

# Diseño y Construcción de un Sistema Móvil Controlado desde Internet

Javier Ruiz-del-Solar, Eyal Shats

Departamento de Ingeniería Eléctrica - Universidad de Chile

Av. Tupper 2007 - Santiago de Chile

Email: [eshats@cec.uchile.cl](mailto:eshats@cec.uchile.cl), [jruizd@cec.uchile.cl](mailto:jruizd@cec.uchile.cl)

**Resumen** – En el presente artículo se describe el desarrollo de un sistema robótico móvil controlado a través de Internet. Este sistema, diseñado y construido especialmente con este propósito, consiste en un robot móvil sobre el cual va montada una cámara de video. Ambos, el robot y la cámara pueden ser manejados desde un sitio WEB, en el cual se despliegan asimismo las imágenes captadas por la cámara.

**Abstract** – In this paper is described a mobile robotic system that is controlled over Internet. The system consists on a mobile platform, which has a mobile CCD-camera mounted over it. Both, robot and camera can be controlled from a WEB site. In this WEB site are also displayed the camera images.

## 1. Introducción

El masivo uso del WWW y el continuo mejoramiento y abaratamiento del hardware computacional han incentivado el desarrollo de diversos tipos de aplicaciones sobre Internet [5]. En particular el desarrollo de sistemas controlados a través de Internet (ej. Webcam) ha suscitado un creciente interés. Se espera que la utilización de este tipo de sistemas se vea incrementado en los próximos años, y que en un futuro cercano tareas de control y sensado sean realizadas normalmente a través de Internet.

En este contexto, el trabajo desarrollado consiste en el control de un sistema robótico móvil a través de Internet. Este sistema, diseñado y construido especialmente con este propósito, está formado por un robot móvil sobre el cual va montada una cámara de video (ver figura 1). Ambos, el robot y la cámara pueden ser manejados desde un sitio WEB, en el cual se despliegan asimismo las imágenes captadas por la cámara (ver figura 2).

La organización de este artículo es la siguiente. En la secciones 2 y 3 se describe los principales componentes de hardware y software del sistema desarrollado. En la sección 4 se presentan algunas conclusiones y proyecciones de este trabajo.

## 2. Implementación de Hardware

Tal como se muestra en la figura 1, el sistema móvil consiste de un robot y de una cámara de video. La cámara de video está montada encima del robot y los dos son manejados por un computador portátil, denominado computador local y que se encuentra al interior del robot. El hardware del sistema móvil consiste principalmente en dos estructuras metálicas. La primera y más grande tiene dos pisos y está montada sobre

dos plataformas móviles de cuatro ruedas cada una (ver figura 4-a). Estas plataformas están conectadas a dos motores de corriente continua, los cuales le permiten realizar los movimientos básicos (avanzar, retroceder y doblar). El primer piso de esta estructura contiene el circuito controlador del robot, una batería que lo alimenta, y la fuente que alimenta los motores de corriente continua (ver figura 4-b). El segundo piso contiene el computador portátil con conexión inalámbrica a Internet (ver figura 4-c). El computador está conectado al circuito controlador de los motores del robot mediante su puerta paralela y al circuito controlador de la cámara mediante su puerta serial (ver esquema en figura 3). La segunda estructura del sistema móvil contiene en su interior el circuito controlador del movimiento de la cámara y sobre esta estructura se encuentra montada la cámara de video (ver figura 4-d). La cámara de video tiene un servomotor conectado en su base, el que le permite realizar movimientos en el eje horizontal.



Figura 1: Sistema Móvil desarrollado.



Figura 2: Sitio WEB del sistema móvil. Se muestran el panel de control y la imagen captada por la cámara de video.

