Capítulo 7
Aplicación de control

En este capítulo se muestra un ejemplo de la programación de una aplicación que se comunica con uno o varios clientes y estos dan instrucciones o reciben datos desde el microcontrolador a través de las funciones de la pila TCP/IP.

La aplicación se divide en dos partes:

1. El servidor implementado en el microcontrolador, al cual tienen acceso los usuarios.
2. La aplicación presentada del lado del cliente (computadora con navegador web).

Tanto en el cliente como en el servidor se ejecutan dos acciones: la primera acción corresponde a la transferencia información a través de TCP/IP (como la transferencia de recursos HTTP y los comandos de control) y la segunda al manejo de los dispositivos conectados al microcontrolador (acciones de control). A esta última acción es a la que nos referimos en este capítulo. Cabe aclarar que el microcontrolador solo ejecuta un programa que se encarga tanto de la comunicación como de las acciones de control.

En el capítulo 5 se explicó el desarrollo de la aplicación web, cuyo objetivo es la transferencia de recursos web entre clientes y servidores. La funcionalidad que brinda el Protocolo de Transferencia de Hipertexto permite que una página u otro tipo de recurso web sea codificado, almacenado y transferido desde el microcontrolador hasta una computadora cliente. Este proceso se comprende fácilmente si el recurso corresponde a una página web HTML estática, la cual se codifica en ASCII y se envía al cliente, en donde la información es interpretada por un navegador web y su contenido es desplegado y visualizado en la pantalla de la computadora, tal como se explicó en el capítulo 5.

Por otra parte, si se requiere realizar una aplicación de control que interactúe con los usuarios de manera remota a través de una red, es preciso que la aplicación tenga una funcionalidad dinámica. Para realizar esta tarea existen varias alternativas: una de ellas contempla el uso de la transferencia de archivos que ofrece HTTP; otra se basa en el uso de sockets para crear aplicaciones independientes de HTTP; también existe la posibilidad de hacer uso de recursos externos (servidores externos o archivos ejecutables en el cliente) y, por último, se tiene la opción de integrar la funcionalidad dinámica dentro del microcontrolador.

En resumen el procedimiento para controlar un dispositivo o proceso conectado al microcontrolador desde cualquier computadora con conexión a Ethernet, es el siguiente (a nivel de software, ya que para el usuario este procedimiento es transparente):

- El usuario se conecta al microcontrolador desde una computadora tecleando en la barra de direcciones del navegador web la dirección IP asignada al microcontrolador.
- En ese momento se establece una conexión TCP por la que el cliente envía la petición del recurso y el servidor regresa una página de bienvenida.
- Dentro de esta página se encuentra la referencia a otro recurso o un enlace a otra página que contiene el recurso correspondiente a la aplicación de control, la cual se mostrará en la página que el usuario visualiza.
- Dependiendo de la implementación de la aplicación, la interfaz gráfica de la aplicación de control se puede transferir desde el microcontrolador o desde algún servidor externo, para lo cual se abre una nueva conexión y se cierra cuando se completa la transferencia y el usuario puede ver en su navegador la interfaz gráfica de la aplicación de control (puede contener botones, formularios, indicadores, etc.). En este momento la transferencia de recursos web terminó y la conexión TCP se encuentra cerrada, ya no hay comunicación entre el microcontrolador y la computadora.
- Cuando el usuario realice alguna acción sobre la interfaz de la aplicación de control, se abre una nueva conexión que puede ser a través de TCP o bien de UDP.
  Si por ejemplo la interfaz de control contiene un botón con la leyenda "Conectar" que el usuario presiona para establecer una conexión, al momento de presionarlo se envía una solicitud de conexión TCP, la cual se mantiene abierta todo el tiempo que el usuario lo decida mientras él interactúa con la aplicación. Para finalizar presiona un botón con la palabra "Desconectar" y se envían los segmentos TCP necesarios para cerrar la comunicación.
  Si por otro lado la aplicación se programa sobre UDP, no es necesario que exista un botón de conectar, cada vez que se quiera enviar un comando al microcontrolador, por ejemplo presionando un botón que le indica que lea el valor de un puerto y lo regrese, se envía un paquete UDP por un socket previamente obtenido.
- En el microcontrolador a su vez, se implementa una aplicación de control que se comunica con la aplicación que corre en el cliente, respondiendo a los comandos enviados por éste.

- Debe existir un acuerdo entre ambas aplicaciones a cerca de los datos que van a intercambiar, de modo que se entiendan adecuadamente. Por ejemplo cuando el usuario presiona el botón "leer puerto X" desde la interfaz gráfica que observa en su computadora, la información que se envía corresponde a un comando previamente acordado, por ejemplo comando 20, lo que el microcontrolador debe interpretar como "leer el puerto X". Al reconocer este comando simplemente hace lo indicado: lee el valor que tiene el puerto X en ese momento y lo envía de regreso formando un nuevo paquete (TCP o UDP).

7.1 Descripción de una aplicación del lado servidor

La programación de la aplicación del lado servidor corresponde a las acciones que se ejecutan en el microcontrolador como respuesta a las solicitudes de los clientes que se conectan a él, por lo tanto debe estar programada en el código que ejecuta el microcontrolador.

El microcontrolador se debe configurar como servidor y hacer uso de las funciones de sockets que proporciona la pila TCP/IP. En la inicialización del sistema se debe obtener el socket que se usará para manejar la comunicación y se le debe asignar un número de puerto. El socket se configura como servidor y espera conexiones o datagramas. Cuando llega un paquete de datos a través del socket se debe analizar su contenido y realizar la acción requerida, que podría corresponder a activar alguna salida del microcontrolador, leer un dato de algún puerto, leer el valor del convertidor analógico digital y escribir un dato al display, entre otros procesos.

7.2 Descripción de una aplicación del lado cliente

La aplicación que se ejecuta en el cliente debe mostrar una interfaz gráfica que le permita al usuario interactuar con el proceso conectado al microcontrolador e impone los límites sobre las acciones que el usuario puede realizar sobre el proceso. Se puede utilizar cualquier lenguaje de programación que tenga funciones para el manejo de sockets, como Visual Basic, Java, C, C#, dotNet, python y PHP, entre otros.

La programación de la aplicación del lado cliente se puede realizar de varias formas, esto depende de los conocimientos y habilidades del programador, de las herramientas y recursos que se tengan disponibles y de los requerimientos de la aplicación. A continuación se muestran algunas de las formas en que se puede programar la aplicación, resaltando que la implementación no está limitada ni condicionada a hacerse de alguna de estas formas. Cada uno de los métodos mostrados tiene ventajas y desventajas y se recomienda que se emplee el que mejor se adapte a las necesidades de la aplicación y conocimientos del programador.

7.2.1 Transferencia de la aplicación como un recurso web

Este método hace uso del Protocolo de Transferencia de Hipertexto para transferir el código de la aplicación de control de manera similar a la transferencia de una página web, como se analizó en el capítulo 5.

Lo primero que se debe hacer es programar la aplicación como una interfaz gráfica con todos los botones, cajas de texto, pantallas, indicadores y demás objetos necesarios. La programación se puede realizar en algún lenguaje de programación como Visual Basic, Java o cualquier otro lenguaje que permita crear interfaces gráficas (objetos como *ActiveX* o *applets*). El código de la aplicación debe contener los parámetros de comunicación con el microcontrolador, es decir, debe existir alguna función en la que se obtenga un socket y realice una conexión activa al microcontrolador usando la dirección IP y el número de puerto asignados al microcontrolador. En las funciones de los botones de la interfaz se debe programar el envío de paquetes TCP o UDP que contengan la información del comando, de manera que al presionar el botón se generen dichos paquetes y se envíe la información hasta el microcontrolador.

