XXV Jornadas de Automática
Ciudad Real, del 8 al 10 de septiembre de 2004

CONTROL REMOTO DE POSICIÓN CON JAVA

Perfecto Reguera Acevedo
Universidad de León, diepra@unileon.es

Juan José Fuertes Martínez
Universidad de León, diejfu@unileon.es

Manuel Domínguez González
Universidad de León, diemdg@unileon.es

Ángel Alonso Álvarez
Universidad de León, dieaaa@unileon.es

Resumen

Internet impulsa cambios en la sociedad y el ámbito de la educación no es ajeno a dichos cambios, de tal forma que la enseñanza no presencial basada en Internet gana terreno continuamente y proliferan aplicaciones que tratan de "compartir en la red" sistemas físicos a los que se puede acceder de forma remota. Este trabajo, expone una experiencia en este sentido basada en una estructura abierta, no propietaria y de bajo coste que utiliza elementos ya existentes. La experiencia expuesta es tremendamente didáctica en su funcionamiento como en su conexión a Internet (siempre que los alumnos a los que se les encomiende la tarea dispongan de unos conocimientos medios de programación).

Palabras Clave: *Java*, *JNI*, cliente, triple-capa, servidor, *socket*, Internet

1 INTRODUCCIÓN

En el campo de la enseñanza, se puede observar cómo desde hace unos años se dispone de estructuras tecnológicas que permiten el acceso vía Internet a equipos remotos para el desarrollo de ensayos y prácticas no presenciales en lo que se ha dado en denominar Laboratorios Remotos. [1], [2], [3], [4]

En este tipo de instalaciones, se ha asistido, de forma generalizada a la utilización de determinados paquetes de software comerciales, en los que mediante extensiones o *tools* específicas se suministra una solución fácil y relativamente sencilla para facilitar el acceso a un sistema físico mediante Internet; un claro ejemplo lo constituye *Matlab* y sus extensiones *Matlab Webserver* y *Matlab Real Time*. La ventaja de utilizar este tipo de paquetes es indudable en el sentido de que evita tener que codificar programas y, además, su carácter modular permite incluir acciones de control distintas, distintos tratamientos de datos adquiridos, … etc. Ahora bien, todos los paquetes de software comerciales plantean los consabidos problemas asociados a los cambios de versiones, *drivers* no disponibles, enlaces no eficientes con bases de datos, limitaciones en los tiempos de respuesta ante peticiones de datos, utilización de gran cantidad de recursos computacionales, sobredimensionado del paquete en aspectos no relacionados directamente con la aplicación, vinculación exclusiva con un fabricante y, por supuesto, un coste elevado.

Se presenta en este trabajo una alternativa mucho más eficiente, flexible y, sobre todo, menos costosa que facilita el objetivo inicial de disponer de equipos remotos a los que acceder vía Internet. Esta alternativa hace un uso intensivo de las funcionalidades implícitas hoy en día en los sistemas operativos y herramientas de software *freeware*. La desventaja que presenta dicha alternativa es el mayor esfuerzo de programación necesario, pero también esto debe ser comparado en términos del esfuerzo que hay que realizar para el aprendizaje y la configuración del paquete comercial a fin de adecuarlo a nuestros fines.

La alternativa que se propone sigue una estructura de tres capas denominadas: **cliente**, **intermedia** y **servidor**, en la que los clientes son siempre *applets Java*, la parte intermedia será implementada en *Java* como servidor TCP/IP y la capa servidora la constituye el sistema físico, pues es éste el que sirve los datos [3], [5].

2 ESTRUCTURA

La idea que subyace en esta estructura es evitar utilizar aplicaciones de terceros y desarrollar una

aplicación cliente-intermedia-servidor totalmente abierta con la estructura expuesta en la Figura. 1.

