmente se desarrolla un simulador. En todos los casos se ha utilizado el lenguaje G de National Instruments.

Palabras clave: Regulación Automática, LabView, Compact Workstation, Web Server.

1. Objetivo

El presente trabajo tiene por objetivo el desarrollo de software que permita la realización de prácticas utilizando el kit educativo "Compact Workstation"[1] de Festo. La idea inicial era disponer de una plataforma fácilmente configurable que permitiera la realización de diferentes prácticas. Posteriormente añadieron al objetivo inicial el desarrollo de un simulador y la posibilidad de la realización de los experimentos a distancia.

2. Introducción

El kit "Compact Workstation" de Festo es una pequeña planta hidráulica, con sensores, actuadores y una tarjeta de adquisición de datos que permite comunicarse con un ordenador a través del puerto serie. El kit tiene muchas posibilidades para desarrollar lazos de control de: nivel, caudal, presión o temperatura, así como programas SCADA. Sin embargo, la intención de desarrollar un sistema para la realización de las prácticas a distancia obligó a la modificación de la planta, de forma que permitiera su total control a través de internet.

El sistema contiene cuatro equipos: la planta piloto, la unidad de adquisición de datos, el ordenador del laboratorio y los ordenadores remotos (fig. 1). El ordenador del laboratorio se comunica con la unidad por el puerto serie usando los protocolos RS-232 y EasyPort [2,3]. Este último es específico de la unidad de adquisición de datos. Así mismo este ordenador actúa como servidor de datos mediante el uso de Web

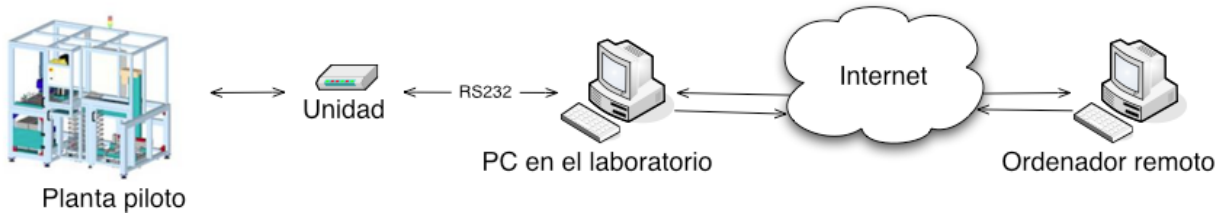

Figura 1: Esquema simplificado del sistema.

Server integrado en LabView. Esto permite visualizar y manipular el panel frontal del programa que corre en el ordenador del laboratorio.

El desarrollo de este trabajo ha requerido del estudio del protocolo Easy-Port del que se encuentra muy escasa información, del desarrollo de un programa controlador, de un simulador y del sistema de control a distancia.

3. La planta

3.1. Cambios

La planta original presentaba dos problemas: la existencia de un circuito neumático y de algunas válvulas manuales. El circuito neumático no puede ser utilizado ya que el laboratorio no dispone de aire a presión. Igualmente la presencia de algunas válvulas manuales en ciertos lugares impedía la modificación de la planta a distancia.

El circuito neumático se eliminó y se añadieron cuatro válvulas eléctricas todo/nada. Se transformó el circuito de forma que se pudiese hacer circular agua desde el depósito inferior al mismo, al superior, a ambos sitios o al depósito de acero. También que vía de salida del depósito superior al inferior pudiera ser cerrada, abierta o controlada por una válvula proporcional. La planta en la disposición final (Fig. 2 y 3) puede configurarse de 7 formas diferentes, activando o desactivando las válvulas todo/nada añadidas. Se mantuvieron algunas válvulas manuales para poder vaciar los depósitos sin tener que arrancar el controlador.

3.2. Sensores y actuadores

La planta dispone (Fig. 3) de cinco sensores digitales: un sensor de flujo a la salida de la bomba (FS+101.1), cuatro sensores de nivel (LS-101.1, LS-101.2, LS+101.3 y LA+101.4). Cuatro sensores analógicos: nivel de líquido en el tanque superior (LIC102.1), caudal a la salida de la bomba (FIC101.1), presión en el interior del depósito de acero (PIC103.1), temperatura del tanque inferior (TIC101,1). Ocho actuadores digitales: calefactor (E101), selector de la forma de control de la bomba (analógico o digital), control de la bomba (P101) si se encuentra en modo digital, activación de la válvula proporcional (V105) y cuatro válvulas todo o nada (V101, V102, V103 y V104). Y de dos actuadores analógicos, una válvula proporcional (V105) y control de la bomba (P101) si se encuentra en modo analógico.