Para que la aplicación funcione como se está explicando es necesario que el lenguaje de programación seleccionado sea capaz de manejar conexiones TCP y/o UDP, esto se hace generalmente a través de librerías de sockets incluidas en los compiladores.

Una vez que se tenga programada la aplicación de control ésta debe codificarse de la misma manera que se codifica una página, una imagen o cualquier otro recurso web y almacenarse en el servidor, es decir, en la memoria del microcontrolador. Ahora lo único que resta por hacer es incluir el recurso web que representa la aplicación de control dentro de una página web, como si se tratara de una imagen, con una etiqueta

especial en el código HTML de dicha página (<object>). La página HTML que contiene la aplicación también se codifica y se almacena en la memoria del microcontrolador, de la manera en que se mostró en el capítulo 5. Cuando se le solicite al microcontrolador la página que contiene la aplicación de control, el código de la aplicación se transfiere hasta el cliente junto con el código de la página HTML. El resultado que el usuario observa en su navegador es una página web que contiene una interfaz gráfica con la que puede interactuar. En la Figura 7.1 se observa un ejemplo de una interfaz con este tipo de aplicación.

El código de la aplicación de control se transfiere a la computadora cliente y una vez ahí lo ejecuta la computadora, es decir, el microcontrolador no juega ningún papel en la ejecución de la aplicación, sólo almacena el código y lo transfiere a la computadora para que ésta lo ejecute. La observación anterior implica que si es necesaria la instalación de alguna herramienta para el funcionamiento de la aplicación, ésta debe estar instalada en la computadora, sin ser responsabilidad del servidor el correcto funcionamiento de la aplicación. Por ejemplo, si la aplicación se programa en Java es necesario que la máquina que va a ejecutar dicha aplicación tenga instalada la máquina virtual de Java para que la aplicación funcione.

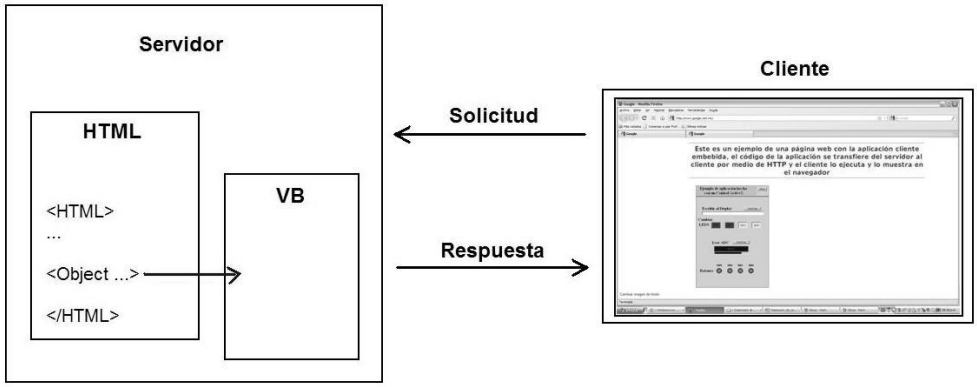

Figura 7.1 Ejemplo de aplicación almacenada en el servidor y ejecutada en el cliente

En la Figura 7.2 se muestra la forma en que se ve una aplicación de este tipo.

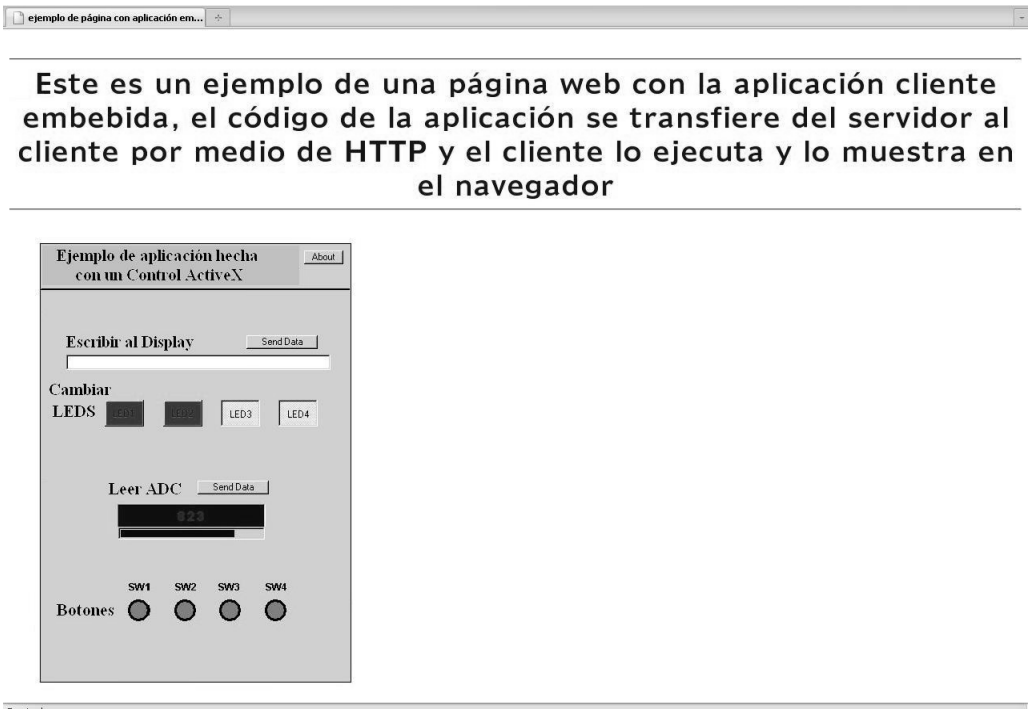

Figura 7.2 Interfaz gráfica que observa un usuario incrustada dentro de la página web

Ventajas:

- La aplicación de control se puede programar como una interfaz común de la misma manera en que se programaría para otros fines, el programador que esté familiarizado con la creación de interfaces en cualquier lenguaje de programación puede crearlas de manera fácil sin la necesidad de realizar configuraciones especiales.
- La manera de incluir los archivos de la aplicación de control al servidor es la misma que se emplea para incluir recursos web, como páginas HTML, por lo tanto no se debe aprender un procedimiento nuevo.
- La transferencia del código de la aplicación se realiza a través de HTTP, por lo tanto se aprovecha la programación de este protocolo para transferir la página y el código de la aplicación de control.
- Al tener la aplicación de control almacenada directamente en el microcontrolador, ésta se tiene disponible en cualquier momento que se establezca comunicación con el microcontrolador.

Desventajas:

- Si no se maneja la programación de interfaces gráficas en ningún lenguaje de programación es necesario aprender a hacerlas.
- Si la computadora en la que se va a correr la aplicación requiere la instalación de programas, software especial o realizar configuraciones, se deben realizar todas las acciones necesarias en cada equipo donde se ejecute la aplicación.
- Es necesario considerar el espacio que utiliza la aplicación de control codificada para ser almacenada dentro de la memoria del microcontrolador, esto impone una restricción en el tamaño de la aplicación de acuerdo a las capacidades del microcontrolador.
- Cada vez que se modifique la aplicación de control es necesario reemplazar el código en el servidor, volver a compilar el proyecto y volver a cargar el programa al microcontrolador.
- Si falla la comunicación HTTP antes de que se termine de transferir el código de la aplicación, el usuario no podrá ver la aplicación ni interactuar con el proceso.