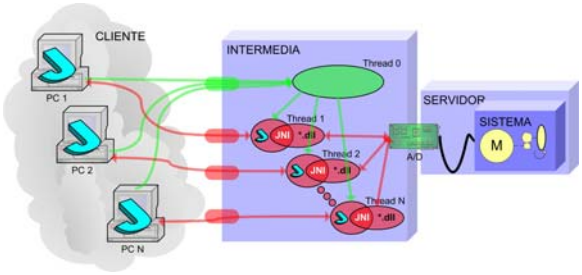

Figura. 1. Servidor *Java*

Se implementa, como capa intermedia, un servidor *Java* que atiende a los clientes que le solicitan datos utilizando *sockets* TCP (atiende peticiones de servicio en un único *socket* y crea un *socket* adicional independiente para cada cliente a fin de proporcionarles el servicio solicitado) y que se comunica con el sistema físico por medio de una tarjeta A/D y con ella por medio del Interfaz JNI (interfaz bidireccional que permite a las aplicaciones *Java* llamar a código nativo y viceversa) [6]

La estructura es muy parecida a la que se obtiene utilizando *Matlab WebServer* y *Matlab Realtime*, pero con la ventaja de que el interfaz *JNI* con la tarjeta puede realizar todo tipo de tareas y puede formar parte de otras aplicaciones directamente, sin la necesidad de instalar obligatoriamente *Matlab* (caso del *Realtime* que lo necesita). Es decir, se dispone de un interfaz que se puede utilizar directamente en otras aplicaciones sin necesidad de instalar más que una dll; esto supone un carácter modular de la estructura propuesta en el sentido de que se puede cambiar la tarjeta y todo operará normalmente siempre que se construya un interfaz JNI adecuado.

Dado que las herramientas de desarrollo de *Java* pueden conseguirse de forma gratuita, el único coste que supone esta estructura es el de las horas destinadas al desarrollo del interfaz *JNI*, del servidor *Java* y los *applets* clientes, coste que también es necesario considerar cuando se trabaja con paquetes comerciales. Como el desarrollo de los clientes y servidor se realiza una sola vez, el desarrollo del interfaz *JNI* para cada tarjeta A/D es lo que puede elevar un tanto dicho coste. Ha de decirse que el coste de desarrollo de la estructura suele ser mayor que el de los paquetes comerciales, pero sólo la primera vez. Posteriormente, no es necesario adquirir nuevas versiones debido a que basta con recompilar el código con nuevas versiones de software de desarrollo que se encuentran actualizadas periódicamente en Internet.

Si se tiene en cuenta la eficiencia de la estructura propuesta frente a la que los paquetes comerciales ofrecen, en pruebas de campo y con el mismo hardware, el servidor *Java* es mucho más seguro en su operación y en el tiempo de acceso al sistema físico.

La operación del servidor *Java* es más segura porque es un servidor pequeño, muy orientado a atender peticiones y responderlas y no se le requiere nada más, con lo que la carga de trabajo que introduce en el PC en el que se halle instalado es muy baja. Los paquetes comerciales, incluyen muchas otras funcionalidades que suelen interferir cuando se les solicita un servicio de acceso concurrente a un recurso físico, necesitan tanta potencia de CPU que se pueden "colgar" con facilidad, además de que no consiguen la misma velocidad de acceso al recurso físico que la opción *Java*.

La eficiencia es mayor, entre otras causas, porque la representación gráfica de las variables, que en el caso de la estructura propuesta se realiza en el cliente, en el caso de los paquetes comerciales se suele realizar en el servidor (se generan páginas HTML dinámicamente para mostrar los resultados). Es decir, se ha balanceado la carga de trabajo repartiéndola entre el servidor *Java* y los clientes.