4. Protocolo EasyPort

EasyPort es un protocolo basado en el envío y recepción de tramas. Todas ellas se escriben en código ASCII, generalmente en mayúsculas. Terminan en el carácter ASCII número 13 (D_H), que es el retorno de carro (CR). Es de tipo cliente servidor. En el que el PC local actúa como cliente mientras que la unidad funciona como servidor. Las tramas pueden ser de tres clases de configuración, de envío de datos o de petición de datos. Cuando la unidad recibe una trama contesta. Si es de petición de datos contesta incluyendo en la contestación el valor de la magnitud. Si la trama es de envío de datos contesta

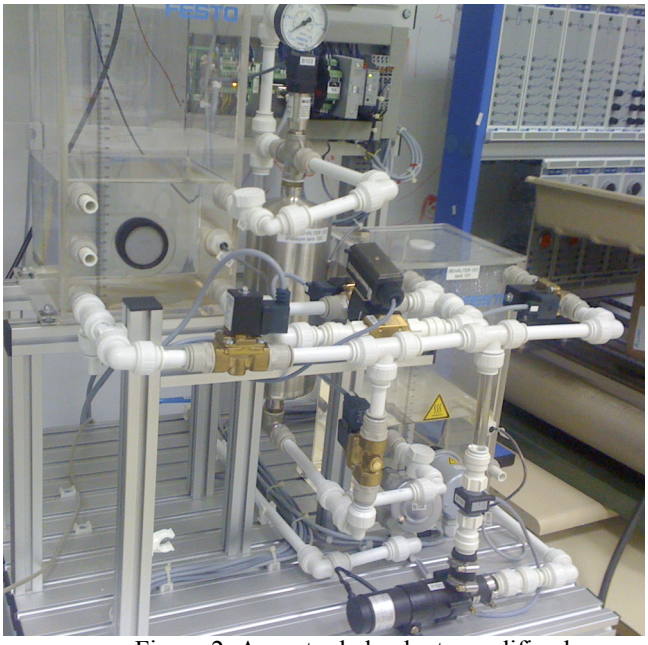

Figura 2: Aspecto de la planta modificada.

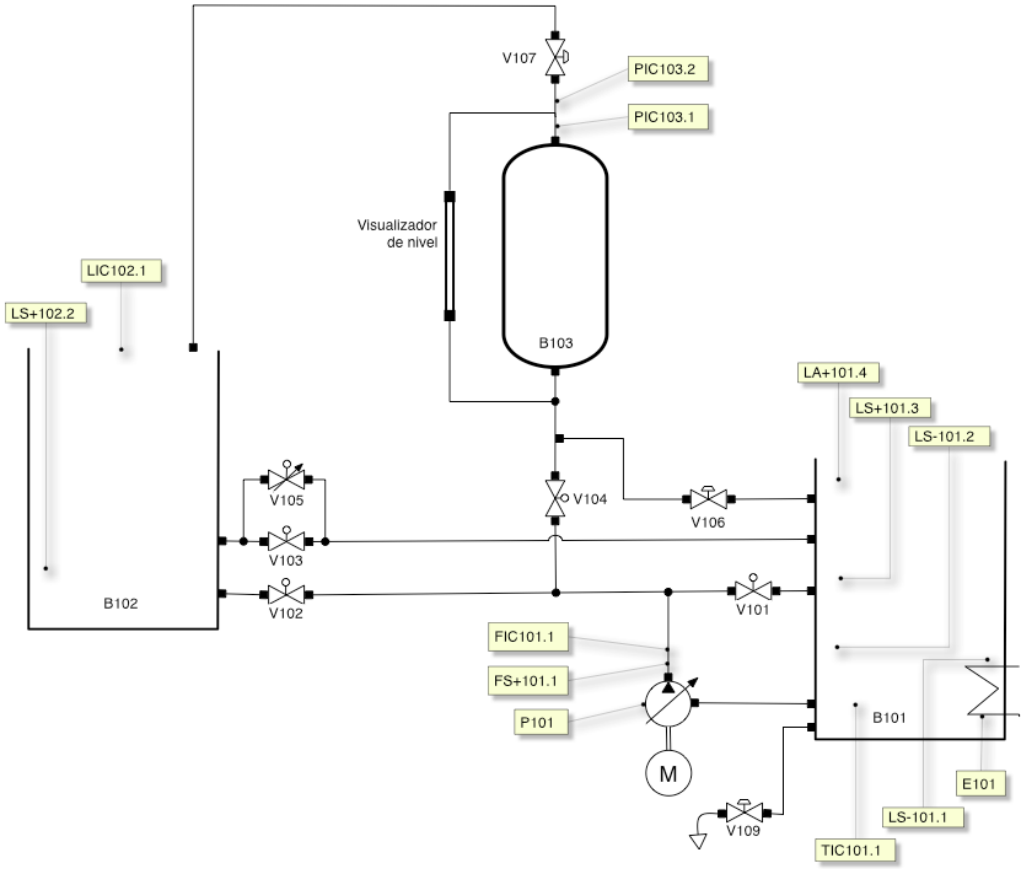

Figura 3: Diagrama de tuberías y sensores de la planta modificada.

indicando que se ha recibido. Antes de que se pueda producir la comunicación se debe configurar la unidad. Se realiza desde el PC enviando la trama “setup0”. La unidad al recibir dicha trama se configura respondiendo con “setup1” ó “setup2”, etc. según sea el número de unidades conectados entre si.

