ENTORNO PARA PROGRAMACIÓN REMOTA DE ROBOTS MANIPULADORES EN ACL CON REALIMENTACIÓN VISUAL

J.L. Guzmán, J.F. Sarabia, F. Rodríguez, J.C. Moreno, M. Berenguel
Universidad de Almería. Dpto. de Lenguajes y Computación. Área de Ingeniería de Sistemas y Automática.
Ctra. Sacramento s/n, 04120 Almería, Tel: +34 950 015683, Fax: +34 950 015129,
Correo electrónico: {joguzman,frrodrig,jcmoreno,beren}@ual.es

Resumen

Este trabajo describe una herramienta software basada en la arquitectura cliente-servidor para la programación y ejecución remota de programas escritos en ACL (Advanced Control Language), permitiendo a los usuarios realizar sus propias aplicaciones en este lenguaje, compilarlas y probarlas. El entorno de ejecución consiste en una célula robotizada compuesta por un robot manipulador Scorbot ER V-Plus, un conjunto de sensores y una cinta transportadora. La principal contribución respecto a otras aplicaciones similares es el desarrollo e implantación de un compilador de ACL. El usuario puede seguir la evolución de la ejecución de sus programas gracias a dos cámaras instaladas en la célula robotizada.

Palabras Clave: ACL, TCP/IP, Web, Robot manipulador, Java, C++.

1 INTRODUCCIÓN

En los últimos años se ha experimentado un considerable avance en el ámbito del desarrollo de laboratorios virtuales y remotos para la práctica a distancia de la Automática [2,4]. Una de las consecuencias directas de este avance es el aumento de la flexibilidad, en términos temporales y espaciales, de acceso a los laboratorios docentes, así como permitir una rápida adaptación de las prácticas de laboratorio a los avances tecnológicos y de innovación que aparecen en el mundo de la ingeniería [9]. Las consideraciones anteriores tienen además especial significado en el ámbito de la robótica, pues constituyen sistemas que, por una parte son susceptibles de ser teleactuados y, por otra, suelen tener un precio muy elevado que impide su implantación masiva en los laboratorios docentes y por tanto exige una planificación de uso que permita alcanzar la flexibilidad anteriormente indicada. Existen diversas aplicaciones de la telerobótica a través de Internet. Un compendio de las mismas se puede encontrar en [3]. Entre los trabajos más relevantes relacionados con el propuesto en este trabajo, cabe destacar el desarrollado por el grupo de la Universidad de Alicante ROBOLAB [11]. Este trabajo permite al usuario teleoperar un brazo robótico SCORBOT ER-IX. Mediante simulación 3D (Java y VRML) se analizan los movimientos del robot para posteriormente ejecutarlos sobre el sistema real. Tras la simulación se presenta un vídeo con el resultado de la ejecución sobre el robot. En tiempo real lo que se obtiene es una realimentación de los movimientos del brazo robot mediante realimentación por simulación. En [10] se presenta una aplicación que permite mediante un applet Java enviar secuencias de comandos a un robot ASEA Irb-6. [7] muestra una aplicación sobre el robot Adept 550 SCARA, donde tras una simulación 3D se envían comandos al robot real. También incorporan una botonera activa e imágenes de cuatro cámaras.

El trabajo se estructura como sigue: en la sección 2 se justifica la necesidad de desarrollo e implantación de un compilador para ACL y se introducen las características básicas del mismo. La sección 3 incluye una breve descripción de la célula robotizada sobre la que se ha implantado el sistema. La sección 4 muestra la arquitectura propuesta del sistema y en la sección 5 se explica su funcionamiento, incluyendo un ejemplo ilustrativo. En la sección 6 se resumen los trabajos futuros y las principales conclusiones de este trabajo.

2 COMPILADOR PARA ACL

ACL (*Advanced Control Language*), es un lenguaje de programación avanzado para robótica desarrollado por *ESHED ROBOTEC Ltd.* (1982) [8]. Sus principales características y prestaciones son:

- Ejecución directa de comandos del robot.
- Control de entrada y salida de datos.
- Programación de usuario del robot.
- Ejecución simultánea de programas.
- Ejecución sincronizada de programas.
- Gestión sencilla de archivos.