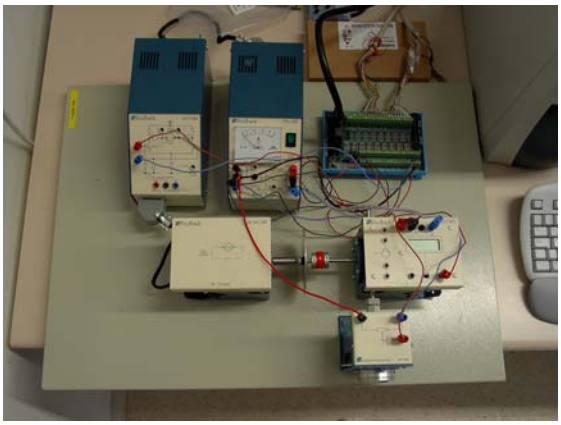

Figura. 2. Foto de control posición feedback

Desde el punto de vista económico, de eficiencia y de flexibilidad, la opción de construir, con la estructura propuesta, un acceso remoto para un sistema físico es mucho más ventajosa que la opción de utilizar un software comercial. Otro punto de vista que ha de tenerse en cuenta es el educativo; en este sentido, la estructura propuesta ofrece a los docentes la posibilidad de proponer trabajos muy interesantes en la docencia de la automática: el efecto del *aliasing*, generación de perturbaciones en la tarjeta A/D, distintas estrategias de control posibles, … etc. Dichos trabajos, cuando son realizados por alumnos con conocimientos de programación medios, ofrecen una perspectiva adecuada de lo que conlleva la

utilización de un ordenador en el control de procesos físicos.

La estructura indicada en este apartado se implementó para distribuir por Internet un sistema físico constituido por un equipo Feedback MS-150 (ver Figura. 2). En él, se implementó un control de posición simple y un interfaz que permitía la elección y configuración de diferentes acciones de control, tanto lineales como no lineales, a fin de comprobar cómo se comporta el sistema ante las distintas posibilidades ofrecidas.

3 IMPLEMENTACIÓN

Una representación funcional de la implementación física realizada es la indicada en la Figura.3, en la que se representa un control de posición clásico. En la implementación realizada, se pretende que los alumnos puedan observar cómo se comportan distintos reguladores de control en el lazo indicado (ellos no pueden cambiar el tipo de regulador es el profesor el que realiza esta acción para conseguir que todos observen el mismo regulador aunque sí pueden cambiar las cosignas).

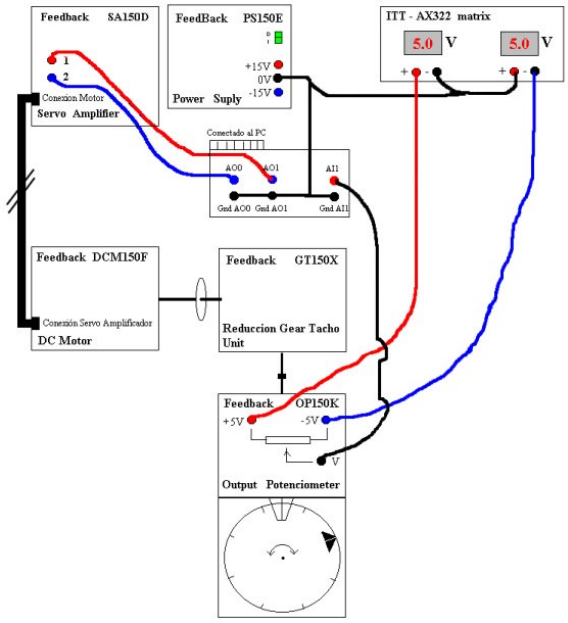

Figura. 3. Implementación realizada

En la parte del servidor, se ofrece la posibilidad de modificar el tiempo de muestreo de la tarjeta A/D (observar cómo afecta este hecho al control), configurar el rango de tensiones de los canales de la misma y configurar qué canales son los que se utilizan para la adquisición de la señal de salida y entrada de la consigna. (ver Figura 4)

