Propuesta de un Framework Multinivel para el diseño de laboratorio de acceso remoto

Proposal of a Multilevel Framework for the design of remote access laboratories

JAIME ALBERTO BUITRAGO
Magister en Ingeniería énfasis Electrónica
Profesor Asistente. Facultad de Ingeniería. Programa de Ingeniería Electrónica
Miembro del Grupo de Investigación en Sistemas de Información y Control Industrial
Universidad del Quindío
jalbertob@uniquindio.edu.co
Armenia, Colombia

JULIAN ALEJANDRO LAMPREA
Ingeniero de Sistemas y Computación
Profesor Auxiliar. Facultad de Ingeniería. Programa de Ingeniería de Sistemas y Computación
Miembro del Grupo de Investigación en Sistemas de Información y Control Industrial
Universidad del Quindío
julian.lamprea@smadit.com
Armenia, Colombia

EDUARDO CAICEDO-BRAVO
Doctor en Informática Industrial
Profesor Titular, Escuela de Ingeniería Eléctrica y Electrónica
Director del Grupo de Investigación en Percepción y Sistemas Inteligentes
Universidad del Valle
eduardo.caicedo@correounivalle.edu.co
Cali, Colombia

Fecha de inicio: 25/03/2013
Fecha de aceptado: 04/12/2013

Forma de citar: BUITRAGO, Jaime, LAMPREA, Julian y CAICEDO, Eduardo. Propuesta de un Framework multinivel para el diseño de laboratorio de acceso remoto. Rev.UIS.Ingenierías, 2013, vol.12, n.2, p.p 35-45.

RESUMEN

Este trabajo presenta la propuesta de un Framework multinivel para la construcción de laboratorios de acceso remoto para la experimentación en robótica. Este Framework provee la estructura de comunicación para el diseño de un laboratorio de acceso remoto, que se lleva a cabo mediante la implementación de tres capas: capa de conexión, capa web y capa cliente. Así mismo, se define la arquitectura software de cómo se deben construir estas capas, de manera que sea éste quien se encargue de controlar la comunicación entre ellas. Además, se presenta la implementación del laboratorio de acceso remoto para el robot industrial Mitsubishi RV-2AJ de la Universidad del Quindío, donde se utiliza la arquitectura multinivel propuesta por el Framework. El sitio web del Framework está disponible en línea y se puede descargar desde http://www.flarer.net.

PALABRAS CLAVES: Framework, laboratorios remotos, robótica, servlets, applets, XML, HTTP.

ABSTRACT

This work presents the proposal of a multilevel framework for building remote access laboratories for experimentation in robotics. This framework provides the communication infrastructure for the designing of a remote access laboratory, which is carried out through the implementation of three layers: link layer, web layer and client layer. Likewise, the software architecture on how these layers should be built is defined, so that this is responsible for controlling communication between them. In addition, the implementation of the remote access laboratory is presented for the industrial robot Mitsubishi RV-2AJ from Universidad del Quindío, where the multilevel architecture proposed by the framework is used. The website of the framework is available online and can be downloaded from http://www.flarer.net.

KEYWORDS. Framework, remote labs, robotics, servlets, applets, XML, HTTP.

1. INTRODUCCIÓN

Este trabajo ha sido realizado en el marco del proyecto de investigación: "Framework para el desarrollo de laboratorios de acceso remoto sobre redes de alta velocidad (RENATA) en el área de la Robótica", financiado por Colciencias, que pretende diseñar e implementar un esquema de trabajo para la creación de interfaces de experimentación que faciliten la formación e inclusión de nuevos laboratorios remotos de robótica a la red de alta velocidad RENATA (Red Nacional Académica de Tecnología Avanzada), de tal forma que se promueva en el país la construcción de una sociedad del conocimiento alrededor de la robótica. Este framework proporcionará la arquitectura de comunicación para la construcción de laboratorios de acceso remotos para la experimentación en robótica. Además, por medio de este framework se busca un punto de equilibrio entre simplicidad y escalabilidad, debido a que facilita la comunicación entre las capas del sistema y deja la aplicación abierta para incorporar nuevas funciones de manera rápida y de bajo impacto en la arquitectura software.

Un laboratorio de acceso remoto se puede definir como un laboratorio que utiliza una red de comunicaciones, donde los usuarios y los equipos del laboratorio están separados geográficamente y las tecnologías de telecomunicaciones se utilizan para acceder a estos equipos (Bamaby, 2001), (Nuño, et al,2004), (Rosado, et al, 2006). Estos laboratorios son sistemas que comparten equipos remotos, permitiendo a los usuarios realizar prácticas a través de un computador conectado a una red de comunicaciones, como Internet. A través de una interfaz Web, los usuarios pueden cambiar parámetros de control, realizar prácticas, observar y descargar resultados. Este tipo de laboratorios ofrecen el acceso a sistemas físicos, permitiendo la interacción a distancia y la realización de tareas que normalmente se desarrollan en forma local por medio del uso de las tecnologías de información y de las telecomunicaciones (Brady, y otros, 1998). Algunos de estos laboratorios existentes son: El telerobot de Universidad del Oeste de Australia (UWA) (Telelabs, 2009), que tiene una interfaz remota construida a través de LabVIEW, y ROBOLAB desarrollado en la Escuela Politécnica Superior de la Universidad de Alicante de España (RobUALab, 2008) que dispone de una arquitectura que se muestra en la Figura 1 y además cuenta con un simulador 3D basado en VRML (Virtual Reality Modeling Language).

