Laboratorio Remoto mediante JSP para el acceso a hardware específico

Jesús Bernal Bermúdez*, Abraham Gutiérrez Rodríguez*, Jesús Bobadilla Sancho*
Jorge Tejedor Cerbel**,José Luis Sánchez Sánchez**
*Dpto. de Informática Aplicada, **Dpto de Organización y Estructura de la Información.
Escuela Universitaria de Informática. Universidad Politécnica de Madrid.
Km. 7 Carretera de Valencia, 28031 Madrid, España
Tfns: * (+34)91336.7862, **(+34)91336.7885
e-mail: jbernal@eui.upm.es, abraham@eui.upm.es, jboby@eui.upm.es
jtejedor@eui.upm.es y jlsanchez@eui.upm.es

XI Congreso Argentino de Ciencias de la Computación
IV Workshop de Tecnología Informática Aplicada en Educación

Resumen

La sociedad actual tiene como reto la difusión del conocimiento a todos sus individuos, independientemente de su nivel económico o situación geográfica. Las nuevas tecnologías de Internet en el ámbito de la educación han permitido ampliar y facilitar, de forma revolucionaria, el intercambio del conocimiento. Este proyecto se ha desarrollado y explotado para ampliar la oferta del modelo educativo actual.

Existen muchas razones por la que resulta beneficioso el uso remoto de hardware: un coste elevado o su mejor aprovechamiento, un uso complejo que necesita de especialistas, una necesidad de ubicación lejana al centro de proceso, un uso compartido entre un grupo de personas, etc. Por todo ello, nos parece interesante el estudio que se realiza.

Con este proyecto se ha desarrollado una base de conocimiento e información que permita el acceso a equipos hardware específicos a través de un navegador situado en cualquier punto de la red Internet. Las tecnologías utilizadas son Java, JSP y MySQL. Se ha querido generalizar la naturaleza de los equipos mediante la utilización del conector estándar RS-232.

Finalmente, se implantó el sistema desarrollado para su utilización como oferta educativa en el aprendizaje y uso de tarjetas de control basadas con el microcontrolador Pic.

Palabras clave

Laboratorio remoto, control remoto, Educación a distancia, Campus Virtual.

Introducción

Existen varias experiencias relacionadas con el objetivo de nuestro proyecto. En [3] Davoli describe la experiencia del proyecto LABNET (laboratorio remoto). Su fin es desarrollar una plataforma que permita el acceso y explotación de periféricos no homogéneos bajo una interfaz unificada de usuario, siendo CORBA la tecnología utilizada. Este proyecto está muy en concordancia con la línea de nuestra propuesta. Por otra parte, [4] Liu describe un estudio de la teleoperación de robots a través de Internet, utiliza una arquitectura cliente-servidor mediante el uso de Java. La interfaz del cliente lo implementa mediante *applets* de Java, permitiendo el envío de comandos a través de *sockets*. El servidor recibe los comandos y los retransmite al robot mediante una conexión serie inalámbrica. Los resultados consisten en la captura de vídeo y trasmitida al cliente mediante imágenes GIF con el protocolo HTTP. Finteswalder [5] por su parte describe una arquitectura ciente-servidor, basada en *applets* de Java en cliente y *servlets* de Java en servidor. Rodríguez [6] describe una herramienta denominada SaViO (Sala Virtual de Operaciones), la cual puede ser una referencia para sistemas que pretenden integrar expertos y equipos distribuidos geográficamente para la construcción de un laboratorio virtual. Las tecnologías utilizadas son *applets* de Java para la interfaz del cliente que interactúan con el servidor mediante CORBA. Este trabajo es una solución interesante para el uso remoto de periféricos en el que la complejidad del sistema se considera elevada y necesita la utilización de la ingeniería del software. Es interesante el trabajo de Nedic [7] realiza un estudio comparativo de los laboratorios reales, virtuales y remotos. El trabajo de Fujita [8] presenta una alternativa muy actual de cómo plantear un laboratorio remoto mediante las tecnologías PHP y MySQL.

Para el desarrollo de la interfaz gráfica del cliente se podría utilizar principalmente HTML dinámico con JavaScript, VRML para interfaces tridimensionales o applets de Java (son aplicaciones limitadas que garantizan la seguridad y portabilidad), ésta última es la tecnología más utilizada [3], [4], [5], [6], y [10]. En ambos casos el cliente se despreocupa por completo del software del cliente y tiene garantizada la seguridad en su máquina debido a que los lenguajes son interpretados por la Maquina Virtual de Java que supervisa el acceso al equipo, controlando el uso que tienen de la propia máquina.