7.2.2 Programación de la aplicación en un servidor externo

Los lenguajes de programación que utilizan los servidores web incluyen funciones para el manejo de sockets que les permite comunicarse con otros servidores a través de las funciones de los protocolos de la pila TCP/IP, esto quiere decir que se pueden crear páginas web que incluyen en su código la implementación de sockets para comunicarse con otros servidores e intercambiar información con ellos de manera dinámica. Entre los lenguajes de programación que brindan estos servicios se encuentran PHP, Java, C#, dotNet y Perl, entre otros. Sin embargo estos lenguajes de programación funcionan en servidores con sistemas operativos y/o software de red específico, por lo tanto no es posible almacenar una página de este tipo en el microcontrolador porque se tendría que ejecutar desde el mismo microcontrolador y éste no cuenta con las características suficientes para que se le instale un sistema operativo o el software de red necesario.

Sin embargo es posible aprovechar las ventajas de estos lenguajes de programación para realizar una aplicación de control, para lo cual se requiere el uso de un servidor externo que soporte la programación en algún lenguaje para páginas web dinámicas.

En este método se programa la aplicación como una página web dinámica, utilizando alguna librería de funciones para sockets. En el código de la página se obtiene un socket y se realiza una conexión activa al microcontrolador ya sea TCP o UDP, usando la dirección IP y número de puerto asociados al microcontrolador. En la programación de la página se pueden incluir botones, formularios, cajas de texto, tablas y cualquier otro elemento que le permita al usuario la interacción con el proceso remoto. También se pueden incluir scripts, validaciones, elementos como animaciones, imágenes, sonido, u otros elementos que se puedan incorporar normalmente en una página. Como ya se dijo, esta página se almacena en un servidor externo que soporta el lenguaje de programación con el que fue creado.

Para que el usuario acceda a la página puede conectarse al microcontrolador y solicitar la página de bienvenida, en donde se encuentra un enlace a la página que contiene la aplicación (residente en un segundo servidor). No es necesario que el usuario sepa la dirección del servidor donde reside la página con la aplicación de control, ya que la dirección se encuentra en el código de la página que se transfiere del microcontrolador a la computadora cliente, por lo tanto el usuario sólo debe conocer la dirección IP del microcontrolador y al visualizar la página de bienvenida debe dar un clic en el enlace que indica la

aplicación de control. Una vez que se visualice la página de la aplicación, la computadora cliente se comunica con el servidor número 2 (el que almacena la aplicación de control programada en un lenguaje específico) y este servidor a su vez se comunica con el servidor número 1 (el microcontrolador) cada vez que el cliente realiza una acción, por ejemplo, si el usuario observa en la página un botón que indica activar alguna variable del proceso, al presionar dicho botón la computadora cliente se comunica con el servidor número 2, que es el que está ejecutando la aplicación que observa el cliente. El servidor número 2 recibe la orden y se comunica con el microcontrolador a través de un socket, indicándole la información que envía el cliente. El microcontrolador recibe la información y realiza la acción correspondiente, podría generar una respuesta y regresarla al servidor número 2, quien a su vez regresa la información al cliente. En la Figura 7.3 se ilustra el proceso de comunicación llevado a cabo mediante una aplicación de este tipo.

Ventajas:

- La programación de la aplicación de control en el cliente se puede programar como una página web con cualquier lenguaje de programación web que el programador domine.
- No es necesario incrustar la página en la memoria del microcontrolador.
- Se cuenta con una mayor libertad en cuanto al tamaño de la página y se puede hacer tan grande como se desee o como sea necesario.
- Si se requiere modificar la aplicación de control del lado del cliente, no es necesario recompilar código en el microcontrolador ni volver a programarlo.
- No se requiere la instalación de software especial en la computadora del cliente.

Desventajas:

- Se depende de un servidor externo por lo que se debe tener en cuenta su disponibilidad para garantizar la comunicación con el microcontrolador.
- Si no se maneja un lenguaje de programación de páginas web dinámicas es necesario aprender a usar alguno.

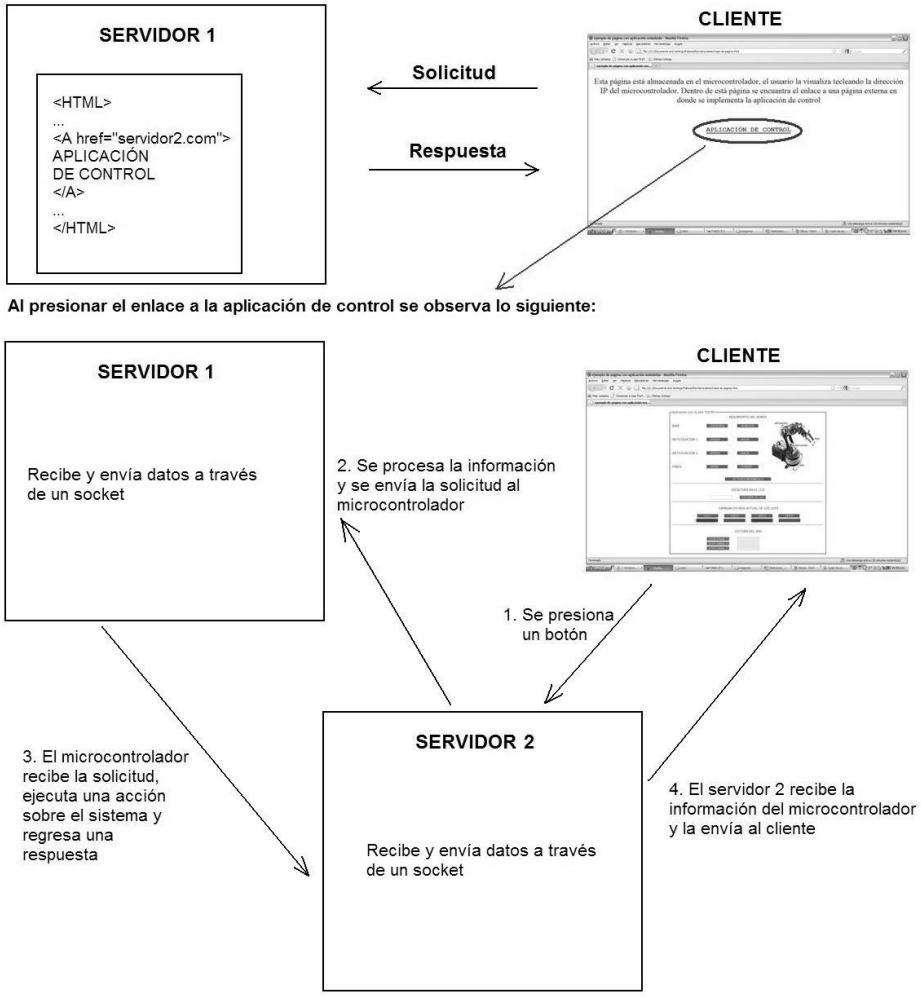

Figura 7.3 Ejemplo de aplicación almacenada y ejecutada por un servidor externo que se comunica con el cliente y con el microcontrolador

7.2.3 Implementación de la aplicación desde el servidor

Es posible implementar una aplicación que se ejecute desde el servidor (microcontrolador) y que genere de manera dinámica el contenido del recurso web que se transfiere al cliente. Esto se hace mediante el uso de una Interfaz de Entrada Común, o CGI (*Common Gateway Interface*).