Para la creación de estos laboratorios se requieren conocimientos específicos en herramientas tales como sockets, Web services, transformaciones XML, manejo de streams, implementación de arquitecturas software y esquemas de comunicación adecuados para la interacción con los equipos de experimentación. Esta situación implica un esfuerzo de desarrollo, debido a la necesidad de capacitarse en el manejo de dichas herramientas y tecnologías, lo que se traduce en mayores tiempos en la implementación de este tipo de aplicaciones. Esta experiencia se encontró en el desarrollo de los proyectos en (Calvache, 2009), (Buitrago, 2010), (Jara C., 2008), (Rodriguez, et al, 2001), (Lassó, et al, 2001), (Song, 2002) y (Barnaby, 2001). Estos proyectos se han elaborado a la medida, es decir, se creó una arquitectura software y de comunicación específica para la implementación de cada uno. Además, estos laboratorios han sido desarrollados por una arquitectura software que se compone de tres capas: una interfaz de usuario para el cliente, un servidor web que atiende las peticiones de los clientes y una interfaz de conexión que comunica la aplicación web con los equipos. La capa del servidor web que actúa como *middleware*, es la encargada de comunicar la interfaz cliente con el sistema de conexión de los equipos; es en esta capa en donde se encuentran las principales diferencias entre las distintas implementaciones de laboratorios remotos, ya que la comunicación entre capas, puede ser realizada a través de diversos lenguajes de programación y tecnologías,

tales como Sockets, RMI, CORBA, WebServices, Servlets, CGI's u otros protocolos.

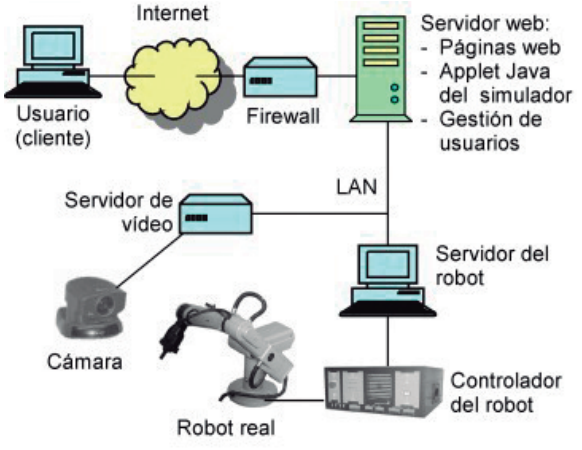

Figura 1. *Arquitectura ROBOLAB (RobUALab, 2008)*

Teniendo en cuenta que las experiencias expuestas requiere una capa de comunicación que, independiente del laboratorio, permita soportar la conexión entre la interfaz del cliente y los equipos, en este trabajo se propone la creación de un framework que defina una arquitectura de comunicación trasparente entre estas dos capas (capa del cliente y capa de conexión a los equipos), sin necesidad de conocer a fondo las tecnologías de comunicación para su implementación. Además, por medio de este framework se busca un punto de equilibrio entre simplicidad y escalabilidad, ya que facilita la comunicación entre las capas del sistema y deja la aplicación abierta para incorporar nuevas funciones de manera rápida y de bajo impacto en la arquitectura software.

2. FRAMEWORK

Un *framework* es una estructura conceptual y una metodología definida que ofrece un conjunto de artefactos o módulos de software concretos, por medio de los cuales se construyen aplicaciones de una forma estructurada. El framework modela un dominio específico y lo representa un diseño abstracto, compuesto por una serie de componentes e interfaces (Riehle, 2000). Así mismo el framework fomenta el uso de un estilo arquitectónico en particular y describe cómo un determinado programa, una interfaz de usuario o un software de comunicación por red, se descompone en capas y componentes.

La principal característica de un framework es que éste es un marco de trabajo para el desarrollo de software a gran escala, con el objetivo de aumentar la productividad y reducir el tiempo de desarrollo. Para lograr lo anterior, se hace necesario utilizar las técnicas de reutilización, que proveen componentes que puedan ser fácilmente conectados para la construcción de un nuevo sistema. El desarrollador de software no tiene que saber cómo un componente está implementado, solo necesita conocer sus especificaciones de uso.

Los frameworks son construidos por flexibilidad y generalidad, tratando de cubrir un dominio amplio en vez de un problema particular (Markiewicz, et al, 2000). Este enfoque produce un generador de aplicaciones que es más complejo y con más puntos de flexibilización que sistemas de software tradicionales. Además, el diseño del framework proporciona el flujo de control, lo que se conoce como inversión de control, porque en un desarrollo usual el flujo de control lo define la aplicación (Morisio, et al, 1999).

Como los frameworks son creados para generar aplicaciones dentro de un dominio, existen puntos flexibles que se personalizan de acuerdo a las especificaciones de un problema en particular (Markiewicz, et al, 2000). Por otra parte, en los frameworks las instancias son producidas por scripts o algún tipo de configuración. Los códigos y clases son creados por una herramienta de instanciación que oculta los detalles complejos al desarrollador. Este enfoque permite un menor conocimiento de la estructura interna del framework, lo que lo hace más fácil de usar.

Este concepto del framework, aplicado al dominio del diseño e implementación de laboratorios de acceso remoto, se ve reflejado en la capa de comunicación cliente/servidor requerida por estos laboratorios. En esta capa, el framework oculta la implementación del sistema que publica las funciones del equipo remoto sobre un servidor web y provee a la aplicación cliente las funciones necesarias para comunicarse con dicho servidor.

3. FRAMEWORK MULTINIVEL PARA LA CONSTRUCCIÓN DE LABORATORIOS DE ACCESO REMOTO

El ***F***ramework para la construcción de ***L***aboratorios de ***A***cceso ***R***emoto para la ***E***xperimentación en ***R***obótica, llamado de forma abreviada *FLARER*, provee la estructura para la implementación de las capas que debe tener un laboratorio de este tipo. Esta estructura está enmarcada en tres capas: Capa de conexión, capa

web y capa cliente. La figura 2, muestra la arquitectura general del Framework. A continuación se describe la estructura que provee cada capa para la implementación de un laboratorio de acceso remoto.