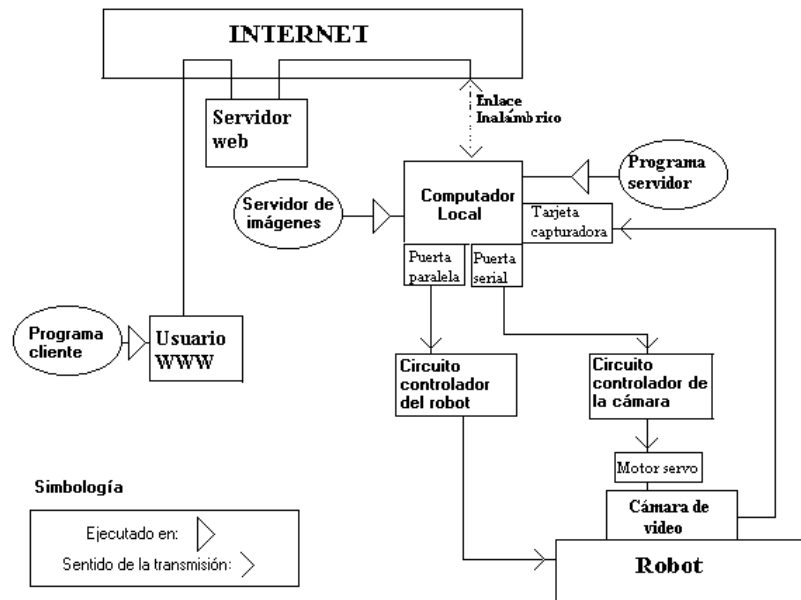


Figura 3: Esquema de la solución de software y hardware implementada.

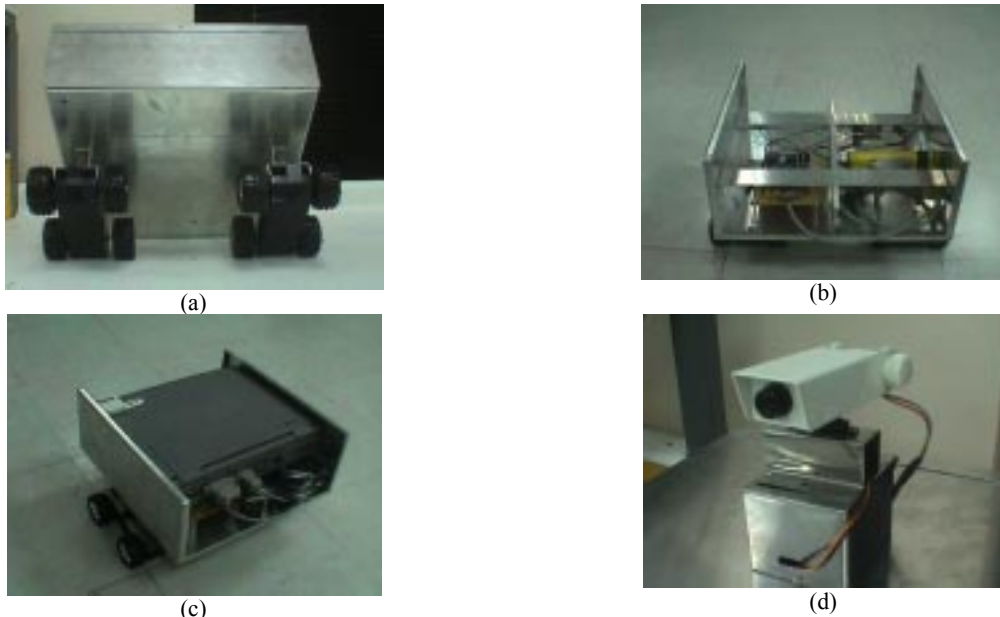


Figura 4: (a) Vista inferior de sistema móvil. (b) Plano general de primer piso de sistema móvil. (c) El computador portátil localizado en el segundo piso del sistema móvil. (d) Vista frontal de cámara de video y su estructura de soporte.

### 2.1. Control del Robot

Los dos motores de corriente continua del robot son controlados desde la puerta paralela (protocolo CENTRONICS) del computador. El circuito controlador utiliza dos relés para controlar cada motor (ver figura 5). Por cada relé se utiliza un transistor ( $T_i$ ), un diodo ( $D_i$ ), un optoaislador ( $O_i$ ) y una resistencia ( $R_i$ ).

El giro de cada motor se controla mediante los bits  $b_1$  y  $b_2$ . Si cualquier de estos bits vale 1, el motor gira en una determinada dirección (el estado 11 no está permitido). Para

controlar ambos motores se utilizan los 4 bits inferiores de la puerta paralela ( $D_0$ - $D_3$ ). En la tabla 1, se muestran las distintas secuencias de bits necesarias para los distintos sentidos de movimiento del robot.