Para el desarrollo del servidor se podría plantear mediante ASP (Active Server Page) y el desarrollo de componentes DLL ActiveX, ésta línea es la elegida por Li Hongsheng [12]. También se podría plantear mediante JSP (Java Server Pages) y *servlets* de Java, en esta línea está desarrollado el trabajo de R. Finsterwalder [5]. Es interesante el trabajo de C.C. Ko [11] basado en CGI, pero esta tecnología ha sido superada. En ambos casos, se implementarían formularios para la recogida de datos del cliente y se ejecutarían las peticiones en el servidor, ello daría como resultado código HTML que se le enviaría al cliente.

JSP es una extensión importante a la tecnología *servlet* que permite construir y mantener de forma más sencilla los contenidos Web dinámicos, pues separa el interfaz de usuario de la generación de contenidos, desliga la lógica del negocio de la presentación de manera simple. Además, dado que estamos en un entorno Java, el diseño está basado en componentes reusables. Usar JSP implica insertar en un fichero HTML instrucciones entre las etiquetas. La tecnología JSP usa etiquetas del tipo XML y *scriptlets* escritos en el lenguaje Java para encapsular la lógica que genera el contenido de la página.

Las ventajas de usar las tecnologías JSP y *servlets* son muchas y variadas: el código es 100% portable ya que el API de Java está completamente estandarizado, existen servidores totalmente gratuitos, el rendimiento es muy bueno puesto que se ejecutan como hebras del propio. Con el uso apropiado de ambas tecnologías, se hace mucho más rápido y sencillo el desarrollo de aplicaciones Web. El trabajo de Qu [16] plantea la necesidad de una herramienta que genere un software

educativo (*courseware*). El problema a que se enfrenta es que los contenidos varían constantemente durante el proceso educativo, y por tanto necesita un motor de edición del software educativo que esté claramente separado de los contenidos del curso y que sea capaz de reflejar cualquier modificación de los contenidos del curso. Para conseguir este objetivo, encuentran como perfectos aliados a XML para el interfaz estándar de los datos y JSP como el corazón de la herramienta de edición de contenidos (*publishing*). Por otra parte, Wang [17] usa las tecnologías que existen alrededor de Java (entre ellas JSP) para el desarrollo de un entorno de trabajo basado en Java y web que permita la colaboración de grupos de trabajo de ingeniería dispersos geográficamente.

Cuando la aplicación en su conjunto alcance una complejidad elevada se debería plantear la utilización de CORBA para poder aplicar los paradigmas de la ingeniería del software. Ésta es la opción utilizada por los trabajos de F. Davoli [3] y J.A. Rodríguez [6].

El catálogo de servicios telemáticos que se desarrollará es muy completo, siendo, sin duda, el servicio de videoconferencia [18], el más ambicioso y llamativo, tanto desde un punto de vista técnico como desde la visión de los médicos y usuarios. Bobadilla [18] describe la manera de diseñar un sistema de videoconferencia en tiempo real, basándose en el soporte Java Media Framework (JMF) que proporciona Sun Microsystems como producto de libre distribución. Así mismo, se hacen referencias explícitas al nivel de red que soporta el protocolo Real Time Protocol (RTP).

El servicio de videoconferencia / audioconferencia, requiere unos medios técnicos y un ancho de banda más complejos y costosos, lo que limita su utilización a las situaciones y los usos que mejor se adaptan a sus características. Otro aspecto a tener en cuenta es la posibilidad de encriptar las comunicaciones, cuando sea necesario, con el fin de aumentar la seguridad de los datos. Mengual [19] describe una arquitectura multiagente que proporciona seguridad sobre las comunicaciones que se realicen. Esta arquitectura ha sido probada en el sistema de videoconferencia del artículo anterior.

Tecnologías utilizadas en el laboratorio remoto

El soporte tecnológico utilizado es JAVA. Se ha elegido esta plataforma por ser potente, abierta, escalable y distribuida. Otro factor importante, es la disponibilidad de una documentación muy completa para el desarrollo de aplicaciones. JAVA fue introducido por Sun Microsystems Inc. [13] y ofrece una plataforma independiente del equipo ya que puede ejecutarse en diferentes entornos.