3.1 Capa de conexión

Es la capa inferior del Framework que maneja las conexiones a puertos externos. En esta capa se realiza la conexión al equipo real a través de una interfaz llamada *IPortConnection*, la cual contiene las operaciones de comunicación que se deben implementar para acceder al puerto donde se conecta el equipo. En esta capa se deben construir las funciones que provee el equipo remoto, las cuales serán publicadas en la web y accedidas a través de peticiones HTTP. También cuenta con funciones para la inicialización de la conexión al puerto especificado, para la finalización de la conexión y cierre de los *streams* necesarios (ejemplo, *streams* de entrada y salida). Así mismo, para enviar y leer un dato en el puerto o conexión y retornar el resultado como un objeto genérico.

Para los proyectos en donde se implemente esta interface, se recomienda que la clase concreta implemente su funcionalidad mediante un esquema *Singleton* (Horstmann, 2006); esto para permitir una única instancia de conexión al puerto y evitar errores por puertos ocupados. Lo cual se debe tener en cuenta si el puerto al que se desea conectar no soporta múltiples conexiones instantáneas. Un ejemplo es una conexión al puerto serial RS-232 que se encontrará conectado el equipo para la experimentación remota. Para este caso el desarrollador deberá construir las funciones que enviarán y/o recibirán los datos de control y monitoreo para el equipo.

3.2 Capa Web

Una vez se haya desarrollado la capa de conexión y se hayan diseñado y construido las funciones para el control del equipo, se debe instanciar la clase que contiene dichas funciones en la capa web para permitir el acceso a éstas a través de peticiones HTTP.

El Framework FLARER provee el sistema de publicación de estas funciones a través de un *Servlet* de manera automática, es decir, permite el acceso público a las funciones implementadas en la capa de conexión a través de peticiones GET o POST por medio del protocolo HTTP. Bajo este protocolo, el equipo puede ser accedido a través del puerto 80, evitando así posibles bloqueos generados por los firewalls de la red.

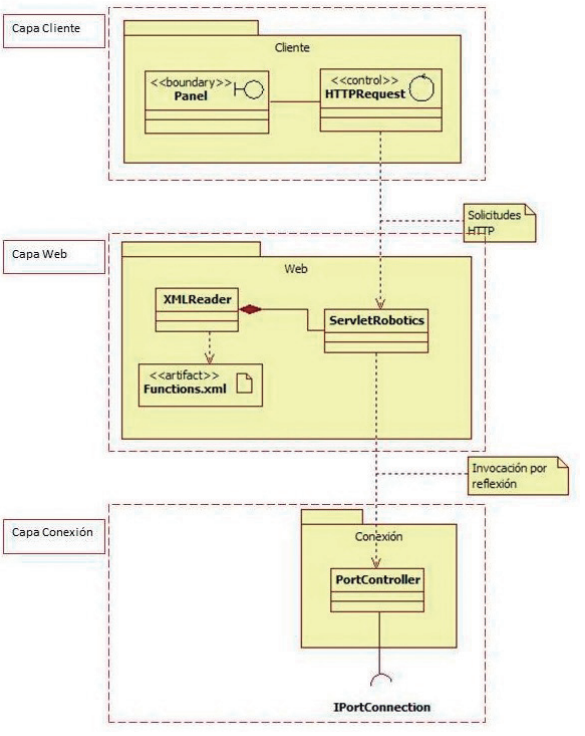

Figura 2. *Arquitectura del Framework FLARER*

En el enlace entre las peticiones HTTP y las funciones implementadas en la capa de conexión, se logra por medio de la *API reflection* de JAVA (Java, 2010), que permite la invocación de las funciones implementadas en la capa de conexión sin necesidad de tener una instancia directa de la clase que las implementa. Esta relación se logra a través de la lectura de un archivo XML (*Extensible Markup Language*), el cual contiene la referencia de las funciones que se desean publicar y sus respectivos parámetros de invocación. Este archivo se debe adaptar a un esquema de reglas específicas para definir su estructura y los parámetros de entrada que estas necesitan. La figura 3 muestra un ejemplo de un archivo XML.

```
<?xml version="1.0" encoding="UTF-8"?>
<server xmlns="http://direccionservidor/robotica/server-scheme">
        <functions reference="robotics.rv2aj.serialcontroller.core.RV2AJ">
                <function name="start" />
                <function name="close" />
                <function name="move">
                        <param name="speed" type="java.lang.Float" />
                        <param name="inc" type="java.lang.Float" />
                        <param name="axis" type="java.lang.String" />
                </function>
        </functions>
</server>
```

Figura 3. *Formato archivo XML*

En la figura 3, la etiqueta *functions* contiene el atributo *reference*, el cual es quien especifica la instancia de la clase que contendrá las funciones del equipo remoto

(*robotics.rv2aj.serialcontroller.core.RV2AJ*). Para este ejemplo se han definido tres funciones: *start*, *close* y *move*. La función *move* tiene tres parámetros de entrada: *speed*, *inc* y *axis*. Con esta especificación, se puede instanciar la clase referenciada por el XML, de manera que es posible acceder a estas a través de una petición HTTP. Un ejemplo de esto sería la siguiente solicitud: *http://direccionservidor:8080/Servlet?function=move &speed=50&inc=10&axis=Z*

En esta petición se está accediendo a la función *move* y se están estableciendo los parámetros *speed*=50, *inc*=10 y *axis*=Z.

El procesamiento de este archivo XML, es realizado a través de la API JAXB (JAXB, 2012) (Network, 2005), en la que se mapean las etiquetas XML en objetos JAVA, para ser enviados a la capa de conexión a través del sistema de reflexión de JAVA.

3.3 Capa Cliente

En esta capa se debe desarrollar la interfaz gráfica que permita a los usuarios interactuar con el equipo remoto. Esta interfaz será implementada por el desarrollador de la aplicación, que deberá especificar los requerimientos de la misma.