Un CGI es una aplicación específica para ejecutar acciones en el lado servidor y regresar datos al cliente que hizo la petición desde un navegador.

La aplicación CGI se ejecuta en el microcontrolador y genera una respuesta HTML que se envía al destino a través de un socket para que un navegador web la interprete y la muestre. Para ejecutar el programa que genera la página en el microcontrolador, se teclea la dirección IP del microcontrolador desde el navegador web de la computadora y se especifica que el recurso que se solicita corresponde a un CGI. Cuando la solicitud llega al servidor, éste revisa si el recurso se encuentra y si es así se genera una página HTML. Si la página generada incluye acciones a realizar por medio de botones de un formulario las peticiones se realizan con el método GET de HTTP indicando en la URL el nombre de las variables y su valor correspondiente separados por un "&". Cuando se encuentra el recurso solicitado se ejecuta una función que extrae las variables contenidas en la petición junto con sus valores y ejecuta alguna acción dependiendo del valor de cada variable. Una petición de un recurso con datos enviados con el método GET se ve de la siguiente forma (el recurso solicitado en esta petición es *archivocgi* dentro de la carpeta *cgi*):

GET cgi/archivocgi?var1=23&var2=123

La cadena GET indica que existen datos desde un formulario con este método y a partir del signo "?" se encuentran las variables y sus valores, en el ejemplo se encuentra la primera variable (var1) que tiene un valor de 23 y la segunda variable (var2) tiene un valor de 123.

La forma de enviar los datos de regreso al cliente es mediante paquetes UDP desde HTTP, leyendo línea por línea el archivo CGI, con cuatro posibles acciones:

1) Ejecutando las funciones de C que corresponden al recurso CGI.
2) Enviando un texto (código HTML) directamente al navegador.
3) Incluyendo un archivo HTML previamente codificado.
4) Terminando la lectura de las líneas del archivo.

La estructura de un archivo CGI incluye los siguientes comandos:

t Se utiliza para indicar que el texto que le sigue en toda la línea se debe enviar al navegador tal como está escrito, por lo general con este comando se envía el código HTML para generar la página a presentar en el navegador.

i Se utiliza para incluir un archivo (previamente codificado en ASCII).

c Se utiliza para ejecutar un comando, el cual es un apuntador a una función de C que se requiere ejecutar.

. (punto) Se utiliza para indicar el final del archivo CGI.

Se utiliza para realizar comentarios, la línea que inicia con este carácter es ignorada.

El texto que se presenta en la Figura 7.4 es un ejemplo de cómo se ve el código fuente de un archivo CGI.

Ejemplo de un CGI

```
i /files_header.html
t <tr><td><a href="/index.html">/index.html</a></td><td>
t </td></tr> <tr><td><a href="/cgi/files">/cgi/files</a></td><td>
c b /cgi/files
t </td></tr> <tr><td><a href="/cgi/stats">/cgi/stats</a></td><td>
c b /cgi/stats
t </td></tr> <tr><td><a href="/cgi/tcp">/cgi/tcp</a></td><td>
c b /cgi/tcp
t </td></tr>
# Include the HTML footer.
i /files_footer.plain
# End of script.
.
```

Figura 7.4 Código de un archivo CGI

Para programar un CGI para microcontroladores y poder acceder a la aplicación se debe realizar lo siguiente:

- Escribir el archivo CGI en un editor de texto plano.
- Codificar en ASCII el archivo CGI y los archivos que incluye con el comando 'i'.
- Agregar el código generado en un archivo de C para que puedan ser leídas en el proyecto (como se explicó en el capítulo 5).
- Escribir el archivo en lenguaje C con las funciones a ejecutar cuando se usa el comando 'c'.
- Compilar el proyecto.

Ventajas:

- La aplicación se ejecuta en el microcontrolador y por lo tanto no se depende de servidores externos.

- No es necesario realizar configuraciones ni instalaciones en las computadoras cliente que se conectan al microcontrolador.

Desventajas:

- La pila que se usa para la comunicación debe tener la capacidad de leer archivos CGI.
- La aplicación tiene limitantes en el tamaño debido a la capacidad de almacenamiento del microcontrolador.
- Si se cambia el código de la aplicación cliente es necesario recompilar y volver a cargar el código en el microcontrolador.
- Cada programa CGI que se ejecuta en el microcontrolador usa un espacio de memoria propio. De esta forma, si tres usuarios ponen en marcha un mismo CGI a la vez se multiplicará por tres la cantidad de recursos que ocupe.

7.2.4 Aplicación ejecutable

Otra forma posible de implementar la aplicación cliente es mediante un archivo ejecutable que se instala en el cliente y se ejecuta desde ahí. Esta aplicación podría programarse por ejemplo en Visual Basic, como en el método que se explicó en el tema 7.2.1, pero a diferencia de dicho método el código no se transfiere desde el servidor, sino que se instala directamente en el cliente y no necesariamente debe funcionar en un navegador. Esta aplicación se comunica con el microcontrolador a través de una red haciendo uso de sockets, pero toda la funcionalidad se encuentra en el lado del cliente. Se puede programar en lenguajes como Visual Basic, dotNet, Java, entre otros.

Ventajas:

- La aplicación es fácil de programar si se conoce algún lenguaje de programación de interfaces gráficas y no es necesario codificar la aplicación para almacenarla en el microcontrolador.
- La aplicación se puede modificar en cualquier momento sin afectar el código del microcontrolador.

Desventajas:

- Es necesario instalar la aplicación en cada una de las máquinas que se comunican con el microcontrolador.

7.3 Ejemplo de aplicación con el sistema de desarrollo

En este tema se presenta un ejemplo de una aplicación a través de una red con el propósito de facilitar la comprensión de los temas antes expuestos.

7.3.1 Descripción de la aplicación

La aplicación del lado servidor está programada en lenguaje C haciendo uso de las funciones de la pila TCP/IP desarrollada y explicada en capítulos anteriores.

El método a utilizar para la programación en el lado del cliente es el que se explicó en el Tema 7.2.2 "Programación de la aplicación cliente en un servidor externo". El lenguaje a utilizar es PHP, usado para la programación de páginas web dinámicas y que cuenta con una librería de manejo de sockets para la comunicación entre redes TCP/IP.

La aplicación de ejemplo se divide en 5 secciones listadas a continuación:

1. **Simulación de un proceso industrial mediante el movimiento de un brazo mecánico.**

   Esta parte de la aplicación simula un proceso industrial para mover objetos de un lugar a otro con un brazo mecánico (procesos similares usan brazos mecánicos en ambientes hostiles para el ser humano, como en la industria química). Este proceso se realiza enviando comandos desde el navegador.

   Para la simulación se utiliza un brazo mecánico de juguete que cuenta con tres motores para posicionarlo y un cuarto motor que manipula una pinza en su extremo para sujetar objetos. Este

robot se conecta a la tarjeta en uno de sus puertos libres por medio de relevadores que manipulan el movimiento de los motores. El uso de los relevadores permite conectar los motores a una fuente interna incluida en el robot y así se separa la etapa de potencia del robot de la alimentación del sistema de control. Esta conexión le brinda protección al sistema de control, ya que la corriente que requieren los motores puede resultar excesiva para ser proporcionada por el sistema de control.