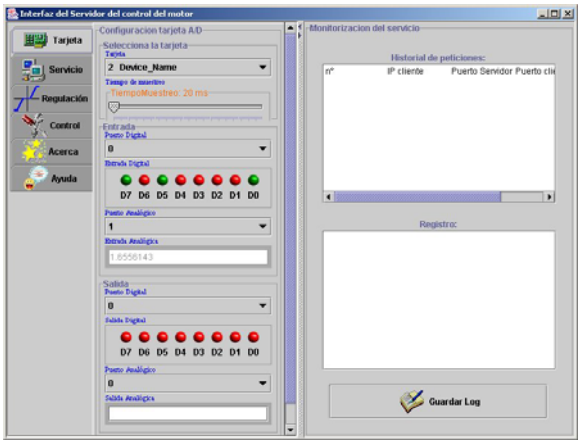

Figura. 4. Interfaz servidor. Tarjeta

Para que la parte servidor opere adecuadamente, es necesario arrancar el servicio. Para ello, se dispone de una interfaz gráfica (ver Figura 5) en la que se indica la dirección IP (*Internet Protocol*) del servidor, el puerto de entrada en el que atiende peticiones, el número de clientes máximo que admitirá y el rango de puertos de salida por los que sirve a los clientes. Además, permite monitorizar en tiempo real los clientes que están conectados y lleva un registro de este hecho a lo largo del tiempo.

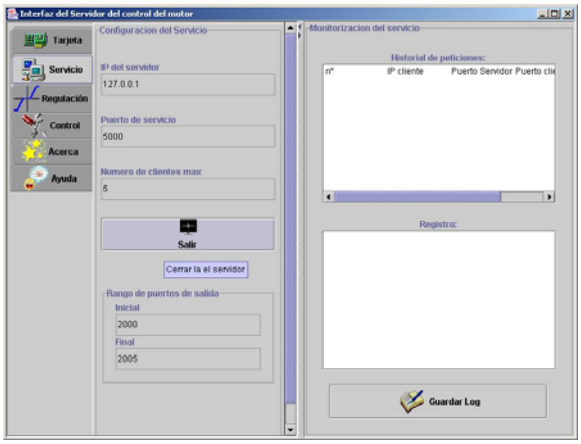

Figura. 5. Interfaz servidor. Servicio

Realmente se ofrecen tres tipos de reguladores: un regulador proporcional al que se le pueden añadir no linealidades como tiempo muerto, histéresis y saturación, un todo nada con las mismas opciones y un regulador genérico que se puede modificar a voluntad con el cursor del ratón (se pueden definir tantas no linealidades como se desee). La modificación de los distintos reguladores, como se puede observar en las figuras 6, 7 y 8 se realiza mediante controles deslizantes para cambiar su magnitud y botones para decidir qué opción tomar.