5. Software desarrollado.

5.1. Elección del lenguaje de programación [4].

La elección del del lenguaje de programación estuvo condicionada por la necesidad de desarrollar un software que fuera fácilmente reprogramable y que a su vez esta se pudiera hacer de forma gráfica, lo más parecido a los esquemas de bloques que se usan en Regulación Automática. Matlab era, desde ese punto de vista el mejor candidato. Pero presentaba dos desventajas: no se pueden crear ejecutables de libre distribución y no existe la posibilidad de manejo del programa a distancia vía Internet. Finalmente se optó por labView que además de ser un lenguaje de programación gráfico permite crear ejecutables de libre distribución y la el control a distancia mediante su “Web Server” y el protocolo http.

5.2. Modo de programación.

Se utilizó la forma de programación denominada “Máquina de Estados” [5]. En la cual se considera que el programa tiene varios modos de funcionamiento y que pasa de unos a otros cuando ocurren ciertos eventos. Estos pueden ser internos o externos al programa. Cada uno de estos estados puede considerarse como un programa independiente pero relacionado con los demás por algunas variables.

Un programa en LabView basado en máquina de estados se realiza mediante una estructura “While loop” que contiene otra estructura “Case” (fig. 4). El bucle más externo se ejecuta hasta que se llega al estado final. Cada vez que se repite este bucle la estructura Case ejecuta la opción correspondiente al estado actual.

5.3. Estructura del proyecto.

El proyecto desarrollado se denominó “Capataz”. La última versión es la 3.1. Contiene tres programas que se ejecutan simultáneamente: Principal, Capataz y Pinocho. El programa Principal realiza las tareas iniciales: puerto de comunicación, simulación o real, etc; Capataz es el controlador propiamente dicho y Pinocho es el simulador.

El programa Principal se ejecuta en tres fases: primero inicia las variables globales, segundo modifica las variables correspondientes al puerto de comunicación, los valores de conversión a unidades de ingeniería y los valores de ruido y tercero lanza Capataz y en su caso Pinocho. Llegado a este punto el programa queda a las espera de que el programa Capataz termine.

5.4. Comunicación con la unidad.

Para poder establecer la comunicación con la unidad es preciso configurarla. Para ello se debe enviar la trama “setup0”. La unidad debe responder en un tiempo máximo de 100ms. En caso contrario se considera que no se puede establecer la comunicación.

Como la unidad es mucho más lenta que el PC, las respuestas a las peticiones de datos de los sensores pueden perderse, especialmente cuando la unidad se está configurando, es preciso pues utilizar una estrategia que evite la generación de errores y por lo tanto la interrupción del programa. Para la lectura de los sensores se envía vía puerto serie la trama correspondiente a la petición de datos, posteriormente se escucha el puerto serie durante 500ms si la trama de respuesta no se recibe se vuelve a enviar la petición y se vuelve a escuchar. El proceso se repite por cinco veces o hasta que se recibe una trama correcta. Si no es así se considera que se ha perdido la comunicación con la unidad. Para el envío de datos la estrategia es