La tecnología Web utilizada es JSP, son páginas de Servidor de Java. Fueron creadas para hacer la creación de contenidos Web dinámicos de una forma mucho más fácil, debido a que tiene una gran cantidad de objetos implícitos. Mientras que la creación de páginas Web dinámicas utilizando servlets de Java puede resultar bastante difícil, por requerir un conocimiento extenso del lenguaje Java, cualquier persona que se inicie por primera vez en el lenguaje Java, puede crear fácilmente una página Web con JSP. La tecnología JSP está realmente construida sobre la base de los servlets, de hecho el servidor compila los ficheros JSP y los convierte a servlets en el momento de su ejecución. Es fácil encontrar aplicaciones en las que se usen de forma conjunta las dos tecnologías: JSP y servlets.

El servidor Web utilizado es Tomcat [14]. Tomcat es una aplicación conocida como “contenedor de servlets”, y es el responsable de recibir las peticiones Web por parte de los usuarios y pasarlas a las aplicaciones Web en Java. Su funcionamiento es como un shell en tiempo de ejecución, y puede ser utilizado como un servidor independiente o como extensión de otros servidores Web, como puede ser Apache, IIS o el servidor de Netscape.

La base de datos empleada en este proyecto nos va a permitir mantener un control de los usuarios que tienen autorización para utilizar nuestro sistema. El diseño de base de datos no es un objetivo de este proyecto, por tanto hemos optado por una base de datos muy sencilla. Debido a su simplicidad, sólo consta de una tabla, no vamos a entra en detalles de diseño de bases de datos, simplemente presentaremos la base de datos utilizada y cómo accede Java a las bases de datos.

La Base de Datos de Nuestro Sitio Web: La base de utilizada está compuesta de una única tabla llamada Usuarios. Los campos que realmente nos interesan son: login y password, estos campos son solicitados nada más acceder a la aplicación, nos van a servir para verificar el login que realice un usuario en el sistema, para ello realizaremos la consulta oportuna en la base de datos.

Se dispone de un gran número de gestores de bases de datos, compatibles con SQL, que será el lenguaje que utilizaremos para comunicarnos con la base de datos (Oracle, Sybase, Informix, Microsoft SQL Server, Access, MySQL…). Si deseamos que nuestra aplicación sea multiplataforma debemos utilizar un gestor de base de datos que también lo sea, en nuestro caso optaremos por MySQL [15].

El puerto serie RS-232C, presente en todos los ordenadores actuales, es un estándar que ha sido muy utilizado. Principalmente se utilizan dos conectores: DB-25 de 25 pines y la versión de 9 pines DB-9. El API de comunicaciones en una extensión estándar del JDK, que permite la comunicación con el puerto serie RS-232 y con el puerto paralelo IEEE-1284, es decir, nos permite realizar aplicaciones que utilicen los puertos de comunicaciones, como es el caso de nuestra aplicación que realiza la programación de tarjetas PIC mediante el puerto serie RS-232.

El paquete proporciona soporte para dispositivos serie y paralelo al estilo Java, es decir, utilizando una semántica semejante a la que se usa con streams y eventos. Para comunicarse con un dispositivo serie a través de uno de los puertos serie de un ordenador, desde una aplicación Java o un applet, es necesario un interfaz. El API de Comunicaciones Java, permite transmitir y recibir datos a través de dispositivos conectados al puerto serie; proporcionando además un conjunto de opciones que permiten la configuración de todos los parámetros asociados a los puertos serie y paralelo. Este API es una proposición para establecer un método estándar de acceso a los puertos de comunicaciones, que permita a los autores de software de comunicaciones escribir programas Java independientes de plataforma. Así se pueden escribir programas para emulación de terminales, programas de fax, lectores de tarjetas, tarjetas PIC, etc. En API se puede obtener de la página Web de Sun http://java.sun.com.

En el paquete de comunicaciones de javax.comm tenemos una serie de clases que nos permiten tratar la comunicación a varios niveles: alto (CommPortIdentifier y CommPort) estas dos clases nos van a permitir acceder a los puerto de comunicación, medio (SerialPort y ParallelPort) estas clases cubren el interface físico para el Puerto serie RS-232 y bajo, en este nivel se encuentra el desarrollo de drivers para el sistema operativo.