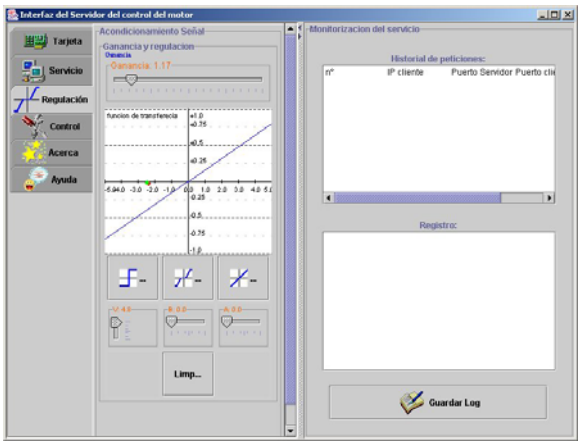

Figura. 6. Interfaz servidor. Regulador proporcional

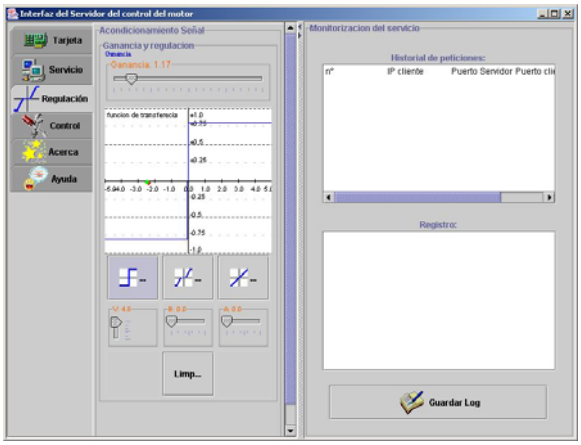

Figura. 7. Interfaz servidor. Regulador todo-nada

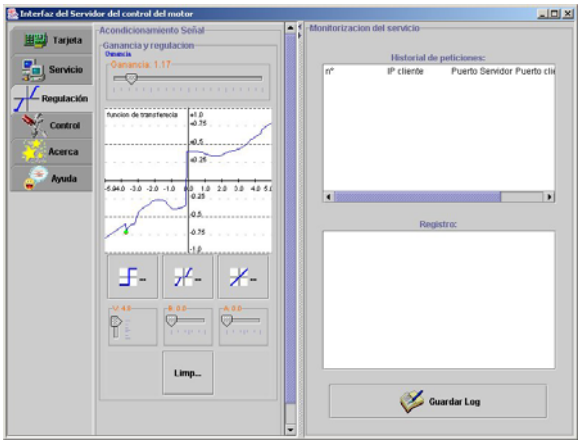

Figura. 8. Interfaz servidor. Regulador genérico

Como es posible que el sistema implementado se vuelva inestable alguna vez o aparezcan ciclos límite, se ha implementado un interfaz de control (ver Figura 9) que permita detectar estas situaciones.

En el caso de que se detecte inestabilidad, el sistema hará que el motor gire continuamente ante un cambio de consigna dado; ante esta situación, se ha definido en el interfaz de control que el número de vueltas máximo sea 4 (o cualquier otro valor que se desee). Cuando se ha alzanzado ese número de vueltas, el interfaz servidor para el control e indica el hecho en los interfazs clientes.

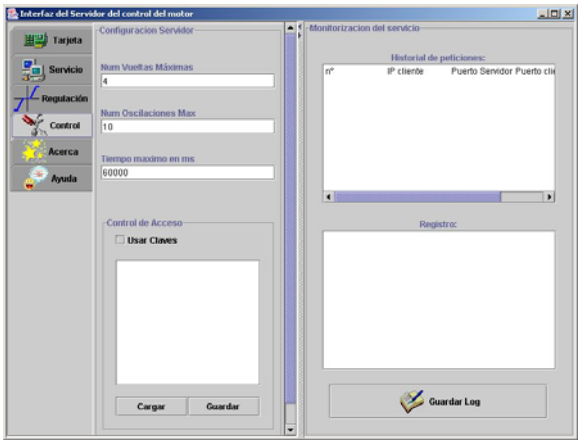

Figura. 9. Interfaz servidor. Control

En el caso de que aparezca un ciclo límite, se realiza la misma acción anterior, sólo que se define un número de oscilaciones máximas y un tiempo máximo que no se deben superar (14 oscilaciones y 60000 ms en la figura).

En el caso de los clientes, el interfaz de los mismos es muy parecido al del servidor (ver Figura 10) y en él los alumnos definen la IP y puerto del servidor al que quieren conectarse y pueden indicar el sentido de giro y el valor que desean, además de visualizar, en tiempo real, la evolución del sistema de control.