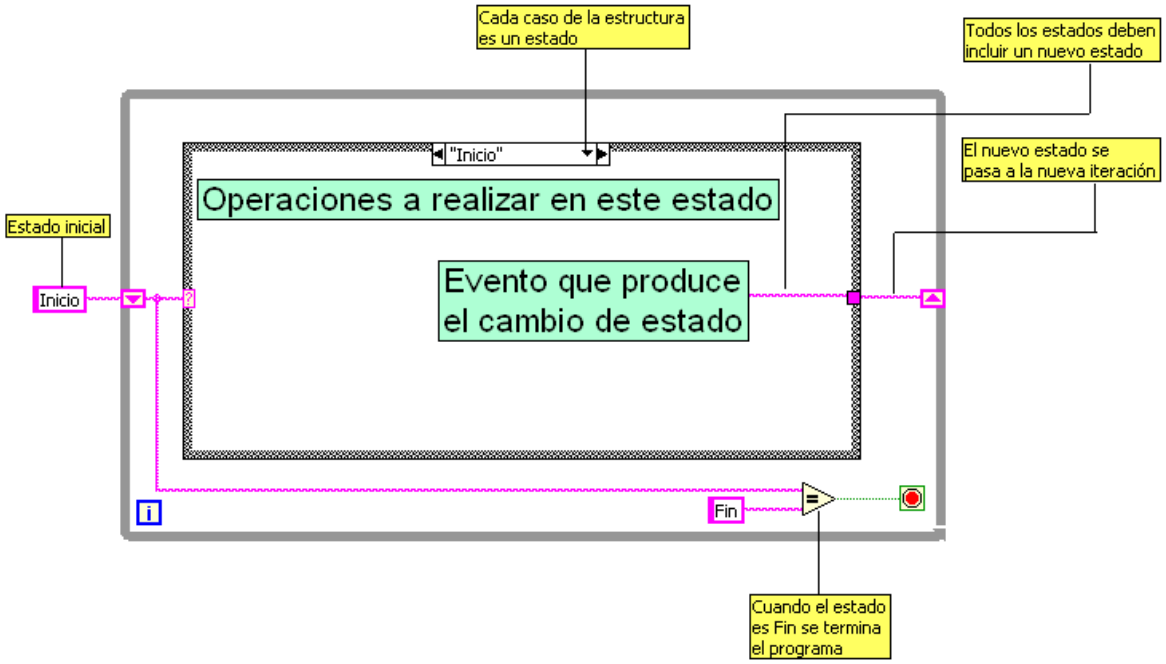

Figura 4: Desarrollo de la máquina de estados con LabView.

parecida: se envía la trama que contiene el dato a enviar y se escucha el puerto serie hasta recibir una trama. Como la unidad gestiona más rápidamente esta clase de tramas sólo se esperan 100ms para considerar que la conexión con la unidad se ha interrumpido.

5.5. Capataz.

El programa capataz está estructurado en 12 estados (Fig. 5). El primer estado es "Configuración" se realizan las operaciones previas que no se realizaron en Principal. "Inicio" es un estado destinado a configurar el experimento: tiempo final, etc. Si se pulsa el botón de inicio del experimento (Marcha) se pasa a "Abre Puerto", si se ha elegido un puerto real se abre. "Configura unidad", "Configura planta" y "Conexión" se ejecutan sucesivamente. "Funcionamiento": Este estado está destinado a la realización de las tareas propias de un lazo de control: lectura de sensores, comparaciones, integración, modificación de los valores de los actuadores, etc. El estado "Control" determina si corresponde parar o continuar el experimento. Esto viene determinado por el tiempo de ejecución, si se ha pulsado o no el botón de paro del experimento (Alto) o de si se han producido errores. En el estado "Proceso final" se para la planta, se pregunta al operador si desea guardar los datos leídos, si desea realizar otro experimento o por el contrario cerrar el programa. En este último caso se cierra el puerto si era real. El estado "Fin" es el último, cierra el simulador si estaba activado y Capataz termina. Los estados "Error de conexión" y "Error de funcionamiento" gestionan los errores que se puedan producir durante la conexión a la unidad o durante el experimento respectivamente.

5.6. Programación a alto nivel.

Se ha deseado que el software desarrollado pueda ser utilizado por un operario (alumno) que no tenga, necesariamente, conocimientos de programación de LabView y que pueda ser modificado por un programador (profesor) que tenga conocimientos básicos. Para facilitar la reprogramación se han creado

un conjunto de SubVI con funciones semejantes a los bloques que se utilizan en teoría y se han agrupado las conexiones para facilitar la visualización del esquema (Fig. 6).

La señal está configurada como un cluster [6] (conjunto de variables de igual o diferente tipo) con dos elementos: magnitud y tiempo. Los sensores y las consignas crean la señal. El tiempo corresponde al momento en el que se empaqueta la señal, tomando como cero el inicio del experimento. Los controladores modifican la señal. Algunos bloques precisan del tiempo, como por ejemplo el Integrador, también es necesario para su representación gráfica o para guardar posteriormente los datos. En las operaciones de suma o resta prevalece el tiempo del segundo operando. Debido a que es por este terminal por donde se introduce la señal que proviene de los sensores, en la mayor parte de los casos.

Existe un bloque para cada uno de los sensores : nivel, caudal, presión, temperatura, sensor digital de flujo y sensores de nivel; y para cada uno de los actuadores: bomba, válvula proporcional, calefactor, selector modo bomba, conexión válvula proporcional, encendido de la bomba en modo digital y apertura de las válvulas V101 a V104. También se crearon SubVI que permiten definir las consignas: escalón, rampa, seno y programable. Para las operaciones con la señal: suma y resta, control proporcional, integrador, integrador saturable, derivador, PID, disparador de Schmitt y control programable entre otros. También se crearon SubVI que permiten guardar la señal en cualquier punto del diagrama.