Para enviar los comandos generados desde esta interfaz hacia el servidor web del equipo remoto, el Framework provee un sistema de comunicaciones para el envío estándar de dichos comandos a través de peticiones HTTP. Este sistema de comunicación provee una clase que permite invocar las funciones públicas del servidor a través de la generación de una URL que concatena el nombre de la función y los parámetros que esta requiere, con el objetivo de abrir una conexión HTTP hacia el servidor a través del método GET, por la cual se envía dicha URL y se abre un *stream* de entrada para capturar la respuesta del servidor. Esta respuesta puede ser, tanto una cadena como un objeto. Este proceso es realizado por el Framework, ocultando al desarrollador la inicialización de la conexión HTTP y las implementaciones de los stream de comunicación de JAVA para la conexión con el servidor.

La tabla 1 presenta un resumen de las principales funcionalidades del Framework. Además en esta tabla se especifica lo que ofrece el Framework y lo que el desarrollador debe implementar para el diseño de un laboratorio de acceso remoto.

Tabla 1. *Tabla de funcionalidades del Framework*

Capa	Funcionalidad del Framework	A realizar por el desarrollador
Capa de Conexión	Suministra una interfaz para conexión con puertos externos: inicialización y finalización de la conexión, control de steams y de datos (objetos).	Diseñar e implementar las funciones propias para el acceso y control del equipo usando la interfaz de conexión propuestas por el Framework (*IPortConnection*).
Capa Web	Publica las funciones de la capa de conexión de manera automática, permitiendo el acceso a ellas a través de peticiones HTTP.	Instanciar las clases que contiene las funciones para el acceso y control del equipo, a través de la declaración de ellas en un formato XML.
Capa Cliente	Sistema de comunicaciones para el envío de comandos desde la interfaz del usuario a través de peticiones HTTP hacia la capa Web.	Construir la interfaz de usuario para el acceso y el control del equipo remoto. Esta interfaz debe usar el sistema de envío de comandos a través de peticiones HTTP para la comunicación la capa Web.

4. VALIDACIÓN DEL FRAMEWORK

Con el objetivo de validar la funcionalidad del Framework FLARER, se instancia un laboratorio de acceso remoto para el robot industrial Mitsubishi RV-2AJ. Este laboratorio se encuentra en el grupo de investigación SINFOCI de la Universidad del Quindío en la Facultad de Ingeniería.

A continuación se presenta el diseño y la implementación de cada una de las capas (conexión, web y cliente) del laboratorio utilizando la metodología propuesta por el Framework. Así mismo se presentan las pruebas de desempeño y funcionalidad realizadas al laboratorio.

4.1 Análisis de requerimientos

Para el diseño del laboratorio de acceso remoto para el robot Mitsubishi RV-2AJ, inicialmente se plantean los requerimientos del sistema:

Funcionales:

- Proveer acceso vía Web al laboratorio del robot Mitsubishi RV-2AJ.
- Permitir el monitoreo y control de las funcionales que ofrece el robot Mitsubishi RV-2AJ.
- Visualizar en tiempo real el robot y su ambiente de trabajo.

Usabilidad:
- Las interfaces de usuario se desarrollan de forma consistente con las funcionalidades de operación que ofrece el robot (grados de libertad, incrementos, velocidades, entre otras).

Disponibilidad:
- El laboratorio de acceso remoto estará disponible las 24 horas del día los 7 días de la semana, siempre y cuando existe suministro de energía eléctrico y disponibilidad de red lógica en la Universidad del Quindío.

Carga y Concurrencia
- El laboratorio permitirá la conexión de hasta 10 usuarios en forma concurrente, donde un usuario será quien controle el robot y los otros serán usuarios visualizadores.

Rendimiento:
- El video del laboratorio debe garantizar una velocidad de transmisión de imágenes de 10 fps (frames por segundos) con una conexión de red de al menos de 500 Kbytes/s.

Compatibilidad:
- El laboratorio podrá ser accedido a través de cualquier navegador web estándar para un computador personal y bajo cualquier sistema operativo que soporte aplicaciones desarrolladas en JAVA.

Restricciones de diseño:
- El laboratorio de acceso remoto se deberá desarrollar a través del lenguaje de programación JAVA.

Interfaces de comunicación:
- Las solicitudes cliente/servidor se deben realizar a través de peticiones HTTP.
- El acceso al robot Mitsubishi RV-2AJ se debe realizar a través de una interfaz serial, que comunica al servidor con el controlador el robot (puerto serial RS-232).

Interfaces de usuario:
- El laboratorio de acceso remoto suministrará interfaces de usuario que permitirán reconocer, identificar, monitorear, operar y controlar el robot Mitsubishi RV-2AJ. Con estas interfaces se realizarán prácticas de laboratorio que permiten realizar movimientos del robot a través de los ejes y sus coordenadas, además del control de la pinza y la visualización en tiempo real. Adicionalmente, los usuarios podrán crear, editar y ejecutar archivos de posiciones y de programación en el lenguaje MELFA BASIC IV, para la realizar tareas específicas con el robot, que finalmente se convertirán en proyectos que serán ejecutados en el controlador del robot.

4.2 Estructura del laboratorio de acceso remoto

El laboratorio de acceso remoto está compuesto por el robot Mitsubishi RV-2AJ, un servidor, un esquema de comunicación y los usuarios. Dada la complejidad y dinámica del sistema, se define la infraestructura de comunicación para la transmisión eficiente de información, que permita enlazar a los usuarios con el robot. Así mismo, esta estructura de comunicación debe integrar estos componentes, garantizando, además de la confiabilidad del sistema, el transporte de información sin pérdidas ni interrupciones entre los usuarios y el robot. La figura 4 muestra los componentes y la estructura general del sistema de experimentación remota.