Las primitivas de ACL se encuentran almacenadas en una EPROM dentro del controlador del robot y se puede acceder a ellas desde cualquier PC

estándar o terminal, por medio de una línea RS232 de comunicaciones. A la hora de desarrollar un programa en ACL es necesario comprobar su corrección desde un punto de vista léxico, sintáctico y semántico. Para ello existen dos aplicaciones que acompañan al controlador:

- `ATS` (*Advanced Terminal Software*) que aporta un entorno de usuario que permite escribir un programa ACL, comprobar su corrección y cargarlo en el controlador para su ejecución.
- `ACL Off-line`, que sólo permite interactuar con el controlador, enviando el programa, que previamente ha sido realizado en un editor de texto estándar, y recibiendo los errores de compilación detectados por el compilador sito en el controlador del robot.

En ambas aplicaciones, en primer lugar se envía el programa por el puerto serie, posteriormente se compila en el controlador, y finalmente se devuelven los mensajes oportunos utilizando el mismo puerto, mensajes que informan al usuario de los posibles errores de compilación o del éxito de la operación. De este modo, cada vez que el usuario tenga un error, debe corregir el programa, guardarlo, reenviarlo por el puerto, esperar a que sea compilado en el controlador y finalmente a que se le envíe el resultado de la compilación.

La limitación inherente a ambas aplicaciones radica en la necesidad de utilizar el PC conectado al controlador tanto para compilar el programa escrito en ACL como para poner el programa en ejecución, utilizando para ambos propósitos la línea de comunicaciones RS-232. En cuanto a la labor docente se refiere, esto conduce a dos alternativas a la hora de organizar sesiones en las que se pretende que el alumno tome contacto con un lenguaje de programación de robots como ACL. La primera consiste en la utilización de simuladores que permitan a los alumnos probar sus programas sin necesidad de utilizar el robot [5]; la segunda consiste en organizar a los asistentes en grupos de forma que sea posible planificar adecuadamente el uso del robot. En cualquier caso, ambas alternativas conducen a una falta de contacto del alumno con el robot real. Como paso previo al planteamiento de una solución a este problema, se ha desarrollado un compilador en ANSI C. Este compilador permite evaluar la corrección de programas escritos en ACL sin necesidad de estar conectados al controlador del robot.

2.1. DESCRIPCIÓN DEL COMPILADOR

El lenguaje ACL consta de un repertorio aproximado de unas 120 instrucciones y, de todas ellas, actualmente el compilador reconoce aproximadamente unas 50 instrucciones que permiten realizar un conjunto de tareas básico (desplazamiento del robot, manejo de E/S, concurrencia,...).

El compilador permite obtener un programa inteligible por el controlador del robot a partir de un programa en ACL (ver [1] para un estudio detallado de los compiladores). En el caso particular de ACL y para la célula robotizada descrita en el apartado 3, el programa es enviado directamente al controlador en formato ASCII; internamente el controlador comprueba la corrección del programa y lo pone en ejecución. El objetivo en este caso es el desarrollo de un programa que permita asegurar que lo que se envía al controlador es correcto léxica (analizador léxico), sintáctica (analizador sintáctico) y semánticamente (analizador semántico).

Además, dos componentes adicionales son necesarios para realizar el compilador: la tabla de símbolos y el manejador de errores. La tabla de símbolos es la estructura de datos usada por el compilador para asociar a cada símbolo del programa fuente (identificadores, constantes, etc...) un contenido semántico. La información más común que se suele almacenar en una tabla de símbolos es el nombre del símbolo, la dirección de memoria, el tipo y los números de línea donde aparece. Para el caso particular que nos ocupa se ha creado una estructura en forma de lista enlazada donde cada nodo almacena: nombre del símbolo, el tipo de dato (en ACL hay dos: variables y posiciones), un valor booleano que indique si dicha variable o posición ha sido definida o no y el nombre del programa (vacío en el caso de ser global). El manejador de errores es la parte del compilador que se encarga de gestionar los errores encontrados durante el proceso de compilación. En esta implementación, cuando se produce un error, ya sea léxico, sintáctico o semántico, no se interrumpe el proceso de compilación. Con esto se trata de evitar que la tarea de compilación no sea tediosa para el programador y que sólo sea interrumpida cuando el número de errores supera la decena (en un esquema de acceso remoto al compilador parece la opción más adecuada, minimizando el número de envíos del programa para compilarlo cuando se detectan errores).