El motor servo sobre el que se encuentra montada la cámara de video es manejado por una tarjeta Mini SSC II [11] (ver figura 6), la que es controlada desde la puerta serial (protocolo RS-232) del computador (ver figura 3). Esta tarjeta permite controlar los movimientos de hasta 8 motores servo con simples comandos ASCII enviados desde la puerta serial.

Funciona como una suerte de traductor de comandos en formato ASCII a comandos PWM (*Pulse Width Modulation*), que son los necesarios para controlar el movimiento de los motores servo.

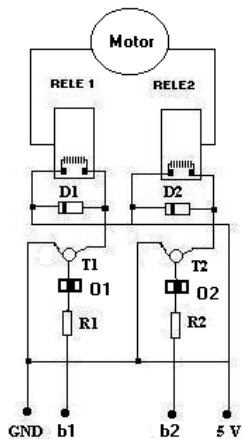


Figura 5: Esquema del circuito controlador de un motor de corriente continua.

Tabla 1: Comandos enviados por puerta paralela para control de movimientos del robot.

Sentido de movimiento del robot	Byte enviado
Adelante	10100000
Atrás	01010000
Doblar a la izquierda	10000000
Doblar a la derecha	00100000
Parar	00000000

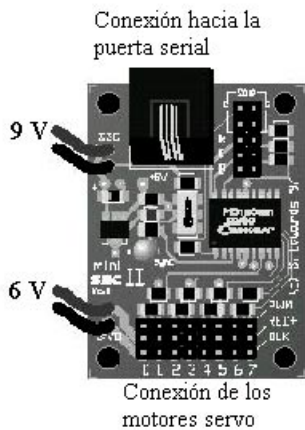


Figura 6: Tarjeta controladora de motores servo "Mini SSC II".

Cuando la tarjeta se configura para funcionar con un rango de movimientos de 90 grados (desde -45 hasta +45), el circuito posee una resolución de 4 microsegundos, dando una precisión de posición de 0.36 grados. Alternativamente es posible configurar la tarjeta para que funcione con un rango de movimientos de 180 grados (desde -90 hasta + 90). Para controlar los servo motores usando el Mini SSC II es necesario enviar la siguiente información desde la puerta serial del computador:

- 255, para inicializar el circuito
- El número del motor que se quiere mover

- La posición a la cual se quiere mover el motor.

El número del motor varía entre 0 y 7 y la posición varía entre 0 y 254. A modo de ejemplo, si se quiere mover el motor número 7 al centro (cero grados), se debe enviar la siguiente información por la puerta serial: 255, 7, 127. Este código altera solamente la posición del motor número 7 y no afecta la posición de los otros motores. Las posiciones se enumeran desde 0 hasta 254 por lo que la posición numero 127 corresponde al centro.

## 2.2 Enlace Inalámbrico

Para obtener una conexión inalámbrica a Internet en el computador local se utilizó el protocolo IEEE 802.11 [8], el cual define un nuevo estándar de redes de área local inalámbricas (WLANs: Wireless LANs). Para la implementación del protocolo IEEE 802.11 es necesario tener tarjetas de red inalámbricas y puntos de acceso. El punto de acceso actúa como puente entre la red inalámbrica que se está implementando y la red fija existente, la cual está conectada a Internet.

### 2.2.1 Tarjeta de red inalámbrica

La tarjeta de red inalámbrica, fue diseñada para usarse en computadoras portátiles, por lo que corresponde a una tarjeta de tipo PCMCIA (*Personal Computer Memory Card International Association*). La tarjeta viene integrada con una pequeña antena que le permite mandar y recibir la información de manera inalámbrica. La velocidad de transmisión de la tarjeta utilizada es de 11 Mbps ya que utiliza la versión mejorada del protocolo IEEE 802.11, el IEEE 802.11b. Cabe destacar que la velocidad de transmisión no es fija sino que depende de la distancia del usuario al punto de acceso y del entorno en que se encuentra el cliente inalámbrico.