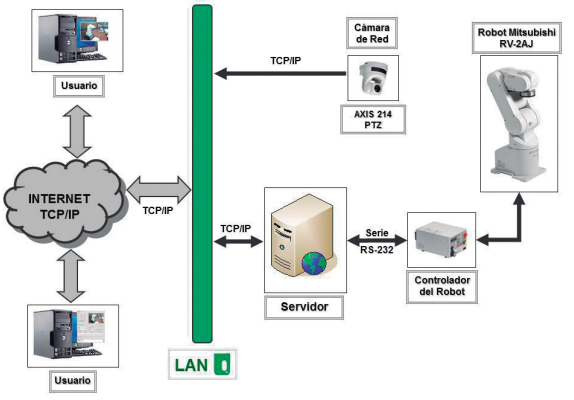

Figura 4. *Infraestructura de comunicación del laboratorio de acceso remoto para el robot Mitsubishi RV-2AJ*

El Robot Mitsubishi RV-2AJ es un robot manipulador industrial que tiene 5 grados de libertad con una articulación de tipo antropomórfico (Mitsubishi, 2011). Este robot viene equipado con el controlador Mitsubishi CR1-571, el cual es la interfaz de comunicación con el servidor del sistema a través de un enlace serie RS-232 (Robots, 2005), permitiendo que el usuario envíe y reciba información hacia y desde el robot y el servidor.

Según la propuesta del Framework, la infraestructura de comunicación para el laboratorio tendrá una arquitectura cliente/servidor de tres niveles, conformada por el robot (controlador del robot), el servidor que actúa como nivel intermediario y la aplicación cliente (usuario).

4.3 Diseño del Laboratorio de Acceso Remoto para el Robot Mitsubishi RV-2AJ

4.3.1 Diseño de la capa de conexión

Siguiendo la metodología que proporciona el Framework, primero se implementa la capa de conexión del robot. Esta capa se construyó a través de dos *APIs*: La primera de ellas que actúa sobre el puerto serie RS-232 para enviar y recibir datos a través de *javacomm* (*Java Communications API,* (Oracle, 2000)), llamada *serial-controller*. Esta API provee la configuración de la inicialización del puerto y los métodos de lectura y escritura específicos para la comunicación con el controlador del robot. La segunda API*,* llamada *rv2aj-controller,* contiene la implementación de las funciones para el control y monitoreo del robot Mitsubishi RV-2AJ. Estas funciones son: Inicializar de la comunicación con el robot, Mover el robot por medio de sus ejes o sus coordenadas, Controlar la pinza, Gestionar programas y/o posiciones, y Monitorear la posición.

Estas dos APIs se encuentran enlazadas por medio de la interfaz *IPortConnection* proporcionada por el Framework, de manera que la implementación de las funciones del robot sea independiente de la comunicación con un puerto físico. La figura 5 muestra el diagrama de componentes de esta capa.

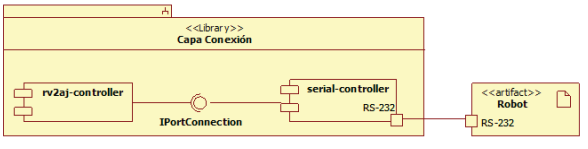

Figura 5. *Capa de conexión UQ-RobWeb*

4.3.2 Diseño de la capa Web

El siguiente paso es crear la aplicación Web, que publica las funciones de la capa de conexión. Para publicar estas funciones en el servidor web, se especifica el archivo de configuración XML, con las mismas funciones y parámetros especificados en la capa de conexión.

La especificación de este archivo se debe crear teniendo como referencia la capa de conexión. Por ejemplo, en la figura 6 (a) se muestra la implementación de la función *moveAxis* en la capa de conexión y en la figura 6 (b) su invocación en la capa Web, donde se observa la correspondencia que se debe mantener entre las dos capas. Esta configuración debe mantener los mismos tipos y parámetros de la capa de conexión, para así publicar de manera automática estas funciones.

```
public String moveAxis(Float speed, Float inc, String axis) {
…
…
…
}
```

a) Capa Conexión

```
<function name="moveAxis">
        <param name="speed" type="java.lang.Float" />
        <param name="inc" type="java.lang.Float" />
        <param name="axis" type="java.lang.String" />
</function>
```

b) Capa Web

Figura 6. *Función moveAxi. a) Capa Conexión. b) Capa Web*

La figura 7, muestra la configuración del archivo XML con todas las funciones que ofrece el laboratorio de acceso remoto del robot Mitsubishi RV-2AJ. Esta aplicación Web se implementó sobre un servidor compatible con JSP y servlets 2.0, para este caso se utilizó *Tomcat*.

```
<?xml version="1.0" encoding="UTF-8"?>
<server xmlns="http://direccionservidor/robotica/server-scheme">
        <functions reference="robotics.rv2aj.serialcontroller.core.RV2AJ">
                <function name="start" />
                <function name="end" />
                <function name="getRpos" />
                <function name="getJpos" />
                <function name="openHand" />
                <function name="closeHand" />
                <function name="moveAxis">
                        <param name="speed" type="java.lang.Float" />
                        <param name="inc" type="java.lang.Float" />
                        <param name="axis" type="java.lang.String" />
                </function>
                <function name="moveXYZ">
                        <param name="speed" type="java.lang.Float" />
                        <param name="inc" type="java.lang.Float" />
                        <param name="axis" type="java.lang.String" />
                </function>
                <function name="moveXYZs">
                        <param name="speed" type="java.lang.Float" />
                        <param name="X" type="java.lang.Float" />
                        <param name="Y" type="java.lang.Float" />
                        <param name="Z" type="java.lang.Float" />
                        <param name="A" type="java.lang.Float" />
                        <param name="B" type="java.lang.Float" />
                </function>
                <function name="moveAxes">
                        <param name="speed" type="java.lang.Float" />
                        <param name="J1" type="java.lang.Float" />
                        <param name="J2" type="java.lang.Float" />
                        <param name="J3" type="java.lang.Float" />
                        <param name="J5" type="java.lang.Float" />
                        <param name="J6" type="java.lang.Float" />
                </function>
        </functions>
</server>
```

Figura 7. *Configuración XML de Laboratorio de Acceso Remoto*

4.3.3 Diseño de la Capa Cliente

En la capa cliente, se diseñaron tres interfaces gráficas de usuario para el control y monitoreo del robot Mitsubishi RV-2AJ. Estas interfaces se construyeron a través *applet´s* de Java y contienen los controles para hacer el llamado a las funciones publicadas en la aplicación Web. Los controles realizan el llamado a las funciones Web por medio del sistema de comunicación HTTP que provee el Framework. El diseño de estas interfaces se realizó con el objetivo que el laboratorio será un complemento a las prácticas de un curso de Robótica Industrial.