Analizador léxico: como primera parte de la compilación, lee los caracteres del programa fuente e identifica los componentes léxicos que posteriormente utilizará el analizador sintáctico. Cuando el analizador léxico detecta un identificador realiza una función extra que es la inserción o modificación del mismo en la tabla de símbolos. En el desarrollo del compilador de ACL cuando se detecta un identificador se busca en la tabla de símbolos y si no se encuentra se introduce. Para el desarrollo del analizador léxico se ha hecho uso de un autómata finito determinista que reconoce cualquier elemento perteneciente al repertorio *R* de

instrucciones reconocidas por el compilador además de símbolos de operación e identificadores.

```
R={"TEACH","TEACHR","PROGRAM","MOVED","MOVE
LD","MOVECD","MOVESD","OPEN","CLOSE","SPEED
","DEFP","DIMP","SETP","SET","DELP","GLOBAL
","DIMG","DIM","DELVAR","DEFINE","IF","ELSE
","ENDIF","END","ENDFOR","FOR","LABEL","GOT
O","DELAY","PEND","POST","QPEND","QPOST","R
UN","STOP","SHIFT","SHIFTC","WAIT","PRINT",
"PRINTLN","GOSUB","READ"}
```

Analizador sintáctico: realiza la segunda fase del proceso de compilación. Solicita los componentes léxicos al analizador léxico y comprueba si la cadena formada por dichos componentes pertenece a la gramática del lenguaje. Para la realización del analizador sintáctico se ha optado por un *análisis descendente recursivo* (método *top-down* en el que se ejecuta un conjunto de procedimientos recursivos para procesar la entrada). El analizador sintáctico a la vez que va comprobando que la estructura del programa se corresponde con la gramática del lenguaje interactúa con la tabla de símbolos donde va señalando aquellas variables y posiciones que han sido definidas, así como los correspondientes tipos. De igual forma cada vez que se detecta un error se informa al manejador de errores. En la figura 1 pueden observarse las distintas etapas y la interacción entre los distintos componentes en el proceso de compilación.

Analizador semántico: la fase de análisis semántico revisa el programa fuente para garantizar que no existan situaciones con significado ambiguo. El componente más importante dentro del análisis semántico es la verificación de tipos (de hecho es el único que se utiliza en el compilador desarrollado por la naturaleza del entorno de trabajo). Para verificar la compatibilidad de tipos, cada vez que se detecta una operación o comparación se accede a la tabla de símbolos para comprobar el tipo de cada operador, y en caso de incompatibilidad se informa del error al manejador de errores.

3 DESCRIPCIÓN DE LA CÉLULA ROBOTIZADA

En la figura 2 se muestra un esquema de la configuración de la célula robotizada, constituida por los siguientes elementos:

- Brazo mecánico. Manipulador robótico Scorbot ER V-Plus de Eshed Robotec Ltd., articulado con 5 grados de libertad (3 de posición y 2 en la muñeca para elevación y giro).
- Controlador multitarea en tiempo real, permite la ejecución simultánea de varios programas. Sus principales características se muestran en la Tabla 1.
- Botonera de enseñanza, que permite el control de las articulaciones, la definición de posiciones por medio del método de aprendizaje y la ejecución de programas que se encuentren en el controlador.