El programador “dibuja” sobre el estado “Funcionamiento” un diagrama de bloques (Fig. 6), colocando en la parte derecha los sensores y consignas, más a la izquierda los controladores y luego los actuadores. Si el programador necesita más espacio puede crear un subVI donde dibuje el diagrama. Se dispone de una pantalla gráfica para monitorizar la señal en cualquier punto del diagrama. Opcionalmente se podrían incluir más de una si se quiere comparar las señales.

5.7. Cambio a unidades de ingeniería.

Los sensores y adaptadores de señal envían a la unidad una tensión comprendida entre 0 y 10V que es proporcional al valor de la magnitud medida. Este valor se envía al ordenador en 16bits en formato de complemento a 2. El controlador debe convertir el número hexadecimal recibido de la unidad a tensión de 0 a 10V y posteriormente a unidades de ingeniería: unidades de nivel, caudal, etc. En esta conversión hay que tener en cuenta que no se recibirán números negativos. Puesto que las condiciones de la planta pueden variar se ha implementado la posibilidad de cambiar los parámetros de la conversión.

5.8. Manejo del programa.

El programa se lanza ejecutando Principal. Cuando se ha elegido el puerto de comunicación (Fig. 7) y realizados los cambios en los parámetros del cambio a unidades de ingeniería y del ruido del simulador. Se pulsa “Hecho”. Aparece el panel frontal de Capataz (Fig. 8). Este panel varía en función del controlador que se diseñe. Una vez ajustados los valores iniciales de los controles, se pulsa Marcha. A continuación se realizan los ajustes iniciales en la planta (Fig. 9). Terminado esto comienza el experimento. Terminado este, el programa pregunta sobre lo que debe hacer con los datos leídos. Hay tres opciones: guardarlos en un fichero, presentarlos en pantalla o borrarlos. La opción de presentación en pantalla permite obtener los datos cuando se utiliza el programa de forma remota. Hecho esto el programa o bien se cierra o bien vuelve al inicio del experimento, según la elección del operador. Los datos estos pueden ser exportados directamente a una hoja de cálculo u otro programa como Derive o Mathematica. El manejo no requiere ningún conocimiento de programación en lenguaje G.