La primera interfaz permite al usuario conocer e interactuar con el robot Mitsubishi RV-2AJ, conociendo y diferenciando los ejes, el espacio cartesiano y los componentes del robot. El objetivo de esta primera interfaz es introducir al usuario con la interacción de un laboratorio de acceso remoto de robótica, además de conocer este tipo de plataformas educativas.

La segunda interfaz (figura 8) permite al usuario controlar y manipular el robot a través de los ejes o llevando a una coordenada específica en el espacio de trabajo, definiendo el incremento y la velocidad para cada uno de estos movimientos. Así mismo, se tiene el control de apertura y cierre de la pinza. Además, la interfaz suministra los eventos y acciones generados en el robot en una consola, entregando al usuario la información de la posición actual del robot con respecto a sus ejes y sus coordenadas.

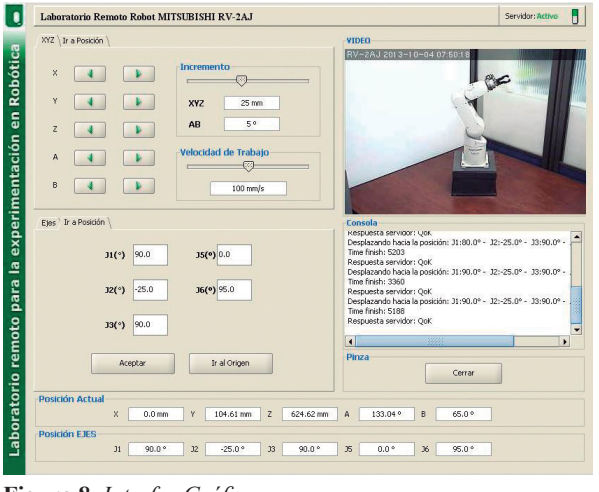

Figura 8. *Interfaz Gráfica*

La ultima interfaz, permitirá a los usuarios crear, editar y ejecutar archivos de posiciones y de programación en el lenguaje MELFA BASIC IV (lenguaje propietario del robot), para la realización de tareas específicas, que finalmente se convertirán en proyectos que serán ejecutados por el controlador del robot.

Para la realimentación visual de las interfaces del laboratorio de acceso remoto se utiliza un *stream* de video MJPG generado por la cámara de red Axis 214 PTZ que actúa como un servidor de video. La comunicación entre los usuarios y el servidor de video utiliza una comunicación TCP/IP a través del puerto 80. El acceso al *stream* se realiza directamente desde el *applet* a dicho servidor por medio de una petición HTTP al iniciar la aplicación. Por medio de esta cámara se capturan imágenes a una resolución de 352x240 píxeles, en formato *Motion JPEG* y a una frecuencia de imagen entre 10 y 15 fps.

5. PRUEBAS Y RESULTADOS

Esta sección presenta una síntesis de los resultados de las principales pruebas realizadas al laboratorio de acceso remoto para el robot Mitsubishi RV-2AJ. El objetivo de estas pruebas es verificar la funcionalidad y el comportamiento del sistema implementado con el Framework. Las pruebas se realizaron sobre el servidor del laboratorio de acceso remoto y en el usuario sobre la interfaz gráfica.

5.1 Servidor

Las pruebas al servidor pretenden evaluar su comportamiento al atender a uno o más usuarios simultáneamente. Se analiza el uso del procesador del equipo que alberga al servidor, el uso de memoria y el uso de los enlaces de comunicación, específicamente de la tarjeta de red. La prueba inicia sin usuarios conectados al servidor. Después de cierto tiempo accede un usuario que tendrá el control de robot manipulador. Un tiempo después empiezan a ingresar, los usuarios invitados (observación) hasta completar un total de ocho usuarios enlazados con el servidor. Cuatro de los usuarios se encuentran ubicados en diferentes equipos conectados a la red de la Universidad y tres más se encuentran por fuera de la LAN de la Universidad y un último usuario desde la Universidad del Valle.

Procesador: El uso del procesador durante la prueba fue bajo, donde su máximo valor de uso fue del 10%. Inclusive cuando se tienen más de 8 usuarios conectados al sistema no supera este valor. Esto indica que el servidor no carga de manera significativa al procesador cuando se atienden las peticiones.

Memoria: Es un indicador de la memoria física disponible para procesamiento en el equipo de cómputo que alberga al servidor. En nuestro caso el servidor tiene una memoria física de 512 Mbytes. Al igual que con los resultados del porcentaje de uso del procesador, la memoria disponible en el servidor no se ve afectada cuando se conectan más de 8 usuarios. Realizando un análisis a esta prueba de encontró que la desviación estándar es menor a 2 Mbytes, lo que indica que la disponibilidad de memoria no se afectada.