### 2.2.2 Punto de acceso

El punto de acceso tiene una interfaz para conectarlo con redes Ethernet mediante 10Base-T y 10Base-2, y puede ser montado tanto sobre Ethernet como sobre Fast-Ethernet. El punto de acceso posee un sistema de seguridad para proteger a la red del acceso de personas no autorizadas. Este sistema utiliza tablas de acceso que incluyen listas de usuarios autorizados, lo cual sirve para restringir el ingreso a la red, y encriptación de datos para proteger la información que viaja en la red. Para el proceso de instalación, gestión y control del punto de acceso, se tiene un software ambientado en Windows que permite de una manera sencilla realizar estos procesos. Debido a las características de nuestro sistema, es más importante la velocidad de la conexión inalámbrica que la distancia entre el usuario inalámbrico y el punto de acceso. Esto se consideró al momento de configurar los equipos, por lo que se eligió la opción de modulación DSSS (*Direct Sequence Spread Spectrum*) [9], que otorga mayor velocidad

en distancias relativamente cortas. Para brindar mayor seguridad al sistema se habilitó la opción de encriptación de datos en el punto de acceso.

### 3. Implementación de Software

La solución de software implementada para permitir a los usuarios manejar el sistema móvil a través de la red Internet consiste en una *interfaz socket* [2], basada en el modelo cliente-servidor.

#### 3.1 Programa servidor

El programa servidor se ejecuta en el computador local (ver figura 3) y es el encargado de recibir comandos remotos, procesarlos y ejecutar la acción correspondiente a cada comando. Para programar el programa servidor se eligió el lenguaje C++ debido a la excelente plataforma que ofrece para la *interfaz socket*, además de su condición de lenguaje orientado a objeto, que facilita la programación. Es importante notar la diferencia entre el programa servidor y el servidor WEB. El primero es un programa que se instala en el computador local, en cambio, el servidor WEB es un computador poderoso en el cual se almacena el sitio WEB del sistema móvil (ver figura 2).

##### 3.1.1 Asignación de direcciones y puertos

Para implementar correctamente la *interfaz socket*, el programa servidor necesita de una dirección IP [7] y de un puerto TCP [6]. La dirección IP asignada al programa servidor es la dirección IP del computador en el cual se está ejecutando el programa servidor. Es importante destacar que esta dirección IP no se incluye en el código del programa servidor, por lo que es posible ejecutar el programa servidor en un computador con otra dirección IP, sin tener que cambiar ningún parámetro. En cambio, sí será necesario configurar este parámetro en el programa cliente. Se observó que los puertos TCP que se encuentran entre los 5000 y los 6000 tienen un buen funcionamiento en el trabajo con la *interfaz socket* por lo que se eligió el puerto número 5678 para este proyecto. Para el uso del puerto TCP en el programa servidor se define la variable global `DEFAULT_PORT` y se le asigna el valor del puerto elegido.

##### 3.1.2 Funciones controladoras

Para que el programa servidor ejecute acciones es necesario definir funciones que al ser llamadas manden los comandos necesarios a la puerta paralela (control de motores del robot) o a la puerta serial (control del motor de la cámara) del computador local.

La primera función definida es *mover\_robot*, que se declara como: `int mover_robot (int comando)`. Primero se inicializa la puerta paralela del computador local para poder mandar información por esta. Luego se envía la variable *comando* que identifica el tipo de movimiento del robot (ver

comandos permitidos en tabla 1). Para enviar esta variable se utiliza la instrucción *\_outp* de la siguiente manera: `_outp(PORT1, comando)`, en que *PORT1* es la dirección de memoria de la puerta paralela.

La siguiente función definida es *mover\_cam*, la que se utiliza para mover la cámara de video. La función *mover\_cam* se declara como: `int mover_cam (int motor, int posicion)`. Para comenzar la acción de control, es necesario inicializar la puerta serial y enviar el valor 255 al circuito controlador para inicializarlo. Para esto se utiliza la función `_outp(PORT2,255)`. Luego hay que ejecutar la acción deseada, para esto se mandan los siguientes comandos `outp(PORT2,motor)` y `outp(PORT2,posicion)`. La variable *motor* indica sobre cual motor se quiere ejecutar la acción. Para los efectos del proyecto se tiene solamente un motor y está conectado en la posición 7. La variable *posicion* indica la posición a la cual se quiere mover la cámara. (ver explicación en 2.1).