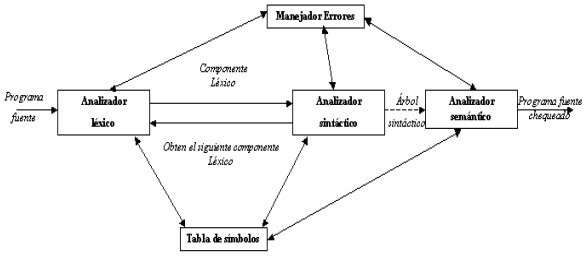

Figura 1: Componentes del compilador.

Tabla 1: Características técnicas del controlador

E/S	16 entradas (TTL, 12v ó 24v) 16 salidas (4 relé/12 colector abierto)
Comunicación	RS-232C
Programación	ACL y SCORBASE (Nivel 5)
CPU	Motorola 68010
Sistema coordenadas	X, Y, Z y articulaciones del robot Definiciones absolutas y relativas
Otras	Autónomo PID, en tiempo real, multitarea, PWM

- Cinta transportadora con motor AC controlado a través de un variador de velocidad conectado al controlador del robot, que gobierna la puesta en marcha, parada y velocidad de la cinta transportadora de forma digital o analógica.
- Sensores fotoeléctricos, para detectar la posición en que están ubicados los objetos en la cinta transportadora.

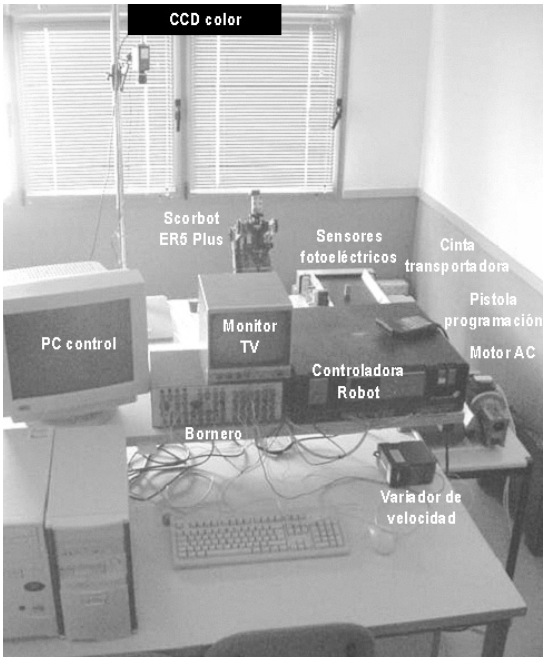

Figura 2: Configuración de la célula robotizada

- Sistema de visión, formado por una cámara CCD color y una tarjeta capturadora con *frame grabber* instalada en el computador de control. Este sistema se ha ampliado para este proyecto incluyendo un servidor de vídeo y otra cámara, como se comenta posteriormente.
- Computador (Pentium 166 Mhz), conectado al controlador del robot vía RS-232.

4 ARQUITECTURA DEL SISTEMA

El principal objetivo del sistema propuesto es permitir la realización de prácticas de robótica a través de Internet. Para ello se aporta un entorno de programación remoto en ACL que permite realizar un determinado programa en dicho lenguaje, compilarlo haciendo uso del compilador descrito en el apartado 2, e interactuar con la célula robotizada para comprobar su funcionamiento.

La arquitectura es la típica de un sistema cliente-servidor. La aplicación Cliente aporta los mecanismos necesarios para la realización de las diversas operaciones permitidas que serán comentadas en el apartado 5.1. El Servidor se encarga de recibir las peticiones enviadas por el cliente y en su caso interactuar con la célula robotizada. La comunicación entre Cliente/Servidor ha sido realizada haciendo uso de sockets TCP/IP. El esquema global de la arquitectura del sistema propuesto se muestra en la figura 3.