Tarjeta de red: La tarjeta de red es el dispositivo que permite que el servidor se enlace a la red de datos donde se conecta el servidor para prestar los servicios del

laboratorio de acceso remoto. Se evalúan los paquetes enviados y recibidos durante el tiempo que dura la prueba. Los resultados mostraron que cuando se conecta el primer usuario al sistema se presenta el pico más alto en las ratas de transmisión. Esto se debe a que cuando el cliente se enlaza al sistema solicita al servidor el *applet*. Los siguientes usuarios que conectan al sistema son observadores y por lo tanto la información que transmite hacia ellos es solo la realimentación visual y es enlazada a través del servidor y suministrada por el servidor de video (cámara de red). Esta prueba presentó un promedio de 2 Kbytes tanto en la transmisión y la recepción de paquetes.

Pruebas de carga y esfuerzo. De las más importantes pruebas que se deben llevar a cabo sobre servidores son la prueba de carga y de esfuerzo (o de estrés). La mayoría de problemas de rendimiento solo se producen cuando el servidor se ataca con una alta carga de usuarios. Por lo anterior, se hace necesario ejecutar este tipo de pruebas. El objetivo de estas pruebas es estimar cuántos usuarios concurrentes puede atender el servidor sin problemas.

La estimación que se realizó al servidor fue una prueba con 20 usuarios simultáneos accediendo en un 1 segundo, repitiendo esta solicitud 10 veces. Esta prueba representa 200 muestras. Durante esta prueba se analizó el tiempo de respuesta del servidor. Estos tiempos mostraron que la media es pequeña, lo que indica que los tiempos de respuesta del servidor a las peticiones de los usuarios son constantes. Así mismo se encontró que la desviación estándar es baja, lo que nos indica que la respuesta de este servidor no presenta variaciones altas entre un usuario y otro. También se confirma que el servidor atendió a todos las peticione y en ningún caso presentó errores.

5.2 Usuarios

Retraso Temporal. Esta prueba tiene como objetivo observar la continuidad en la transmisión de video y el comportamiento del retraso temporal cuando se accede al sistema de experimentación desde diferentes ubicaciones y empleando diferentes tipos de enlace (redes académicas y comerciales). Para ello, diferentes usuarios (uno a la vez y en diferentes horarios) se enlazan al laboratorio de acceso remoto y aceden a controlar el robot durante un tiempo aproximado de 10 minutos. Durante este tiempo se evalúa la rata de actualización de imágenes del video (frames por segundo - fps) y los retrasos temporales (máximo, mínimo y promedio) observados por el usuario.

Primero se realizó una prueba con un usuario conectando al laboratorio desde un equipo ubicado dentro de la red interna de la Universidad. Se encontró que el promedio del retraso temporal es alrededor de 60 ms y la transmisión de video se mantiene en 10 fps. La segunda conexión se realizó utilizando la red académica RENATA desde la Universidad del Valle. Se observó que el retardo promedio en esta conexión es menor que la primera prueba con un promedio de 30 ms y una oscilación en la trasmisión de video entre 12 y 15 fps. Esto evidencia un mejor comportamiento de sistema cuando se utiliza esta red académica. Finalmente se realizaron pruebas estableciendo enlaces con usuarios ubicados en España, utilizando la red académica CLARA (la cual se enlaza con la red RENATA por un canal de 45 Mbps). En los dos casos la recepción del video se mantiene casi constante, oscilando entre los 7 y 10 fps y el retraso promedio es menor a 500 ms.

5.3 Experiencias Académicas

El laboratorio de acceso remoto el robot Mitsubishi RV-2AJ es una herramienta de apoyo para el componente práctico de cursos de robótica industrial de cualquier institución, donde los estudiantes realizarían sus prácticas a través de este laboratorio. El laboratorio propone tres interfaces de usuario, las cuales fueron descritas anteriormente y complementan las guías de laboratorio para su realización. El objetivo final es adquirir las competencias necesarias para controlar y programar un robot industrial.

El laboratorio de acceso remoto ha venido siendo utilizado por estudiantes de pregrado de la Universidad del Quindío, de la Universidad Valle y de la Universidad del Cauca, en sus respectivos espacios académicos de robótica industrial de cada universidad. Esta experiencia fue un primer contacto de los estudiantes con un laboratorio remoto de acceso remoto. El desarrollo de las prácticas fue exitosa, tanto a nivel de funcionalidad, comunicación y disponibilidad del laboratorio. Además de lo anterior, se identificaron algunos aspectos a tener en cuenta para el mejoramiento del uso del laboratorio de acceso remoto:

- Se hace necesario realizar una primera visita al laboratorio real, para que el estudiante conozca el robot manipulador e identifique su espacio y dimensiones reales. Además de generar confianza y credibilidad del control remoto del robot.
- El laboratorio debe proporcionar más información visual y sensorial. Es decir, se hace necesario mostrar otras vistas del laboratorio.

- Se debe mejorar la iluminación y el contraste del robot con respecto a su entorno. Esto es debido al color del espacio del laboratorio y color del robot.

6. CONCLUSIONES

Se consolidó un Framework que provee la arquitectura de comunicación para la construcción de laboratorios de acceso remoto para la experimentación en robótica y además define la estructura de cómo se deben construir las capas de las aplicaciones que se integren a él. Por medio de este Framework se busca un punto de equilibrio entre simplicidad y escalabilidad, ya que facilita la comunicación entre las capas del sistema y permite la incorporación de nuevas funciones de manera rápida y de bajo impacto para la arquitectura software.

El Framework *FLARER* provee las capas para controlar peticiones HTTP y flujos de entrada y salida cliente/servidor de manera transparente para la aplicación final, como se puede presentó en el diseño y la implementación del laboratorio de acceso remoto para el robot Mitsubishi RV-2AJ.

La estructura de comunicación implementada para el laboratorio de acceso remoto, presenta un enlace de comunicación persistente entre los usuarios y el controlador del robot, logrando evitar los bloqueos generados por los firewalls de red, permitiendo que el robot pueda ser accedido desde cualquier punto de Internet. Así mismo, esta arquitectura permite alcanzar niveles de transparencia, modularidad y extensibilidad que facilitan la adición y puesta en funcionamiento de nuevos componentes.