##### 3.1.3 Manejo de Sockets

Para definir la interfaz socket se llama a la función *socket()*. Esta función define un nuevo socket y retorna un descriptor de éste para ser usado a lo largo de todo el proceso de comunicación. La función *socket()* [3] es llamada de la siguiente manera:

```
sock=socket(AF_INET, SOCK_STREAM, IPPROTO_TCP),
```

en que `AF_INET` representa la familia de direcciones correspondientes al protocolo TCP/IP, `SOCK_STREAM` corresponde al tipo de socket confiable ya que utiliza el protocolo TCP en su capa de transporte y `IPPROTO_TCP` indica que se está creando un socket sobre una red que utiliza el protocolo de transporte TCP (la red Internet). Una vez creado el socket se definen los parámetros de la estructura socket (en nuestro caso se utilizó la variable *address*). El parámetro más importante para el caso del servidor es el puerto TCP asignado para la aplicación. La variable de la estructura socket correspondiente al puerto TCP es *sin\_port*, la cual es accedida como `address.sin_port`. Dado que se ha definido anteriormente la variable `DEFAULT_PORT` como el valor del puerto TCP asignado (5678), basta con escribir: `address.sin_port=DEFAULT_PORT`.

A pesar de haber designado los valores a las variables de la estructura socket, tal como el puerto TCP y la dirección IP, el socket en sí todavía no conoce esta información. Para ligar ambos tipos de información se utiliza la función *bind()*:

```
bind(sock, (struct sockaddr *)&address, sizeof(address)),
```

en que el primer argumento corresponde al descriptor del socket creado anteriormente, el segundo argumento indica que se deben considerar todas las direcciones definidas anteriormente en la estructura de socket y el tercer argumento indica el largo de las direcciones utilizadas.

Es importante notar que después de utilizar la función *bind()* el socket tiene asignadas las direcciones correspondientes pero falta definir si el socket creado será socket activo o pasivo. Para definir el socket como pasivo, es

decir, dejarlo preparado para recibir comandos, se utiliza la función *listen()*. Esta función se utilizó de la siguiente manera: *listen(sock, 5)*, en que el primer argumento corresponde al descriptor del socket creado anteriormente y el segundo argumento indica la cantidad máxima de usuarios que pueden estar en la cola de espera del programa servidor de manera simultánea. La cola de espera del servidor almacena las conexiones de usuarios nuevos que el programa servidor no puede atender inmediatamente por estar ocupado. Se decidió utilizar una cola máxima de 5 usuarios debido a la baja capacidad de procesamiento del computador local, que es el computador en el cual se ejecuta el programa servidor.

A continuación se crea un ciclo (loop) para iniciar el proceso de aceptación de conexiones remotas. Para poder aceptar las conexiones remotas se utiliza la función *accept()* de la siguiente manera:

```
accept(sock, (struct sockaddr *)&from, &fromlen),
```

en que el primer argumento corresponde al descriptor del socket creado anteriormente, el segundo argumento es un puntero a la nueva estructura *from*, en la que se definen las direcciones de la nueva conexión remota, y el tercer argumento define el tamaño de la estructura *from*. Con esta información es posible conocer las características de la conexión remota, accediendo a los parámetros de la estructura *from*. Por ejemplo, para conocer el puerto TCP desde el cual se comunica el usuario se debe acceder a la variable *from.sin\_port*. Cuando una conexión es aceptada por la función *accept()* se procede a esperar la recepción de comandos remotos a ser procesados.

### 3.1.4 Procesamiento de Comandos Remotos

Para recibir los comandos enviados por los usuarios, se utiliza la función *recv()* de la siguiente manera: *recv(sock,buffer, 255, 0)*, en que el primer argumento corresponde al descriptor del socket, el segundo argumento corresponde a la variable que almacena el comando que se recibe, el tercer argumento corresponde al tamaño de la variable, y el cuarto argumento corresponde a *flag*, argumento que se utiliza siempre con valor nulo. Los comandos que son permitidos por el programa servidor son los siguientes: *adelante*, *atrás*, *doblar\_izq*, *doblar\_der*, *parar*, *cam\_der*, *cam\_cen*, *cam\_izq* y *salir*. Los cinco primeros corresponden a comandos de control del robot (ver tabla 1) y los tres siguientes a comandos de control de la cámara. En caso de recibir el comando *salir* es necesario cerrar la interfaz socket, lo cual se realiza mediante la función *closesocket()*.