El Cliente se basa en un navegador cualquiera que permite conectarse a un centro web desde donde se aportan una serie de páginas HTML, un applet Java y dos controles ActiveX:

- Las páginas WEB muestran la interfaz gráfica que permitirá al usuario poder realizar las diferentes operaciones. Destaca un editor que permite escribir el programa en ACL y una serie de botones para realizar las acciones correspondientes (apartado 5.1.). Para la codificación de dichas páginas se utiliza el lenguaje JavaScript para enlazar con el applet Java y poder establecer la comunicación remota vía socket.
- La presencia del applet desarrollado en Java es transparente al usuario ya que únicamente es utilizado como puente para la comunicación remota con el servidor haciendo uso de los sockets Java.
- Los controles ActiveX permiten acceder al servidor de vídeo AXIS [6] que se encuentra situado en el lado del Servidor y que aporta imágenes de 2 cámaras CCD. Realmente los controles ActiveX adquieren las imágenes desde el servidor utilizando llamadas a *cgi's* que se encuentran implementados en el interior del servidor de video. Dichos cgi reciben una serie de parámetros que permiten configurar el tamaño, la compresión, el color y el refresco de la imagen; así como indicar cual de las cámaras del servidor se desea visualizar. Un ejemplo sería:

  http://videoserver/cgi-bin/fullsize.jpg?camera=1&compression=5&motion=0&color=2

En este ejemplo se está indicando que se desea el máximo tamaño de la imagen (*fullsize.jpg*), adquisición desde la cámara 1 (*camera=1*), compresión máxima (*compression=5*), refresco máximo (*motion=0*) e imagen en color (*color=2*).

El servidor está situado en el Laboratorio de Control Automático, Robótica y Visión Artificial de la Universidad de Almería, y está constituido por:

- Servidor de vídeo AXIS [6] que incorpora un servidor WEB que aporta una serie de controles ActiveX a través de los cuales es posible recuperar la imagen en tiempo de real de las cámaras conectadas al mismo (cuatro como máximo). Se encuentran conectadas 2 cámaras CCD que permiten una visualización desde diferentes enfoques de la célula robotizada.
- Servidor WEB Apache versión 1.3.24 que gestiona el centro web.
- Servidor de operaciones. Es una aplicación realizada en C++ Builder que se encuentra a la espera de peticiones generadas desde un navegador a través del applet Java y que está encargado de:
  a) Identificación de usuarios a través de un *login* y un *password* interactuando con una base de datos realizada en Paradox.
  b) Dar servicio a las diferentes operaciones de la aplicación como son la edición, compilación, etc. (comentadas en el apartado 5.1).
- Un PC (Pentium II a 233 Mhz con 192 MB de RAM y 2 GB de HDD) en el que se encuentran en ejecución el servidor Apache y el servidor de operaciones. Dicho computador se encuentra conectado con el controlador del robot a través de puerto RS-232.
- Célula robotizada detallada en la sección 3.

De esta forma un usuario podrá utilizar la célula robotizada haciendo uso de su programa en ACL desde cualquier lugar necesitando únicamente una conexión a Internet y un navegador.

5 IMPLEMENTACIÓN DEL SISTEMA

En el sistema desarrollado toda la carga de trabajo la realiza el servidor, siendo éste el encargado de almacenar los programas que realiza cada usuario, y de compilar y ejecutar estos programas. El cliente se encarga de enviar las peticiones al servidor y de mostrar a través de la interfaz de usuario los resultados de las operaciones que realiza el servidor a través de realimentación visual.