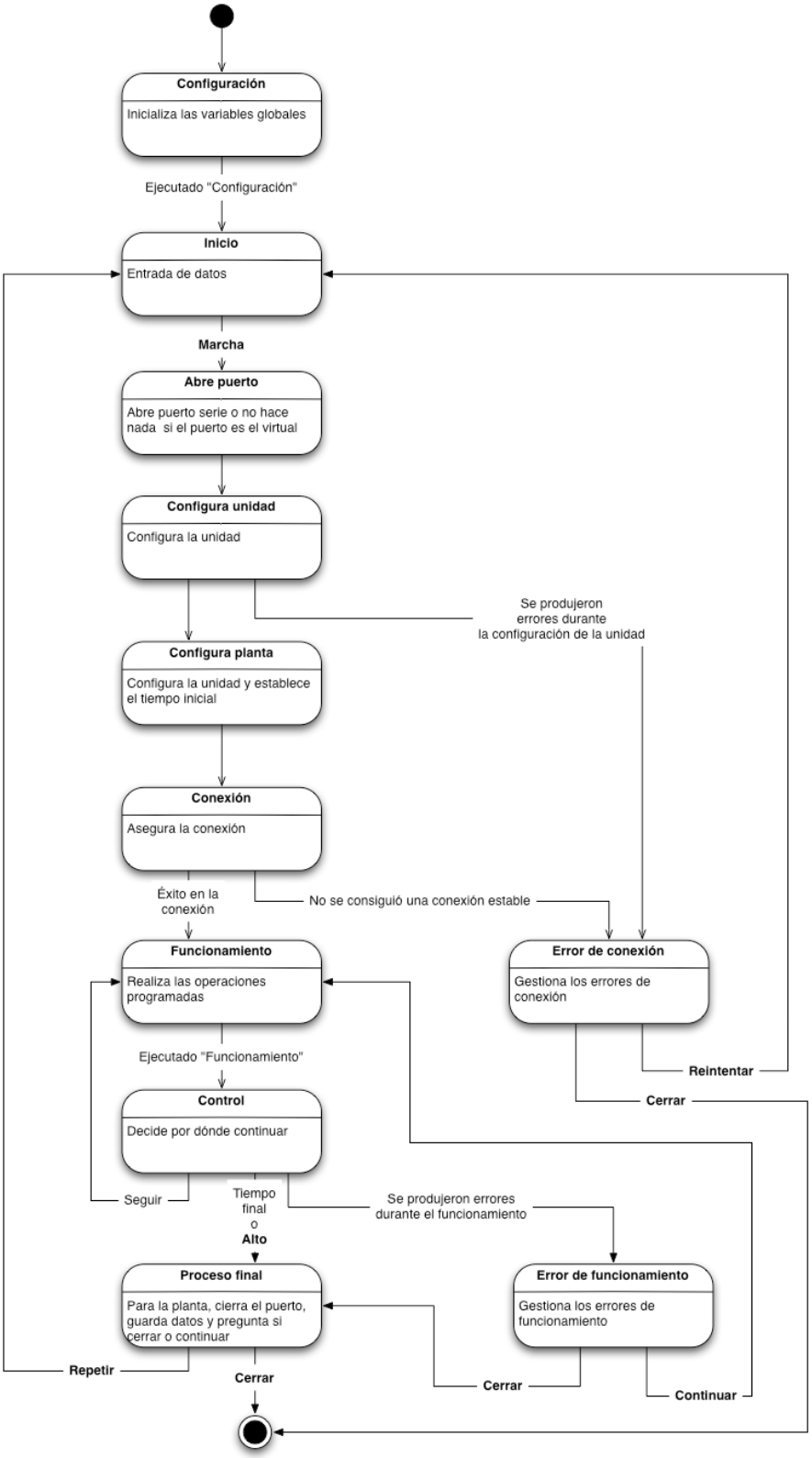

Figura 5: Diagrama de estados del programa Capataz.

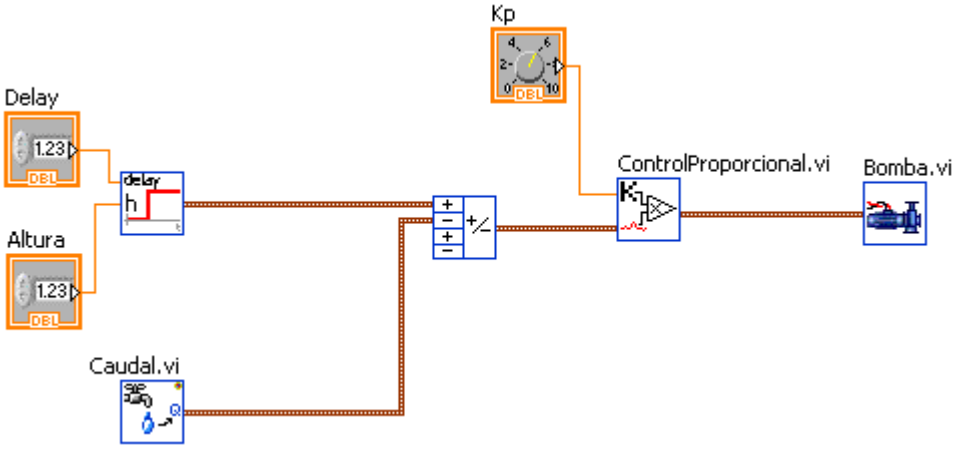

Figura 6: Diagrama de un controlador P de caudal.

5.9. Simulador.

Se desarrollaron dos simuladores muy diferentes: Pinocho v1 y Pinocho v2. La primera versión estaba destinada a servir de ayuda al desarrollo del programa principal. Su uso es muy limitado. La segunda versión trata de reproducir más fielmente el comportamiento de la planta.

El simulador en su versión 2 es un programa que calcula cada 10ms las variables de estado de la planta, que son 23 de las cuales 10 son analógicos y 13 digitales

El simulador utiliza una estructura "Timed Loop" [7] que engloba a otra estructura "Flat Secuence" con dos marcos. En el primero calcula el incremento de tiempo desde el cálculo anterior y en el segundo se calculan los valores de las variables de estado.

Cuando se elige el puerto virtual, es decir simulación, el programa no envía tramas de petición o de envío de datos, simplemente lee o escribe sobre las variables de estado que están almacenadas en variables globales.

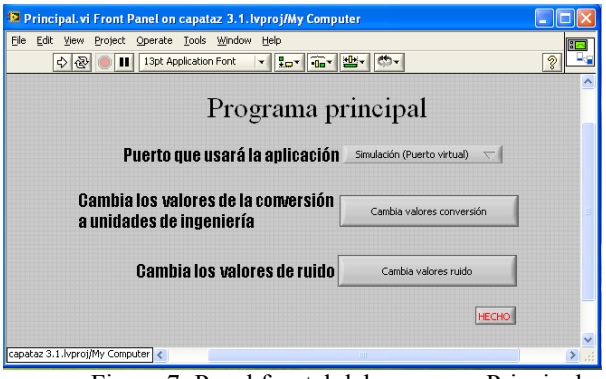

Figura 7: Panel frontal del programa Principal.