El robot que se usa para la aplicación es el modelo k–680 de la compañía *Steren*. Este robot se maneja de manera cableada y puede levantar objetos de peso menor a 100 gramos. La base del robot puede girar horizontalmente hasta 270°. Encima de la base se encuentra una articulación que puede girar verticalmente hasta 180°. La siguiente articulación corresponde a la parte superior del brazo y tiene un movimiento vertical de hasta 300°. La última articulación se encuentra en la pinza y puede moverse de manera vertical 120°. Los movimientos descritos anteriormente se ilustran en la Figura 7.5.

Figura 7.5 Movimiento del robot utilizado para la demostración

2. **Escribir en el LCD.** Esta parte de la aplicación envía texto desde el navegador y lo presentar en el display conectado a la tarjeta.

3. **Cambiar el estado de los leds de la tarjeta.** Esta parte de la aplicación cambia el estado actual de los leds. Si un led de la tarjeta se encuentra encendido y se envía el comando para que cambie de estado se apaga y si se encuentra apagado entonces se enciende.

4. **Lectura de valores en sensores conectados a la tarjeta.** Esta parte de la aplicación lee el voltaje de tres canales del Convertidor Analógico-Digital. El canal 0 del ADC está conectado a una resistencia variable, el canal 1 a una fotoresistencia (ver Figura 3.41) y el canal 2 a un sensor de temperatura (ver Figura 3.44).

5. **Emisión de sonido en el *buzzer* de la tarjeta.** Esta parte de la aplicación hace que el buzzer emita uno de cuatro sonidos predeterminados al enviarle los comandos disponibles para dicha acción.

Las aplicaciones que se pueden implementar con este sistema pueden ser tan diversas como se requiera, teniendo en cuenta las capacidades del microcontrolador.

7.3.2 Especificaciones de la aplicación

7.3.2.1 Configuración de los puertos

En la Tabla 7-1 se presentan las configuraciones realizadas y los puertos del microcontrolador que se usan para cada parte de la aplicación.

Tabla 7-1 Configuraciones realizadas para cada parte de la aplicación de control

Sección	Descripción
Movimiento del brazo mecánico	El robot se conecta en el puerto B, el cual se configura como salida para enviar las señales de control que abren y cierran los relevadores que a su vez manipulan el movimiento de los motores. Para controlar el movimiento y sentido de giro de los cuatro motores se usan cinco relevadores. El relevador k0 selecciona el sentido de giro para cualquiera de los cuatro motores, permitiendo

	que el motor se conecte a un voltaje positivo o negativo para cambiar el sentido de giro. Cada uno de los otros cuatro relevadores se cierran o abren para encender alguno de los motores conectados a la salida del primer relevador k0 (ver Figura 7.8).
Cambiar el estado de los leds de la tarjeta	Los leds del microcontrolador están conectados en la parte alta del puerto K y esta parte se configura como salida.
Lectura de valores en sensores conectados a la tarjeta	Los dispositivos conectados al convertidor analógico digital ocupan los canales 0, 1 y 2, correspondientes a un potenciómetro, sensor de luminosidad y sensor de temperatura respectivamente. Al recibir un comando se debe configurar la lectura del canal correspondiente y regresar el valor leído.
Escribir texto al LCD	El display se encuentra conectado en el puerto H y es necesario configurar el puerto como salida así como realizar la inicialización del display por software antes de enviarle datos. Junto con este comando se debe agregar el dato que se quiere desplegar en el display, considerando que se tiene un display de 16 caracteres por dos líneas, el número máximo de caracteres que se toman en cuenta para mostrar el mensaje en el display es 32.
Emisión de sonido en el *buzzer* de la tarjeta	El buzzer está conectado al bit 1 del puerto J por lo que este bit se debe habilitar como salida.

7.3.2.2 Comandos, puertos y protocolo de transporte.

La programación de la aplicación requiere definir las acciones que se requiere que el microcontrolador ejecute y crear comandos para cada una de ellas, asignando un código numérico para cada uno. Cada comando tiene un propósito en específico y se envía a través de un puerto de la capa de transporte para ser procesado por la aplicación que corresponde. La relación de comandos y puertos para la aplicación es la que se presenta en la Tabla 7-2.

La codificación de cada comando debe ser enviada al puerto respectivo en formato hexadecimal ya que así se ha definido en la aplicación del lado servidor. Por facilidad de programación el protocolo usado es UDP.

Tabla 7-2 Puertos que utiliza la aplicación y sus comandos

Sección de la aplicación	Puerto	Comando	Código	Descripción del comando
Movimiento del brazo mecánico	2007	0	0000	Ejecutar un movimiento predeterminado partiendo de la posición de inicio
		1	0001	Mover la base a la izquierda
		2	0010	Mover la base a la derecha
		3	0011	Mover la articulación 1 hacia arriba
		4	0100	Mover la articulación 1 hacia abajo
		5	0101	Mover la articulación 2 hacia arriba
		6	0110	Mover la articulación 2 hacia abajo
		7	0111	Abrir la pinza
		8	1000	Cerrar la pinza
		9	1001	Detener el movimiento
		10	1010	Regresar a la posición de inicio después del comando 0
Escribir texto al LCD	2005	0	0000	Escribir en el display un texto determinado (este texto se debe agregar a los datos enviados después del comando y como máximo 32 caracteres)
Cambiar el estado de los leds de la tarjeta	2000	4	0100	Cambiar el estado del led ubicado en el bit 4 del puerto K
		5	0101	Cambiar el estado del led ubicado en el bit 5 del puerto K
		6	0110	Cambiar el estado del led ubicado en el bit 6 del puerto K

		7	0111	Cambiar el estado del led ubicado en el bit 7 del puerto K
Lectura de valores en sensores conectados a la tarjeta	2006	0	0000	Obtener el voltaje de una resistencia variable conectada en el canal 0 del convertidor Analógico-Digital
		1	0001	Obtener el voltaje de la fotoresistencia conectada en el canal 1 del convertidor Analógico-Digital
		2	0010	Obtener los grados Celsius del sensor conectado en el canal 2 del convertidor Analógico-Digital
Emisión de sonido en el buzzer de la tarjeta	2000	0	0000	Emitir el sonido predeterminado 1 en el buzzer
		1	0001	Emitir el sonido predeterminado 2 en el buzzer
		2	0010	Emitir el sonido predeterminado 3 en el buzzer
		3	0011	Emitir el sonido predeterminado 4 en el buzzer

7.3.2.3 Archivo de la aplicación en el proyecto de CodeWarrior

Para crear las funciones de la aplicación se creó el archivo udp_user_application.c (Figura 7.6) que contiene todas las funciones de la aplicación, incluyendo la función udp_app_server() que llama a las funciones de cada sección de la aplicación dependiendo del comando recibido y es llamada desde la función principal del servidor (microcontrolador).

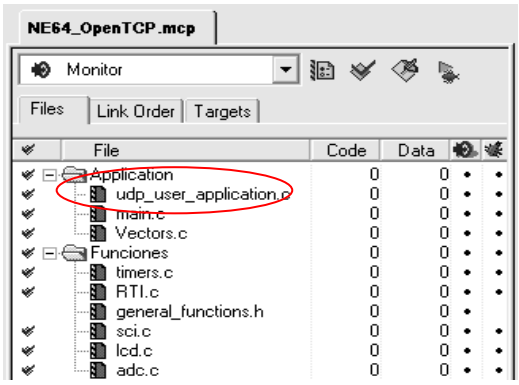

Figura 7.6 Archivo udp_user_application.c del proyecto en CodeWarrior

7.3.3 Aplicación del lado servidor

Ahora que se tienen los comandos definidos y se ha escogido un protocolo de transporte, se puede programar la aplicación tanto en el cliente como en el servidor.