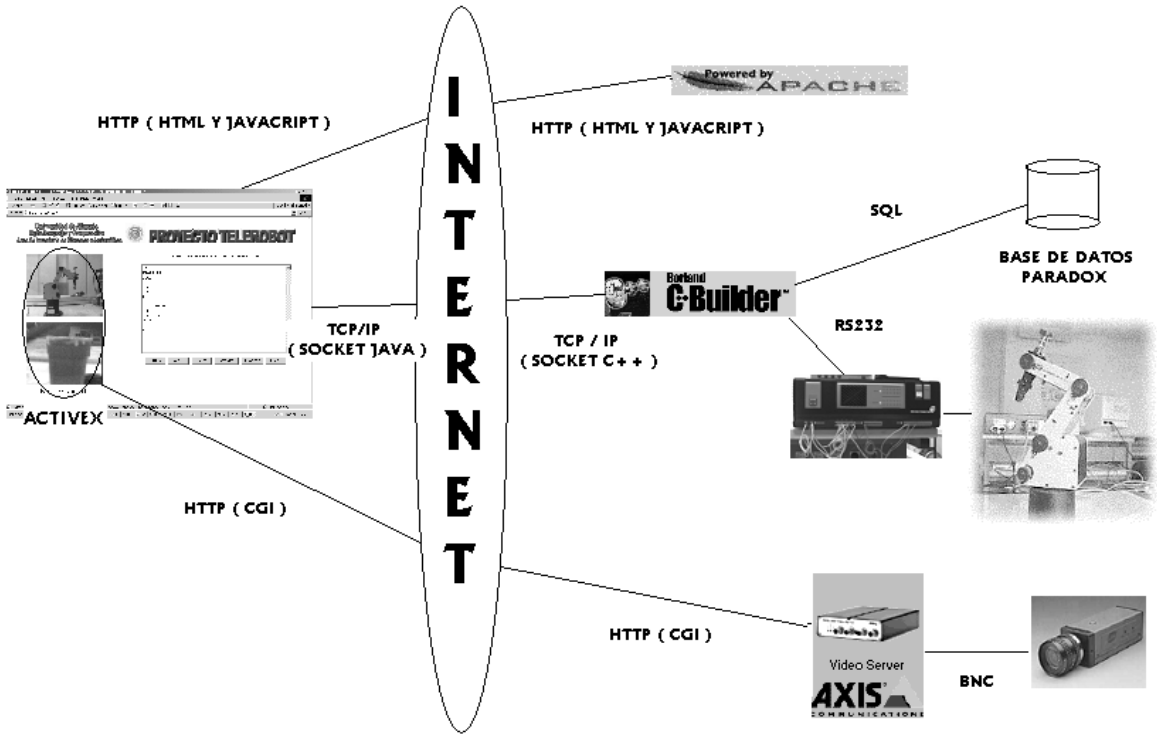

Figura 3. Arquitectura del Sistema

5.1 PROTOCOLO DE COMUNICACIÓN Y SERVIDOR DE OPERACIONES

Como se ha comentado en el apartado 4 anterior el sistema propuesto se basa en el modelo Cliente/Servidor. Para este modelo de comunicaciones es necesaria la utilización de un protocolo de comunicaciones que defina:

- Cómo se codifican las peticiones.
- Cómo se sincronizan entre sí los procesos.

Para la codificación de las peticiones se han utilizado cadenas de caracteres aprovechando el soporte que ofrecen los lenguajes Java y C++ Builder para el envió de cadenas vía socket. Para la sincronización entre los clientes y el servidor se ha utilizado una comunicación bloqueante, es decir, los clientes después de enviar una petición al servidor deben esperar la respuesta del servidor para poder continuar, además el servidor no realizará nuevas operaciones hasta que no haya terminado la operación actual.

Las peticiones que envían los clientes al servidor tienen el siguiente formato:

```
Usuario:Cod_operación:Param1:Param2…
```

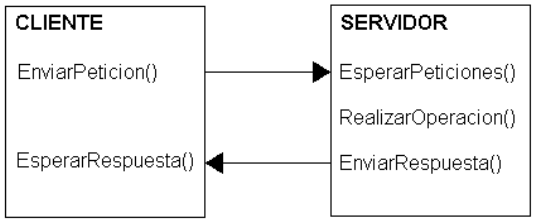

Figura 4: Protocolo de comunicaciones

Los diferentes códigos de operación utilizados son:

1. Para la identificación de usuario: *Usuario:I:Password*
2. Para petición de ficheros existentes en el directorio virtual: *Usuario:L:*
3. Para guardar un fichero: *Usuario:G:nombre_fichero*
4. Para compilar un fichero: *Usuario:C:nombre_fichero*
5. Para ejecutar un programa: *Usuario:R:nombre_fichero:nombre_programa*
6. Para petición de fichero por email: *Usuario:M:nombre_fichero*

Por su parte el servidor después de realizar la operación que le ha pedido el cliente envía una respuesta que será: *OK:Resultado_de_la_operación.*