### 3.2 Programa cliente

El programa cliente es el encargado de enviar comandos al programa servidor para que ejecute una acción determinada. Este programa puede ejecutarse desde cualquier computador conectado a Internet (ver figura 3). El programa cliente puede definirse de dos maneras distintas. La primera consiste en la utilización de un programa (archivo) de tipo

CGI, residente en el servidor WEB (ver figura 3), pero ejecutado en el computador del usuario (usuario WWW en figura 3) cuando éste invoca la página WEB correspondiente. Para realizar esta operación el usuario debe conocer únicamente la dirección de la página WEB, la que por ejemplo puede ser "<http://robotMovil.die.uchile.cl>". La segunda posibilidad consiste en utilizar la aplicación TELNET como un programa cliente. El programa TELNET utiliza la *interfaz socket* para crear una comunicación, proceso que se realiza mediante el uso de la dirección IP y del puerto TCP del programa servidor (ver 3.1.1). Ambos valores deben ser ingresados, y por tanto ser conocidos de antemano por el usuario.

Para utilizar el programa cliente es necesario crear una interfaz para el usuario, a través de cual éste pueda ingresar comandos para mover el sistema móvil. En el caso del programa cliente que utiliza TELNET la interfaz se crea mediante un mensaje de bienvenida y un conjunto de instrucciones, enviados ambos por el programa servidor al programa cliente. En el caso del programa cliente de tipo archivo CGI, esta interfaz está dada por la página WEB correspondiente (ver figura 2). La primera misión del archivo CGI programado para este proyecto es la de realizar una conexión de tipo socket con el programa servidor. El archivo CGI se programó usando el lenguaje PERL [4], específicamente el módulo *IO::Socket*. La forma de crear el socket usando este módulo es la siguiente:

```
$socket = IO::Socket::INET->new( PeerAddr  
=> '146.83.6.118', PeerPort => '5678', Proto => 'tcp');
```

En que *PeerAddr* corresponde a la dirección IP del computador en donde se está ejecutando el programa servidor, es decir, la dirección IP del computador local. *PeerPort* corresponde al puerto TCP asignado para ejecutar la aplicación (se ha asignado el puerto 5678 en el programa servidor por lo que es necesario definir el mismo en el programa cliente) y *Proto* es la definición del protocolo sobre el cual se está ejecutando la aplicación. En este caso la aplicación se ejecuta sobre la red Internet por lo que el protocolo especificado es el correspondiente al protocolo de la capa de transporte (TCP).

En el caso de que se use TELNET para establecer la conexión, se debe especificar la dirección IP del computador en donde se ejecuta el programa servidor junto con el puerto TCP habilitado para esta aplicación. Para los efectos del proyecto se debe especificar la dirección IP del computador local junto con el puerto 5678 que fue el puerto habilitado para la aplicación. El llamado a la aplicación TELNET tendrá la siguiente forma: *telnet 146.83.6.118 5678*

### 3.3 Servidor de Imágenes

El servidor de imágenes consiste en un programa que permite desplegar las imágenes captadas por una cámara de video en un sitio WEB, es decir, permite transformar una cámara de video normal en una Webcam. En nuestro caso

este programa se instala en el computador local (ver figura 3), para poder transmitir las imágenes captadas por la cámara de video al sitio WEB. El servidor de imágenes seleccionado es el *webcam32* [10], el cual puede funcionar de dos maneras diferentes, la primera denominada método FTP y la segunda método HTTP. El método FTP de funcionamiento del servidor de imágenes consiste en capturar las imágenes obtenidas por la cámara de video transformando la secuencia de imágenes que forman el video en imágenes individuales de extensión JPEG, y a continuación enviar estas imágenes al servidor WEB utilizando el protocolo FTP. El servidor WEB requiere de un sitio WEB en el cual se desplieguen las imágenes, por lo cual una vez integrada la información enviada con el código fuente del sitio será posible visualizar estas imágenes en Internet. El método HTTP de funcionamiento consiste en configurar al servidor de imágenes como un pequeño servidor WEB. De esta manera el servidor de imágenes captura las imágenes desde la cámara de video y las envía directamente al navegador del usuario.