En este tema se analiza la aplicación que corre en el microcontrolador como lado servidor.

Primero se inicializa el servicio, obteniendo y abriendo un socket UDP por cada aplicación con los parámetros necesarios. La función que hace esto se llama udp_app_server() y debe ser llamada por la función principal después de haber inicializado los sockets UDP. En esta función también se inicializan los dispositivos y puertos para las acciones del robot, los leds, el buzzer y el display. El convertidor analógico digital se configura en el momento de realizar la lectura. La función de inicialización no recibe argumentos ni tiene valor de retorno. En la Figura 7.7 presenta el contenido de la función udp_app_server().

```
void udp_app_server(){
  char socket1, socket2, socket3, socket4;
  socket1 = udp_get_sock();//obtener un socket para la comunicación
    //abrir el socket como aplicación servidor cambio de estado de leds y buzzer
  udp_open_sock(socket1,UDP_SERVER, 0, 0, UDP_LED_BUZZER_PORT, udp_led_buzzer_app);
  DDRK |= 0xF0;  // todos los pin del puerto K como salida
  DDRJ |= 0xF2;  // parte alta y bit 2 del puerto J como salida

  socket2 = udp_get_sock();//obtener socket para aplicacion con LCD
  //abrir el socket como aplicación servidor
```

```
  udp_open_sock(socket2,UDP_SERVER, 0, 0, UDP_LCD_PORT, udp_lcd_app);
  lcd_init();

  socket3=udp_get_sock();//obtener un socket para la aplicacion ADC
  //abrir el socket como aplicacion servidor
  udp_open_sock(socket3, UDP_SERVER, 0, 0, UDP_ADC_PORT, udp_adc_app);

  socket4=udp_get_sock();//obtener un socket para la aplicacion robot
  //abrir el socket como aplicacion servidor
  udp_open_sock(socket4, UDP_SERVER, 0, 0, UDP_ROBOT_PORT, udp_robot_app);
  DDRB=0xFF; //salidas
}
```

Figura 7.7 Código de la función udp_server_app

El último argumento de la función udp_open_sock() corresponde al nombre de la función que procesa los datos recibidos que a su vez recibe tres parámetros correspondientes al número de socket que recibe los datos, la dirección de los datos en el buffer de recepción y su longitud.

En cada función se realizan validaciones de los datos, como por ejemplo la longitud para verificar que el paquete recibido contiene información. En caso de cualquier anomalía la función termina su ejecución y no se procesan los datos. Si la validación resulta correcta se inicializa la lectura del buffer de recepción por medio de las funciones del controlador de red y el parámetro recibido que indica la posición de los datos. Ya que nos situamos en la posición del buffer donde comienzan los datos que nos interesa leer, se almacena el primer valor del paquete recibido en una variable de ocho bits, se lee su valor para determinar el comando que se ha recibido y se toma una acción basada en las acciones de control definidas por los comandos y el puerto.

7.3.3.1 Movimiento del brazo mecánico

La función que mueve el brazo mecánico utiliza el puerto 2007 para recibir los comandos. La conexión entre el puerto B del microcontrolador (salidas para el movimiento del brazo) y los motores del robot es como se muestra en la Tabla 7-3.

Tabla 7-3 Conexión entre el puerto B y las señales de mueven del robot

Puerto B	Señal que manipula	Descripción
PB0	Dirección de los motores	Esta salida controla el relevador k0 que selecciona el nivel de voltaje al que se conectan los otros relevadores para indicar el sentido de giro. Este voltaje proviene de la fuente interna del robot y es un voltaje de cero o 6 volts. Cada uno de los motores tiene una terminal conectada a un nivel de 3 volts y la otra terminal se conecta a cero volts para girar en un sentido o a 6 volts para cambiar de sentido (ver Figura 7.8).
PB1	Motor 1	Esta salida controla el relevador k1 que conecta el motor de la base a la salida del relevador de dirección. Cuando se le manda una salida en nivel lógico uno a esta terminal el motor se conecta al nivel de voltaje seleccionado por el relevador k0 y la base gira en una dirección. Cuando se le manda una salida en nivel lógico cero el relevador se desconecta de la fuente y el motor queda desconectado y sin movimiento (ver Figura 7.8).
PB2	Motor 2	Esta salida controla el relevador k2 que conecta el motor de la parte inferior del brazo a la salida del relevador de dirección. Cuando se le manda una salida en nivel lógico uno a esta terminal el motor se conecta al nivel de voltaje seleccionado por el relevador k0 y la articulación sube o baja. Cuando se le manda una salida en nivel lógico cero el relevador se desconecta de la fuente y el motor queda desconectado y sin movimiento (ver Figura 7.8).
PB3	Motor 3	Esta salida controla el relevador k3 que conecta el motor de la parte superior del brazo a la salida del relevador de dirección. Cuando se le manda una salida en nivel lógico uno a esta terminal el motor se conecta al nivel de voltaje seleccionado

		por el relevador k0 y la articulación sube o baja. Cuando se le manda una salida baja el relevador k3 se desconecta de la fuente y el motor queda desconectado y sin movimiento (ver Figura 7.8).
PB4	Motor 4	Esta salida controla el relevador k4 que conecta el motor de la pinza del robot a la salida del relevador de dirección. Cuando se le manda una salida en nivel lógico uno a esta terminal el motor se conecta al nivel de voltaje seleccionado por el relevador k0 y la pinza abre o cierra. Cuando se le manda una salida en nivel lógico cero el relevador k4 se desconecta de la fuente y el motor queda desconectado y sin movimiento (ver Figura 7.8).