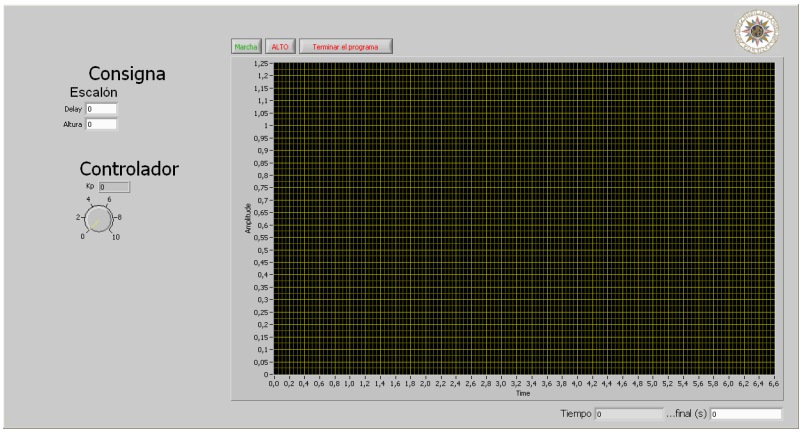

Figura 8: Panel frontal del programa Capataz.

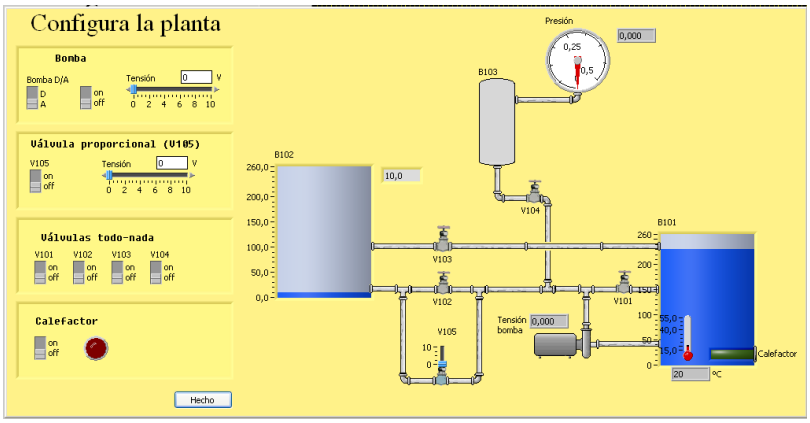

Figura 9: Panel de configuración de la planta.

6. Prácticas.

La utilidad de del programa desarrollado es la realización de prácticas de Regulación Automática. que pueden ser de dos niveles: básico y avanzado.

6.1. Prácticas de nivel básico.

Estas están destinadas a los alumnos con un nivel bajo o medio en sistemas de Regulación Automática y sin conocimientos de programación de LabView. Los objetivos principales son: mostrar algunos conceptos y comprobar diseños en un sistema real. Consisten en ejecutar una aplicación cerrada en la que sólo se podrá modificar algunos parámetros del controlador, que son fijados por profesor del curso.

6.2. Prácticas de nivel avanzado.

Están destinadas a personas con un buen nivel de conocimientos de Regulación Automática y un nivel básico de programación en LabView. Consisten en la ejecución de la aplicación abierta de forma que se pueda realizar la programación del controlador desde cero. También pueden considerarse en este nivel la programación de prácticas de nivel básico.

6.3. Proceso de creación de una práctica.

En primer lugar se debe programar el controlador. Para ello se hace una copia del proyecto (la carpeta que contiene todos los ficheros), se cambia el nombre al proyecto y se borra el fichero con la extensión ".aliases". Luego se abre el proyecto y "Capataz.vi". En el panel frontal se colocan los controles que se deseen. Luego en el diagrama de bloques se busca el estado "Funcionamiento" y en él se dibuja el diagrama del controlador.

7. Aplicación.

Existen tres formas en las que se puede utilizar el programa: presencial, distribución de ejecutables y a distancia.

7.1. Presencial.

El programa se ejecuta en el computador del laboratorio situado al lado de la planta y el usuario opera directamente en él. Permite la realización de prácticas a nivel básico y avanzado, tanto sobre la planta como sobre el simulador. Pero presenta el inconveniente de que el alumno debe desplazarse hasta el laboratorio.

7.2. Distribución de ejecutables.

Una de las ventajas de la utilización de LabView es la posibilidad de crear ejecutables que pueden correr sin licencia en cualquier ordenador PC, con el único requisito de tener instalado "Run Time Engine" [8, 9]. Puede crearse sólo el ejecutable, en este caso es preciso instalar Run-Time Engine manualmente o bien crear un instalador que lo hace todo.

Una vez se tienen creados los ejecutables pueden distribuirse libremente ya que no requieren licencias de LabView, pero sólo se pueden realizar prácticas a nivel básico y con simulación, salvo que se ejecute en el ordenador del laboratorio.

7.3. Ejecución a distancia.

Otra forma de utilización del programa es mediante la utilización del servidor integrado en LabView "Web Server"[10]. Mediante este servidor se pueden configurar Instrumentos Virtuales como servicios Web utilizando el protocolo http.