En el caso de la operación se haya realizado correctamente o: *ERROR:Mensaje_de_error*

Si no se ha podido realizar la operación correctamente. Algunos de los errores comunes son:

- *Error: Usuario no existente.*
- *Error: Programa no encontrado.*
- *Error: Lista ERRORES COMPILACIÓN.*

Para poder conectar con el servidor los clientes deben identificarse mediante un nombre de usuario y una contraseña que el servidor verificará en su base de datos. Una vez que un cliente se ha identificado el servidor le ofrecerá la posibilidad de realizar las siguientes operaciones:

- Abrir programa: esta operación permite al cliente abrir cualquier programa que tenga guardado dentro de su directorio (situado en el servidor).
- Guardar programa: guarda el programa en el directorio del usuario (situado en el servidor).
- Enviar programa vía correo electrónico: dado que los programas se guardan en el servidor, el alumno podrá hacer que el servidor se los envié vía correo electrónico para poder almacenarlos en su computador (obviamente, el alumno puede disponer de ellos si los ha editado previamente con cualquier editor).
- Ver programas disponibles: el servidor devuelve la lista de programas que un usuario tiene en su directorio.
- Compilar programa: compila un programa que este disponible en el directorio del alumno.
- Ejecutar programa: ejecuta un programa enviándolo al controlador del robot vía comunicaciones RS-232.

Cabe destacar en el desarrollo del servidor que para cada alumno existe un *directorio virtual* donde son almacenados los programas que ha realizado. De esta forma se eliminan los problemas de seguridad que podrían surgir si desde la web se permitiese almacenar el programa en el disco del cliente. Así una vez que el alumno haya realizado un programa, lo almacenará en el servidor y para poder guardarlo en casa tendrá la posibilidad de recibirlo por correo o bien ponerse en contacto con el administrador del servidor y recuperarlo en disquete.

5.2 EJEMPLO DE FUNCIONAMIENTO

En esta sección se describe brevemente un ejemplo de una sesión básica de trabajo.

1. Inicialmente el usuario deberá conectarse al servidor (`http://robot.ual.es`). Aquí el alumno deberá introducir su nombre de usuario y su contraseña para poder empezar a trabajar con el sistema. Una vez validados los datos, se presentará al cliente una página web como la que se muestra en la figura 5.
2. Si no lo ha hecho previamente, el alumno deberá instalar el *plugin* que suministra el fabricante del servidor de vídeo para poder visualizar las imágenes de la cámara.
3. Para empezar a trabajar el usuario puede o bien abrir un programa que tenga guardado en su directorio virtual, o crear un programa nuevo (figura 6). Una vez editado el programa y guardado en su directorio virtual, el alumno podrá compilar el programa para corregir los errores que tenga el programa (se permite que varios usuarios estén compilando simultáneamente sus programas).