Dado que se eligió el método de funcionamiento HTTP, fue necesario incluir en el código fuente de la página WEB del sistema móvil (ver figura 2) la etiqueta <IMG> [1], la cual se encarga de la visualización de imágenes en el sitio WEB. Esta etiqueta, perteneciente al código HTML se incluye de la siguiente manera:

```
<IMG SRC = http://IP/video/JPEG\ width=320 height=240>
```

en que IP corresponde a la dirección IP del computador local y *Width* y *Height* corresponden al tamaño de la imagen a colocar en el sitio.

#### 4. Conclusiones

En este artículo se ha descrito el control de un sistema robótico móvil a través de Internet. Este sistema está formado por un robot móvil sobre el cual va montada una cámara de video. Ambos, el robot y la cámara pueden ser manejados desde un sitio WEB, en el cual se despliegan asimismo las imágenes captadas por la cámara. Entre las características interesantes del sistema descrito se cuenta el hecho de que éste se conecta en forma inalámbrica a Internet, utilizando un protocolo de comunicaciones Wireless-LAN estándar (IEEE 802.11).

El desarrollo de este trabajo ha permitido demostrar la amplia disponibilidad actual de tecnologías de software y hardware que facilitan considerablemente el desarrollo de este tipo de aplicaciones. En este contexto se espera que la utilización y construcción de este tipo de sistemas se vea incrementado en los próximos años, y que en un futuro cercano tareas de control y sensado sean realizadas normalmente a través de Internet.

Actualmente se está ampliando las capacidades del sistema móvil descrito, de tal forma de aumentar su funcionalidad y de permitir su uso en tareas exploratorias. En particular se desea:

- controlar la velocidad de desplazamiento del sistema móvil por Internet, y

- agregar sensores de distancia al sistema móvil, de tal forma que se despliegue un mapa tridimensional del área en la cual éste se encuentra, en el sitio WEB visto por el usuario del sistema.

#### Referencias

- [1]. Castillo, C., "Tejedores del Web, Manual de HTML", <http://www.tejedoresdelweb.com>.
- [2]. Comer, D., "InternetWorking with TCP/IP" Volume III. *Client-Server Programming and Applications Principles, Protocols and Architecture*, Prentice Hall, Marzo 2000.
- [3]. Hall, B., "Beej's Guide to Network Programming, Using Sockets", Versión 1.5.5, Enero 1999. <http://www.ecst.csuchico.edu/~beej/guide/net/com>.
- [4]. Harlan, D., Foghlu, M., Doyle, P., Powers, S., Hery, M., "Using Perl 5 for Web Programming, Special Edition" Ed. QUE, Agosto 1996.
- [5]. Meeker, M. Depuy, C., "The Internet Report", Morgan Stanley, 1996.
- [6]. RFC-793, The Transmission Control Protocol (TCP).
- [7]. RFC-791, The Internet Protocol (IP).
- [8]. The IEEE 802.11 Wireless LAN standard, Wireless LAN Alliance, Raytheon Company 1997.
- [9]. Boer, J., "Direct Sequence Spread Spectrum Physical Layer, IEEE P802.11-96/49E, March 1996.
- [10]. Disponible en <http://www.webcam32.com>
- [11]. SSC-ASD2 V.2.1, "Users Manual Mini SSC-II", Scott Edwards Electronics, June 2000.

#### Agradecimientos

El desarrollo de este trabajo ha sido financiado por el proyecto DID (U. de Chile) ENL-2001/11 y por el Departamento de Ingeniería eléctrica de la Universidad de Chile.

**Javier Ruiz-del-Solar** recibió el título de Ingeniero Civil Electrónico y el grado de Magister en Ingeniería Electrónica de la U. Técnica Federico Santa María en 1991 y 1992, respectivamente. En 1997 recibió el grado de Doctor en Ingeniería de la U. Técnica de Berlín. Desde Marzo de 1998 se desempeña como Profesor Asistente del Departamento de Ingeniería Eléctrica de la U. de Chile. Entre sus líneas de investigación destaca la aplicación de Inteligencia Computacional a problemas de Reconocimiento de Patrones, Procesamiento de Imágenes y Visión Artificial.

**Eyal Shats** recibió el grado de título de Ingeniero Civil Electricista de la U. de Chile en 2001.