El programa se ejecuta en el ordenador del laboratorio pero se controla desde el PC remoto. Este envía los cambios en los controles y recibe el panel frontal. Para ello se emplea el modelo "petición respuesta". Es el PC remoto el que envía las peticiones, que son gestionadas por Web Server de LabView y enviadas al VI, y este remite los datos requeridos al Web Server que los envía al PC remoto (Fig. 10).

Para llevar a cabo la ejecución a distancia es preciso crear una página web que se aloja en C:\Archivos de programa\National Instruments\LabView 8.5\www del PC del laboratorio. Esta página se crea automáticamente mediante la herramienta "Web Publising Tools". Posteriormente puede ser

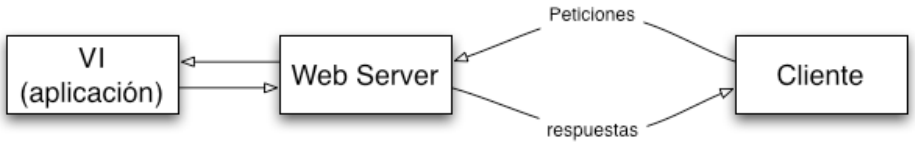

Figura 10: Control remoto de una aplicación: flujo de información.

Figura 11: Integración del programa en Moodle.

modificada para adaptarse a las necesidades de cada práctica.

7.4. Integración en Moodle.

Se hizo una prueba de integración en la plataforma Moodle [11]. Creando una categoría denominada "Regulación Automática I" y un curso llamado "Prácticas de Regulación Automática I" dentro de esa categoría.

El modo del curso fue por temas, uno por práctica diseñada. Cada práctica contiene (Fig. 11): una lección: el procedimiento, tres recursos: descarga del ejecutable, un enlace a la página que permite la ejecución a distancia y una carpeta conteniendo diversos ficheros relacionados con la práctica; y una tarea: enviar el informe de la práctica al profesor. Ademas existe una parte general con instrucciones y con un modelo de informe.

El proceso desde el lado del alumno es: leer el procedimiento, bajar el ejecutable y realizar la práctica con el simulador. Luego realizar la práctica a distancia. Posteriormente redactar el informe y enviarlo al profesor.

Desde el lado del profesor el proceso es: desarrollar o modificar el curso, analizar los informes enviados por los alumnos y calificarlos.

8. Conclusiones

Se ha desarrollado software para el control de una planta piloto de Festo denominada "Compact Workstation". Dicho software resulta fácil de de reprogramar y de usar. Además se ha diseñado un simulador de la planta. El control de la planta puede hacerse vía el Web Server integrado de LabView, lo que permite realizar las prácticas a distancia. Finalmente se ha realizado una prueba de integración en la plataforma Moodle.

Referencias

[1] Festo. *Compact-Workstation*. [en linea]. [Consulta: 22 de julio de 2009]. http://www.festo-didactic.com/es-es/productos/nuevo-automatizaci-n-de-procesos/compact-workstation/

[2] Festo. *EasyPort*. [pdf] [Consulta: 8 de junio de 2009]. http://www.festo-didactic.com/ov3/media/customers/1100/00613626001078835657.pdf

[3] Festo. EasyPort D8A Data Sheet [CD] [Consulta: 21 de julio de 2009]. Editor. Festo.

[4] J. García, D. López, G. Alves, P. Orduña. A*nálisis de requisitos software para un weblab*. Tecnologías Aplicadas a la Enseñanza de la Electrónica. Congreso TAEE año 2008.

[5] National Instrument. NI Developer Zone. *Application Design Patterns: State Machines*. [en linea]. [Consulta: 7 de septiembre de 2009] . http://zone.ni.com/devzone/cda/tut/p/id/3024

[6] A. M. Lázaro. *LabVIEW 6i Programación Gráfica para el Control de Instrumentación*. Editorial Paraninfo, Navalcarnero (Madrid) (2001)

[7] G. W. Johnson, R. Jennings. *LabVIEW graphical programming*. Editorial McGraw Hill, USA (2006).

[8] National Instrument. Support. *Run-Time Engine* [en linea] http://digital.ni.com/softlib.nsf/webcategories/85256410006C055586256BBB002C130D?opendocument&node=132070_US

[9] National Instrument. NI Developer Zone. *Using the LabVIEW Run-Time Engine*. [en linea] [Consulta: 29 de octubre de 2009] http://zone.ni.com/reference/en-XX/help/371361B-01/lvhowto/using_the_lv_run_time_eng/

[10] National Instrument. NI Developer Zone. *Web Services in LabVIEW*. [en linea] [Consulta: 27 de de octubre de 2009]. http://zone.ni.com/devzone/cda/tut/p/id/7350

[11] Moodle. Moodle. [en linea] [Consulta: 26 de noviembre de 2009]. http://docs.moodle.org/es/Página_Principal