La invocación de métodos remotos de RMI (Remote Method Invocation) de Java permite que un objeto que se encuentra bajo el control de una Máquina Virtual Java pueda invocar métodos de un objeto que se encuentra en ejecución bajo el control de una Máquina Virtual Java diferente. Las dos máquinas pueden estar ejecutándose en el mismo servidor o, lo que lo hace realmente interesante, que se encuentren en máquinas distintas conectadas mediante una red TCP/IP.

La máquina que contiene el objeto cuyos métodos se pueden invocar se llama *servidor*, y la máquina que invoca métodos sobre el objeto remoto recibe el nombre de *cliente*. Por supuesto se puede tratar de la misma máquina, hablaríamos de dos aplicaciones: una aplicación cliente y una aplicación servidor.

Los sistemas distribuidos requieren que las partes que los componen y que se ejecutan en diferentes espacios de direcciones (posiblemente en diferentes máquinas), tengan la capacidad de comunicarse entre sí. Una de las primeras soluciones a esta problemática fueron los sockets, que precisamente tienen la capacidad de comunicar dos procesos, ya sea mediante datagramas o flujos de datos (streams). Sin embargo, los sockets requieren que las aplicaciones implanten sus propios protocolos para codificar y decodificar los mensajes que intercambian, lo que introduce una problemática diferente a la naturaleza del problema a resolver y aumenta la posibilidad de errores durante la ejecución.

La primera alternativa que surgió al empleo de los sockets (y que se implementa en base a ellos), son las llamadas a procedimientos remotos (RPC) donde la comunicación entre los elementos que componen el sistema distribuido, se realiza mediante la invocación de funciones que se encuentran en espacios de direcciones diferentes. En este caso, el programador tiene la "impresión" de trabajar con procedimientos locales, mientras que en realidad el sistema RPC se encarga de empaquetar los argumentos y enviarlos al proceso que contiene el código que implementa a la rutina remota. Los sistemas codifican los parámetros de la invocación, así como los valores de vuelta en una representación externa de los datos. Un ejemplo de este tipo de representaciones externas es XDR (eXternal Data Representation) especificado dentro de la comunidad Internet en el documento RFC 1832.

Como es de esperar, en los lenguajes cuyo paradigma de programación no es el procedimental, sino el orientado a objetos, se requiere ya no invocar procedimientos remotos, sino a métodos de objetos remotos. El empleo de objetos distribuidos Java, en lugar de procedimientos remotos, implica varias ventajas como la orientación a objetos misma, movilidad de las aplicaciones Java, los patrones de diseño, la seguridad, la recolección de basura distribuida, etc.

La principal desventaja de los objetos distribuidos de Java, con respecto a las llamadas a procedimientos remotos y Sockets, es definitivamente el rendimiento. Esta desventaja es análoga a la del mismo lenguaje Java, con respecto a los lenguajes completamente compilados como C, pero de la misma manera que en ese caso, todas las características positivas del modelo de objetos distribuidos de Java, lo hacen una alternativa bastante interesante.

Descripción del laboratorio remoto

El objetivo de este proyecto es desarrollar un sitio Web que sea capaz de interactuar con una placa con un microcontrolador PIC. Mediante el envío de tramas seremos capaces de ordenar al microcontrolador la ejecución de diferentes comandos. Los comandos que deseamos implementar son:

- Cargar aplicación: Permite enviar una aplicación al microcontrolador, dicha aplicación irá contenida en una trama.
- Abortar Aplicación: Este comando tienen como consecuencia la interrupción de la ejecución del programa que previamente hayamos cargado en el microcontrolador.
- Volcado de Memoria RAM: La ejecución de este comando hará que el microcontrolador nos devuelva el contenido de la memoria RAM.
- Volcado de Memoria EEPROM: Este comando nos devolverá el contenido de la posición de memoria EEPROM que le indiquemos en la trama enviada.

Al tratarse de un recurso al que sólo puede acceder un usuario al mismo tiempo, es necesario implementar algún sistema que impida que varios usuarios accedan a la tarjeta PIC, y que al mismo tiempo sea un mecanismo justo y que carezca de inanición. Para ello recurrimos al uso de una cola. Cada usuario dispondrá de una porción de tiempo, pasado ese tiempo el usuario será desconectado de la aplicación, dando paso al siguiente usuario de la cola.