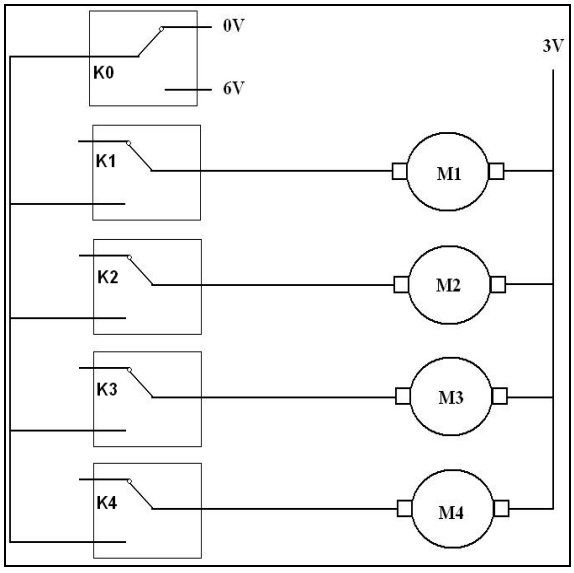

Figura 7.8 Conexión de los motores del robot mediante relevadores

En la función udp_robot_ap()p se lee el comando que incluye la solicitud del cliente y de acuerdo a esta información se toma la acción correspondiente sobre las salidas del puerto B. En la Figura 7.9 se presenta el fragmento de código que activa los motores del brazo mecánico a partir del valor del comando leído. Los bits del puerto B se manipulan para que se mueva la articulación deseada en el sentido deseado de acuerdo a lo indicado en la Tabla 7-3.

```
stop();//detener motores
switch(command){
case  '10': //caso regresar al inicio
      art1_up(2250);
      art2_up(6250);
      gripp_close(950);
      base_izq(6200);
    break;
case '1': //caso base izquierda
        PORTB_BIT1=1;
        PORTB_BIT0=1;
    break;
case '2': //caso base derecha
        PORTB_BIT1=1;
        PORTB_BIT0=0;
    break;
case '3': //caso art1 arriba
        PORTB_BIT2=1;
        PORTB_BIT0=0;
    break;
 case '4': //caso art1 abajo
        PORTB_BIT2=1;
        PORTB_BIT0=1;
    break;
 case '5': //caso art2 arriba
```

```
        PORTB_BIT3=1;
        PORTB_BIT0=1;
    break;
case '6': //caso art2 abajo
        PORTB_BIT3=1;
        PORTB_BIT0=0;
    break;
case '7': //caso gripp abrir
        PORTB_BIT4=1;
        PORTB_BIT0=1;
    break;
case '8': //caso gripp cerrar
        PORTB_BIT4=1;
        PORTB_BIT0=0;
    break;
 case '9': //caso detener
       stop(); //detener motores
    break;
case '0':  //caso movimiento precalculado
       base_izq(6350);
       art1_down(2500);
       gripp_open(1000);
       art2_down(3500);
       gripp_close(950);
       art2_up(2500);
       art1_up(2500);
       base_der(11050);
       art1_down(1500);
       art2_down(2300);
       gripp_open(1000);
       break;
    }
```

Figura 7.9 Fragmento de código de la función udp_robot_app

7.3.3.2 Escribir en el LCD

La función udp_lcd_app() lee el comando 0 y escribe el texto en el display. En la Figura 7.10 se presenta el fragmento de código que escribe los datos enviados al display al recibirlos por red. Se lee carácter por carácter y se envía al display, al mismo tiempo se guarda en un arreglo para enviar la misma cadena de regreso al navegador.

```
for(i=0;i<datalen;i++){
   string = NE64ReadByte();//se lee character por caracter
   data_buff[j++]=string;   //se guarda el character en un arreglo buffer para regresarlo al cliente
   lcd_data(string);   //se escribe el caracter
}
data_buff[j++]=' ';
data_buff[j++]='O';
data_buff[j++]='k';
data_buff[j++]='!';     //se agrega la cadena ' OK!' a los datos
udp_send(socket,data_buff,datalen+4);//enviar a través de UDP
```

Figura 7.10 Fragmento de código de la función udp_lcd_app

7.3.3.3 Cambio de estado de los leds

La función udp_led_buzzer_app() lee el comando enviado y dependiendo de su valor cambia el estado del led indicado. La Figura 7.11 presenta el fragmento de código que cambia el estado del led.

```
if(command=='4'){
     PORTK_BIT4=~PORTK_BIT4; //cambia el estado del bit 4 de Puerto K
   }else if(command=='5'){
     PORTK_BIT5=~PORTK_BIT5; //cambia el estado del bit 5 de Puerto K
   }else if(command=='6'){
     PORTK_BIT6=~PORTK_BIT6; //cambia el estado del bit 6 de Puerto K
   }else if(command=='7'){
     PORTK_BIT7=~PORTK_BIT7; //cambia el estado del bit 7 de Puerto K
   }
```

Figura 7.11 Fragmento de código de la función udp_led_buzzer_app

7.3.3.4 Lectura de un canal de ADC

La función udp_adc_app lee el comando enviado desde el navegador, llama a la función read_adc() que almacena el comando en una variable de un byte y dependiendo su valor configura el convertidor Analógico-Digital para leer el canal solicitado y formar un paquete UDP con el valor leído. Cabe aclarar que los datos leídos por el ADC son cantidades numéricas que pueden representar variables físicas como niveles de voltaje y es necesario hacer la conversión al dato requerido. Esto se puede implementar en el programa del microcontrolador antes de que el dato sea enviado, o puede enviarse el valor numérico y hacer la conversión en el programa cliente que muestra el resultado al usuario. En este caso sólo se envía el valor leído del ADC como una cadena de caracteres que representa un número en hexadecimal (el cual se encuentra entre 0x0000 y 0x03FF, ya que el convertidor se configura en tamaño de 10 bits) y la aplicación del lado del cliente convierte el valor leído al valor correspondiente a la variable física.

La función adc_convert() recibe el canal del ADC que se va a leer, configura la conversión y regresa una variable entera que corresponde al valor leído y la función chartohex() recibe una variable de ocho bits, de los cuales la parte alta es cero y la parte baja se codifica en un carácter entre 0 y 9 ó entre A y F.

```
dato=adc_convert(argumento);
//se forma el paquete UDP con una cadena que representa el número leído
en hexadecimal
      data_buff[i++]=`0`;
data_buff[i++]=`x`;
data_buff[i++]=`0`;              //ejemplo: dato=0x03A8
data_buff[i++]=chartohex(dato>>8); // dato>>8=0x03A8>>8=0x0003
tmp=dato>>4; //tmp=0x03A8>>4=0x003A
data_buff[i++]=chartohex(tmp&0x000F);  //tmp=0x000A
data_buff[i++]=chartohex(dato&0x000F;) //0x0008
//se forma un paquete UDP con los 6 caracteres que representan el valor
hexadecimal del dato
udp_send(socket,data_buff,6);
```

Figura 7.12 Fragmento de la función udp_adc_app

7.3.3.5 Sonido en el buzzer

La función udp_led_buzzer_app() lee un comando y a partir de su valor se hace variar el valor del bit 1 del puerto J fijando su valor en uno durante algún tiempo y después asignándole el valor de cero durante otro periodo de tiempo, para emitir un sonido. La Figura 7.13 presenta el fragmento de código en el que se llama a la función buzzer con diferente argumento dependiendo del valor del comando y la función buzzer que genera el estado en 0 y 1 del bit.

```
if(command=='0'){
     buzzer(10);
   }else if(command=='1'){
     buzzer(15);
   }else if(command=='2'){
     buzzer(20);
   }else if(command=='3'){
     buzzer(25);
   }

 void buzzer(unsigned char i){
  unsigned int j,k,m;
  j=i*10;
  for(m=0;m<=1000;m++){
    PTJ_PTJ1=0;
    for(k=0;k<j+50;k++);
    PTJ_PTJ1=1;
    for(k=0;k<j+100;k++);
  }
  PTJ_PTJ1=0;
}
```

Figura 7.13 Fragmento de código de la función udp_led_buzzer_app

De esta manera queda configurada y programada una aplicación en el servidor (microcontrolador) que muestra cómo manipular señales de entrada y salida del microcontrolador. Al generalizar el procedimiento aquí mostrado se pueden lograr aplicaciones muy potentes que hagan toda clase de manipulaciones sobre sistemas físicos controlados a distancia.

7.3 Aplicación del lado cliente

La aplicación cliente se programa en PHP y se ejecuta en una computadora de la red local que tenga instalado el software necesario para funcionar como servidor de páginas web.

El lenguaje de programación PHP cuenta con una librería de funciones para manejo de sockets que permiten programar páginas web que interactúan de manera dinámica con equipos remotos a través de las funciones de la pila TCP/IP. Las funciones que son de interés para la aplicación son las presentadas en la Tabla 7-4.