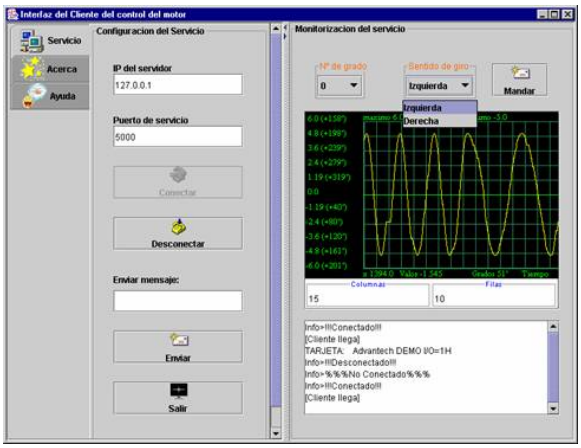

Figura. 10. Interfaz cliente

En la gráfica (ver Figura 11) que se proporciona a los alumnos se indica la posición absoluta en la que se desea que se sitúe el motor. La gráfica se puede detener si se hace click sobre ella y se vuelve a arrancar repitiendo la misma operación. Mientras la gráfica esté detenida, el cliente va almacenando los mensajes que llegan desde el servidor para que cuando se vuelva a arrancar continue en el estado en

el que estaba, aunque va retrasado respecto al estado actual en el que se encuentra el motor.

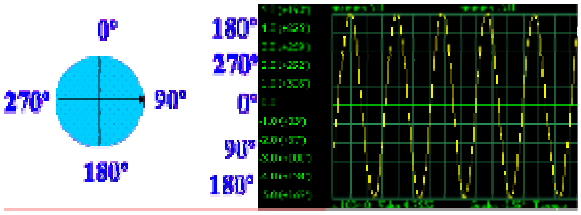

Figura. 11. Gráfica de posicionamiento

En la parte inferior derecha del interfaz de los clientes, se encuentra la ventana de estado en la que van apareciendo los eventos ocurridos durante la ejecución del programa cliente, para ayudar al usuario a conocer su situación y tener presentes las acciones que ha realizado y las que puede realizar.

Figura. 12. Ventana de estado

4 CONCLUSIÓN

De la utilización de la estructura expuesta en este trabajo, se pueden extraer las siguientes conclusiones:

- Es muy útil, desde el punto de vista docente, que el profesor pueda controlar totalmente qué estructura de control se ofrece a todos los alumnos, ya que controla perfectamente las condiciones de la práctica que se propone a los mismos.

- La estructura propuesta es bastante flexible en el sentido de que se pueden visualizar muchos efectos: cómo afecta el tiempo de muestreo al control, aparición de ciclos límite con controles no lineales, cómo afecta la ganancia de un controlador al control y a las características del ciclo límite, cuál es el efecto de la zona muerta, cuál es el efecto de la saturación, cuál es el efecto de la histéresis y cómo evoluciona el sistema cuando se introduce un regulador cualquiera.

- Los alumnos prefieren realizar las prácticas cuando a ellos mejor les convenga y suelen ser bastante receptivos a este tipo de propuestas de carácter remoto.

- Sería muy deseable ofrecer a los alumnos no sólo una gráfica de evolución de la señal de salida, sino también una imagen, en tiempo real, de evolución del motor. Esto supone otros problemas y de ahí que se deje para un desarrollo posterior.

- La implementación realizada es abierta y modular y se ha construido sin ningún software adicional, salvo las dll del fabricante de la tarjeta A/D necesaria para comunicarse con el sistema físico desde el PC.

- Su funcionamiento es muy estable y prácticamente no da ningún problema de operación.

Referencias (10 ptos, negrita)

[1] Aktan, B., Bohus, C.A., Crowl, L.A., and Shor, M.H., (1996) Distance learning applied to control engineering laboratories. IEEE transactions on Education, 39(3), 320-326.

[2] Antsaklis, P., Basar, T., De Carlo, R. (1999). IEEE Control Systems Magazine. Report on the NSF/CSS Workshop on New Directions in Control Engineering Education. Nº 19.

[3] Domínguez, M., Marcos, D., Reguera, P., González, J.J., Blazquez, L.F., (2001) Connection Pilot Plant to the Internet", *IFAC Internet Based Control Education.* IBCE01, Madrid. España.

[4] Dormido,S. (2002). Control Learning: present and future. *15th IFAC World Congress*.Barcelona.

[5] URL_01; http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnpatterns/html/ArcThree LayeredSvcsApp.asp.

[6] URL_02; http://java.sun.com/j2se/1.3/docs/guide/jni/)