Otro de los objetivos de este proyecto es desarrollar la aplicación de forma que sea idependiente de la plataforma, para ello recurrimos a la tecnología Java, e implementaremos el sitio Web mediante JSP.

La aplicación debe ser capaz de cubrir los objetivos descritos, para ello nos basaremos en una arquitectura como la que se muestra a continuación.

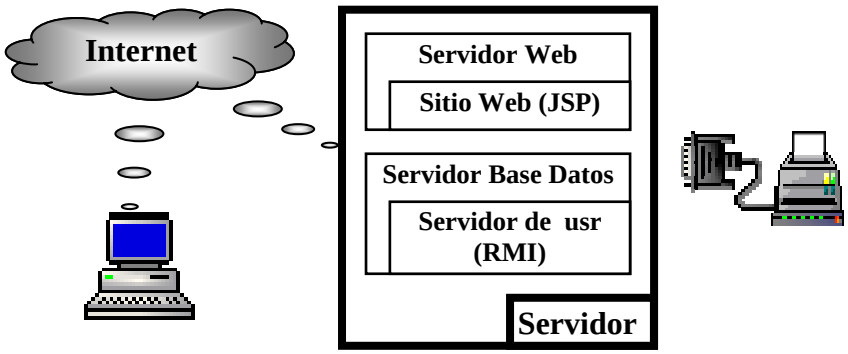

Fig. 1. Arquitectura del laboratorio virtual

Vamos a dividir nuestra aplicación en dos partes:

- Sitio Web: Implementado mediante JSP, contiene las páginas JSP y HTML sobre las que navegará el cada usuario. El sitio Web es el encargado de comunicarse con el Servidor de Usuarios y con la tarjeta PIC. Se ejecuta en un Servidor Web, que es el encargado de atender las peticiones http de los clientes, y devolver la página JSP o HTML que corresponda a cada petición. En nuestro caso el servidor Web utilizado es Apache Tomcat.
- Servidor de Usuarios: Es el encargado de gestionar el acceso de los usuarios a la aplicación, manteniendo una cola con los usuarios y controlando el tiempo del que dispone cada usuario. Se ejecuta en una JVM (Máquina Virtual de Java) que en nuestro caso se encuentra en el mismo servidor.

El cliente siempre accede al Servidor Web, y mediante el uso de clases que posteriormente analizaremos, serán las páginas JSP las encargadas de comunicarse con la tarjeta PIC y con el servidor de usuarios. Este proceso es totalmente transparente al usuario, ya que él sólo navegará por páginas JSP o HTML.

Aunque ambas partes de la aplicación están situadas en la misma máquina, esto no tendría por qué ser siempre así, basta con modificar la dirección de los recursos RMI, indicando la dirección de la máquina donde está situado el servidor de usuarios.

Funcionamiento de del laboratorio remoto.

En este apartado no vamos a entra en detalles sobre la implementación de la aplicación, daremos una descripción de su funcionamiento.

Nada más acceder a la dirección Web donde se encuentre nuestra aplicación aparecerá una página Web llamada *index.html* que muestra una pantalla donde debemos introducir un login y un password. Este documento HTML se encargará de comprobar el acceso no autorizado a la aplicación mediante la consulta de una base de datos dónde se encuentran registrados los usuarios.

Si debemos esperar nuestro turno, una pantalla nos informará del número de usuarios que se encuentran antes que nosotros, y esta misma pantalla dará paso a la aplicación cuando sea nuestro turno. Desde el momento en el que sea nuestro turno, dispondremos del tiempo estipulado por el administrador, para tener un uso exclusivo de la tarjeta PIC. Pasado el tiempo que tenemos asignado la aplicación se desconectará automáticamente.

Si cerramos el navegador o abrimos varias ventanas, nuestro turno se conservará, de forma que nuestro usuario sólo puede encontrarse una vez en la cola de usuarios, evitando de este modo el abuso que se podría realizar por parte de los usuarios registrándose más de una vez en el sistema.

A continuación detallaremos las diferentes opciones del menú y como implementan estas los comandos incorporados en la tarjeta PIC. El microcontrolador PIC recibe una serie de comandos que deben estar envueltos en un protocolo, cada protocolo lo definimos con una trama. Vamos a obviar *Ayuda* y *Acerca de*, ya que se explican por sí solas.