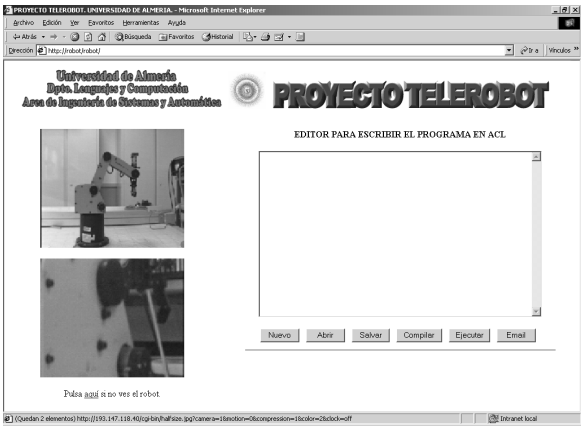

Figura 5: Página web en Cliente

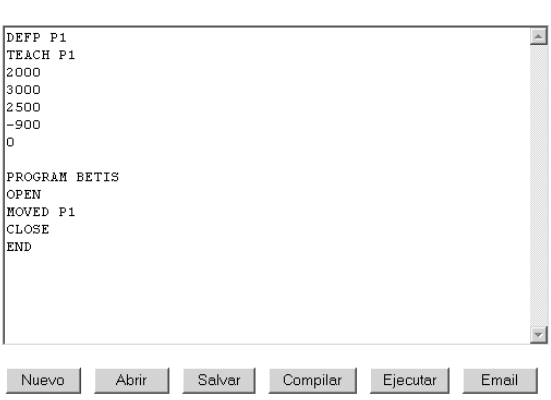

Figura 6. Ejemplo de programa

4. En este punto el alumno puede enviar el programa para su ejecución, si la operación de ejecución esta libre en el servidor (ningún otro usuario está accediendo al robot en ese momento) podrá ver el comportamiento de la célula robotizada gracias a las dos cámaras instaladas en el entorno de trabajo (figura 7).

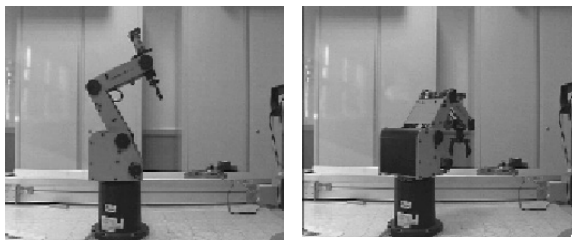
Figura 7. Visualización de ejecución

6 CONCLUSIONES Y TAREAS PENDIENTES

Se ha presentado un entorno de programación remota en ACL para robots manipuladores cuya principal novedad es la incorporación de un compilador de ACL y permite la compilación simultánea y el acceso remoto al robot por parte de los usuarios. Se enumeran a continuación las tareas pendientes a corto plazo

- Teleoperación directa desde cliente (botonera "virtual").
- Aumentar seguridad tanto a nivel de acceso a la práctica como de seguridad física del robot.
- Realimentación real de señales de robot.
- Integrar en entorno de simulación y control remoto previamente desarrollado [5].

Agradecimientos

Los autores agradecen a la CICYT la financiación de algunos equipos usados en las experiencias realizadas (proyectos QUI99-0663-C02-02 Y DPI2001-2380-C02-02).

Referencias

[1] Aho, A. V., R. Sethi, y J. D.Ullman. "Compiladores. Principios, técnicas y herramientas". Ed. Addison Wesley, 1990.

[2] Dormido, S., J. Sánchez, F. Morilla, (2000) "Laboratorios virtuales y remotos para la práctica a distancia de la Automática", sesión plenaria. XXI Jornadas de Automática, Sevilla 18-20 de septiembre de 2000.

[3] Nasa, (2000). http://ranier.oact.hq.nasa.gov/ telerobotics_page/ realrobots.html

[4] Sánchez Moreno, J. (2001) "Un nuevo enfoque metodológico para la enseñanza a distancia de asignaturas experimentales: análisis, diseño y desarrollo de un laboratorio virtual y remoto para el estudio de la Automática a través de Internet". U.N.E.D. Tesis Doctoral.

[5] Sarabia, J.F., F. Rodríguez, L. Iribarne, M. Berenguel (2001). "Herramienta de simulación de una célula robotizada". XXII Jornadas de Automática, Barcelona.

[6] Servidor de vídeo AXIS 2400. http://www.axis.com

[7] Soon, T.H., L.K. Hong, K.K. Kuen. VR. Telerobot System. 5th International Conference on Manufacturing Technology, Beijing. 1999.

[8] Scorbot ER-V (1984) Eshed Robotec Ltd.

[9] Tan, K.K., T.H. Lee and F.M. Leu (1999). "Development of a Distant Laboratory using LabVIEW". National Instruments Docs.

[10] Taylor, K.,J. Trevelyan (1994). Telerobot on the WWW. http://telerobot.mech.uwa.edu.au

[11] Torres, F., S.T. Puente, J. Pomares, F.A. Candelas, F.G. Ortiz (2001). "Robolab: Laboratorio virtual de robótica básica a través de Internet". EIWISA 2001, Madrid.