Por medio de la implementación del laboratorio de acceso remoto para el robot Mitsubishi RV-2AJ, se demostró cómo este Framework proporciona los recursos necesarios para la puesta en marcha de una aplicación con los requisitos principales de un laboratorio de acceso remoto para la educación en ingeniería.

Los resultados de las pruebas realizadas a los componentes del laboratorio de acceso remoto confirman que el sistema de experimentación se puede poner al servicio de la comunidad educativa y es de gran utilidad como una herramienta de aprendizaje para cursos de robótica.

Las características de portabilidad y las herramientas del lenguaje de programación Java, como los *applet´s* y los *servlets*, lo hacen idóneo para la implementación de aplicaciones Web interactivas. A partir de los *applet´s* es posible crear aplicaciones con interfaces gráficas altamente dinámicas, capaces de ejecutarse en navegadores Web. Usando *servlets* es posible crear aplicaciones que se ejecuten del lado del servidor, con capacidad de atender peticiones HTTP de manera robusta y confiable a través de conexiones TCP/IP, además de soportar todas las funcionalidades de Java, como por ejemplo enlazarse a bases de datos o ejecutar métodos remotos empleando la tecnología RMI, entre otros.

7. REFERENCIAS

BARNABY, Dalton. *Techniques for Web Telerobotics.* Australia : Department of Mechanical and Materials Engineering University of Western Australia, 2001.

BRADY, K. y Tarn, T. J. *Internet-Based Remote Teleoperation.* Lexington, MA, USA : Proceedings IEEE International, 1998. págs. 65-70.

BUITRAGO, Jaime A. *Interfaz Remota para la experimentación en Robótica de Manipuladores.* Universidad del Valle. Cali : s.n., 2010. Tesis de Maestría.

CAICEDO, Eduardo., Buitrago, Jaime. y Calvache, Bayron. *Laboratorio Distribuido con acceso remoto para la enseñanza de la Robótica.* Bogotá D.C. : REVISTA EDUCACIÓN EN INGENIERÍA, 2009. págs. 51-61. 1900-8260.

CALVACHE, Bayron A. *Diseño e implementación de una interfaz remota para la experimentación en robótica móvil.* Universidad del Valle. Cali : s.n., 2009. Tesis Maestría.

HORSTMANN, Cay S. *Object-Oriented Design and Patterns.* s.l. : John Wiley & Sons, 2006.
Java, Oracle. 2010. The Reflection API. [En línea] 2010. http://docs.oracle.com/javase/tutorial/reflect/index.html.

JAXB, Project. 2012. *JAXB Reference Implementation Project.* . [En línea] Marzo de 2012. URL: http://jaxb.java.net/.

MARKIEWICZ, Marcus E. y Lucena., Carlos J.P. de. 2000. *Understanding Object-Oriented Framework Engineering.* Pontifical Catholic University of Rio de Janeiro. Rio de Janeiro : s.n., 2000. Articulo Interno.

MITSUBISHI. Robots Mitsubishi. *Mitsubishi Robot Manuals RV-2AJ.* [En línea] 2011. http://www.mitsubishirobots.com/manuals.html.

MORISIO, Maurizio, Romano, D. y Moiso, C.,. *Framework based software development: investigating the learning effect.* Boca Raton, FL : s.n., 1999. págs. 260-268.

Network, Oracle Technology. Java Architecture for XML Binding. [En línea] Oracle, 2005. http://docs.oracle.com/javase/tutorial/jaxb/index.html.

NUÑO, O. E. y Basañez, L. Teleoperación: técnicas, aplicaciones, entorno sensorial y teleoperación inteligente. [En línea] 2004. http://bibliotecnica.upc.es/reports/ioc/IOC-DT-P-2004-05.pdf.

RIEHLE, Dirk. *Framework Design, A Role Modeling Approach.* Universidad de Hamburgo. Hamburgo : s.n., 2000. Doctor of Technical Science.

ROBOTS, Mitsubishi. Controller CR1-571. [En línea] 2005. http://www.roboex.com/.

RobUALab. 2008. Servidor Web del Laboratorio Virtual RobUALab.Ejs. [En línea] Universidad de Alicante, 2008. http://robualab.eps.ua.es/.

RODRIGUEZ, F., y otros. *A Remote Laboratory for Teaching Mobile Robotics.* Alemania : s.n., 2001.

ROSADO, L. y Herreros, J. R. *Nuevas aportaciones didácticas de los laboratorios virtuales y remotos en la enseñanza de la Física.* Sevilla-España : s.n., 2006.

TELELABS, The UWA. 2009. *The Telelabs Project* . [En línea] Enero de 2009. http://telerobot.mech.uwa.edu.au/.

8. CURRÍCULUM

Jaime Alberto Buitrago. Docente del Programa de Ingeniería Electrónica e Investigador de del Grupo de Investigación SINFOCI de la Universidad del Quindío, Carrera 15 Calle 12 Norte. Áreas de interés: Laboratorios de Acceso Remoto, Robótica, Educación en Ingeniería, Arquitectura de Procesadores.

Julian Alejandro Lamprea. Docente del Programa de Ingeniería de Sistemas y Computación e Investigador de del Grupo de Investigación SINFOCI de la Universidad del Quindío, Carrera 15 Calle 12 Norte. Áreas de interés: Desarrollo de aplicaciones Web, Interfaces Visuales, Computación Ubicua.

Eduardo Caicedo-Bravo. Profesor titular de la Universidad del Valle, Cali (Colombia) y Director del Grupo de Investigación Percepción y Sistemas Inteligentes – PSI, Ciudad Universitaria Meléndez Edificio 354. Áreas de interés: Instrumentación Electrónica, Educación en Ingeniería, Inteligencia Artificial, Robótica.