Cargar Aplicación. Esta opción implementa el comando *Cargar Aplicación* de la tarjeta PIC. Seleccionaremos un fichero en hexadecimal que previamente hayamos compilado con un programa adecuado para tal fin. Se procederá a una comprobación sintáctica, y si es correcto, se formará la trama correspondiente que entiende la tarjeta PIC y se procederá a su envío a través del puerto serie RS-232, utilizando para ello la API de comunicaciones.

Volcado de Memoria RAM. El envió de esta trama a la placa provoca que nos devuelva el contenido de la memoria RAM. Indicaremos el tiempo de estudio que deseamos capturar de la tarjeta PIC, este tiempo se indica en segundos mediante la selección en una lista desplegable con el título *Tiempo de estudio*. También es necesario indicar cada cuanto tiempo queremos realizar una lectura de la memoria RAM, este tiempo se indica en milisegundos utilizando una lista desplegable llamada *Tamaño muestra*.

Volcado de Memoria EEPROM. Este comando nos va a permitir obtener el contenido de una posición de memoria EEPROM. Debemos indicar el tiempo de estudio que deseamos capturar de la tarjeta PIC, este tiempo se indica en segundos mediante la selección en una lista desplegable con el título *Tiempo de estudio*. También es necesario indicar cada cuanto tiempo queremos realizar una lectura de la memoria EEPROM, este tiempo se indica en milisegundos utilizando una lista desplegable llamada *Tamaño muestra*.

La diferencia con respecto al Volcado de Memoria RAM radica en que debemos indicar la dirección de memoria que queremos capturar, utilizando dos listas desplegables, la de nuestra izquierda se corresponde con los cuatro bits altos de la dirección de memoria, y la de nuestra derecha con los cuatro bits bajos. El rango de direcciones posibles va desde 00 hasta FF.

Abortar Aplicación. La ejecución de este comando provoca que se detenga la ejecución de la aplicación que se hubiese enviado al microcontrolador anteriormente.

Salir. Esta opción nos permite abandonar la aplicación, cediendo el turno al usuario que se encuentre después de nosotros en la cola de usuarios, en el caso de que exista algún usuario esperando.

Implementación del Servidor de Usuarios. El objetivo del servidor de usuarios es mantener una cola de usuarios, de forma que se pueda controlar el orden de llegada de los usuarios a la aplicación. Cada usuario dispone de un tiempo para poder utilizar la tarjeta PIC, mediante este

servidor podremos hacer que finalizado ese tiempo, o una vez que el usuario pulse la opción de salir, el siguiente usuario que se encuentre en la cola pueda disfrutar del uso exclusivo de la tarjeta PIC. Este servidor es una aplicación realizada en RMI y será el servidor Tomcat, mediante las JSP correspondientes el que acceda a los distintos métodos remotos del servidor.

Fig. 2. Página Web principal

Impacto en el uso del laboratorio remoto.

El laboratorio remoto ha sido un prototipo de experimentación. Se utilizó para la implementación de una práctica de programación de procesadores PIC sobre una placa de propósito genérico. En la Fig. 3 se muestra la imagen de la tarjeta.

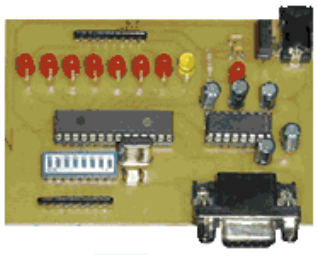

Fig. 3. Placa de propósito general basada en el procesador PIC

El alumno debía implementar un programa que realizara un juego de luces mediante la utilización del Timer. Para ello debe acceder al sitio Web mediante su cuenta y clave y cargar el programa en el periférico. Una vez arrancada la ejecución puede comprobar mediante Web-Cam, si las luces funcionan en la secuencia prevista. Podía contar con la ayuda del maestro de laboratorio mediante los sistemas previstos: chat, audioconferencia, pizarra y videoconferencia.

Debido a la disponibilidad del sistema, 24 horas, ofrecía un servicio excelente y permitía una mejor organización del tiempo del alumno. Respecto a la solicitud de ayuda, se comprobó que la video-conferencia no era práctica debido al ancho de banda necesario. Se observó que el sistema más utilizado en la comunicación con el maestro de laboratorio era el chat, seguido por la pizarra, debido a su bajo consumo de ancho de banda.

Conclusiones

Este artículo describe el diseño y desarrollo de un laboratorio remoto y su impacto en la utilización por los alumnos. Hemos dado una visión generalizada para empezar a plantear como podríamos llevar a buen fin el desarrollo de un laboratorio remoto. No hemos pretendido que se comprenda el concepto de las diferentes tecnologías mencionadas, sino orientar en las diferentes tecnologías disponibles. Se ha presentado una característica interesante al combinar el control remoto con la asistencia remota, ya que facilita la resolución de problemas de forma inmediata. Esta ayuda es muy interesante en la toma de contacto del laboratorio por parte de los alumnos y en la resolución de problemas puntuales.

Bibliografía

[1] K. K. Tang & C.Y. Soh, "Instrumentation on the Internet", *Engineering science and education journal*, april, 2001.

[2] M. Shor & A. Bhandari, "Access to an instructional control laboratory experiment through the World Wide Web", in *Proc. Amer. Contr. Conf.*, 1998, pp. 1319-1325.

[3] F. Davoli, P. Maryni, M. Perrando & S. Zappatore, "A general framework for networked multimedia applications enabling access to laboratory equipment: the LABNET project experience", *Proc. IEEE on Information Technology: Coding and Computing*, 2001, pp. 359 – 365.

[4] Y. Liu, C. Chen & M. Meng, "A Study on the Teleoperation of Robot Systems via WWW", Proc. IEEE, 2000, pp. 836-840.

[5] R. Finsterwalder, "A generic client/server architecture for distributed Web-based simulation experimentation", IEEE Symp. on Computer-Aided Control System Design , 2000, pp. 185 – 189.

[6] J. A. Rodríguez Vázquez, "Sistemas distribuidos de operación y control basados en la Web", *Ingeniería Química*, Madrid, diciembre 1999, pp. 119-123.

[7] Z. Nedic, J. Machotka & A. Nafalski, "Remote Laboratories versus Virtual and Real Laboratories", *Frontiers in Education*, 2003, pp. 1-6.

[8] J. S. T. Fujita, R. F. Cassaniga & F. J. R. Fernandez, "Remote Laboratory", *Industrial Electronics*, june 2003, PP. 1104 - 1106 vol. 2.

[9] Summary: Remote laboratory implements an environment monitoring system via Internet. In the monitored environment, a luminosity sensor that is connected to an analog/digital converter (ADC) is installed. This ADC is linked to an "intelligent module" that cont.....

[10] F. A. S. Goncalves & C.A. Canesin, "Java applets for a www-HTML-based course in power electronics", *Proc. IEEE Power Electronics Specialists Conference*, vol. 1, 2001, pp. 85-90.

[11] C.C. Ko, Ben M. Chen, Jianping Chen, Y. Zhuang & K. Chen Tan, "Development of a Web-Based Laboratory for Control Experiments on a Coupled Tank Apparatus", *Proc. IEEE Transactions on Education*, vol. 44, february, 2001, no. 1,.

[12] Li Hongsheng; Shi Tielin; Yang Shuzi; Li Zhijian; Tao Yunfeng & Chen Wenwu, “Internet-based remote diagnosis: concept, system architecture and prototype”, *Proc. IEEE on Intelligent Control and Automation*, vol.1, 2000, pp. 719 -723.

[13] http://java.sun.com

[14] http://jakarta.apache.org/tomcat

[15] http://www.mysql.com

[16] C. Qu, J. Gamper, & W. Nejdl, "A collaborative courseware generating system based on WebDAV, XML, and JSP ", *Proc. International Conference on Advanced Learning Technologies*, 2001, pp. 197 – 198.

[17] W. Lihui, B. Wong, S. Weiming & L. Sherman, "A Web-based collaborative workspace using Java 3D", *Proc. IEEE Computer Supported Cooperative Work in Design*, 2001, pp. 77 – 82.

[18] J. Bobadilla and S. Delgado, “Real time multicast videoconference system with digital dynamic information”, SCI´2001.

[19] Mengual L., Barcia N.,Bobadilla J., Jiménez E., Setien J., Yáguez, “Arquitectura multiagente segura basada en un sistema de implementación automática de protocolos de seguridad”, SNE´2001.