Tabla 7-4 Funciones de PHP para manejo de sockets [26e]

Función	Descripción
socket_create	Crea un socket
socket_conect	Realiza una conexión activa a un servidor
socket_send	Envía datos a través de un socket conectado (TCP)
socket_sendto	Envía un mensaje a un socket que puede estar conectado o no (TCP o UDP)
socket_recv	Recibe datos de un socket conectado (TCP)
socket_recvfrom	Recibe datos de un socket que puede estar conectado o no (TCP o UDP)
socket_close	Cierra un socket

En la aplicación cliente lo primero que se hace es obtener un socket que va a ser utilizado para comunicarnos con el microcontrolador. El socket que se obtiene es de tipo UDP. Los parámetros que recibe la función son el dominio, el tipo de socket y el protocolo de transporte que se va a emplear. En la Figura 7.14 se muestra la obtención del socket, la variable $sock es el identificador de socket que se usa para la comunicación.

```
$sock = socket_create(AF_INET, SOCK_DGRAM, SOL_UDP);
```

Figura 7.14 Función de PHP para obtener un socket

Para enviar datos se utiliza la función socket_sendto, a la cual se le indica el destino de la información cada vez que se invoca, ya que el protocolo UDP no maneja conexiones. La información que se envía corresponde al comando y se debe enviar al puerto adecuado. Pensemos por ejemplo que se presiona un botón en la página que indica que se mueva el motor de la base del robot a la derecha. El comando corresponde al número 2 de la aplicación que se atiende en el puerto 2007, de acuerdo a los comandos definidos en la Tabla 7-2. En las Figura 7.15 se muestra el envío del paquete.

```
$datos=0x02; // comando 2 de la aplicación ROBOT
socket_sendto($sock, $datos, 1, 0, '192.168.2.3', 2007);
```

Figura 7.15 Envío de los datos a través del socket obtenido

Los parámetros que recibe la función socket_sendto son el identificador de socket usado para la comunicación, la variable que contiene los datos, la longitud de datos que se van a enviar, una bandera (en este caso es cero), la dirección IP del servidor (la que se le asignó al microcontrolador) y el número de puerto que se usa para la aplicación. Los datos que se envían por el socket dependen del comando y se envían en formato hexadecimal.

Cuando la aplicación requiere leer datos que envía el microcontrolador, por ejemplo la cadena de 6 caracteres que representa el dato que se lee del convertidor analógico digital, se emplea la función socket_read. Esta función recibe el identificador de socket y la cantidad de bytes que se deben leer, el valor que regresa es una cadena que contiene los datos leídos. En la Figura 7.16 se muestra la programación del comando de lectura del ADC en la aplicación cliente.

```
$datos=0x02; // comando 2. Leer canal 2 del ADC
socket_sendto($socket,$datos,1,0,"192.168.2.3",2006);
$adc_string=socket_read($socket,6);
```

Figura 7.16 Programación del comando de lectura del ADC

En la Figura 7.17 se muestra la interfaz de la aplicación cliente, cada uno de los botones tiene programado el envío del comando correspondiente.

a)

b)

c)

d)

e)

Figura 7.17 Ejemplo de interfaz de usuario programada en PHP. a) Movimiento del brazo mecánico. b) Escritura en el LCD. c)Cambio de estado de los LEDS. d) Lectura del ADC. e) Sonido en el buzzer.

El lado cliente de esta aplicación requiere una lógica más sencilla que la requerida del lado servidor ya que consiste en el envío de los comandos (en este caso al puerto específico de acuerdo a la Tabla 7-2) y la recepción y presentación de los datos que envía el servidor. En este punto se debe reiterar que la aplicación

puede ser programada con cualquier lenguaje de programación que pueda manejar sockets ya que es por este medio que se envían los comandos y se reciben las respuestas.

7.4 Ejemplo de comunicación entre dos tarjetas de desarrollo

Esta parte de la aplicación consiste en comunicar dos tarjetas (sistemas con el microcontrolador MC9S12NE64) mediante la red. Ambas tarjetas tienen cargada la pila TCP/IP, una de ellas contiene la aplicación cliente y la otra el servidor. Al accionar alguna tecla de la tarjeta cliente (teclado matricial) se envían mediante el protocolo UDP los comandos que accionan las articulaciones del brazo mecánico, conectado al puerto B de la tarjeta servidor.

La conexión de las dos tarjetas puede realizarse mediante un cable cruzado conectando cada extremo del cable a una tarjeta o se pueden conectar ambas tarjetas a un *switch* o un *hub* usando cables UTP5 con terminales uno a uno.

7.4.1 Aplicación del lado cliente

La aplicación del lado cliente consiste en un ciclo infinito que espera a que alguna tecla sea presionada, cuando esto ocurre el comando asignado al botón es enviado al puerto 2007 del servidor desde un socket previamente abierto. En la Tabla 7-5 se muestran los comandos y su descripción para el ejemplo de comunicación entre las dos tarjetas.

Tabla 7-5 Puertos que utiliza la aplicación y sus comandos para la comunicación entre las dos tarjetas

APLICACIÓN	PUERTO	COMANDO	CÓDIGO	DESCRIPCIÓN DEL COMANDO
Movimiento del brazo mecánico	2007	1	0001	Mover la base a la izquierda
		2	0010	Mover la base a la derecha
		3	0011	Mover la articulación 1 hacia arriba
		4	0100	Mover la articulación 1 hacia abajo
		5	0101	Mover la articulación 2 hacia arriba
		6	0110	Mover la articulación 2 hacia abajo
		7	0111	Abrir la pinza
		8	1000	Cerrar la pinza
		9	1001	Detener el movimiento

En Figura 7.18 se presenta el fragmento del código que lee el valor del botón presionado y envía el comando al servidor.

```
socketUDP=udp_get_sock(); //obtener un socket para enviar y recibir datos
i=UDP_HEADER_OFFSET;//inicio de los datos en buffer de transmisión
//se abre el socket para enviar datos la Puerto 2007 del servidor
udp_open_sock(socketUDP,UDP_CLIENT,ip_rem,2007,2000,(void*)0);
for(;;){//inicia el ciclo infinito para esperar a que algún botón se presione y enviar el comando respectivo
    tecla=leer_tecla();    //obtener numero de tecla presionada
    switch(tecla){
      case 1:data_buff[i]='1'; //comando 1, base a la izquierda
      break;
      case 2:data_buff[i]='2'; //comando 2, base a la derecha
      break;
      case 3:data_buff[i]='3'; //comando 3, articulación 1 arriba
      break;
      case 5:data_buff[i]='4'; //comando 4, articulación 1 abajo
      break;
      case 6:data_buff[i]='5'; //comando 5, articulación 2 arriba
      break;
      case 7:data_buff[i]='6'; //comando 6, articulación 2 abajo
      break;
      case 9:data_buff[i]='7';  //comando 7 abrir la pinza
      break;
```

```
      case 10:data_buff[i]='8'; //comando 8 cerrar la pinza
       break;
       default:data_buff[i]='9'; //default detener movimiento
       break;
     }
     udp_send(socketUDP,data_buff,1);// se envía el comando al servidor
     if(NE64ValidFrameReception()){//si existe respuesta, esta se procesa
         ethernet_receive();
     }
    arp_manage();
}//fin for
```

Figura 7.18 Fragmento de código del lado cliente de la comunicación entre dos tarjetas

7.4.2 Aplicación del lado servidor

El lado servidor responde a los comandos enviados por el cliente dependiendo de su valor según se indica en la Tabla 7-5. En la Figura 7.19 se muestra el código que ejecuta la acción requerida dependiendo del valor del comando enviado, accionando los motores del brazo mecánico.

```
switch(command){
case '1': //caso base izquierda
        PORTB_BIT1=1;
        PORTB_BIT0=1;
    break;
case '2': //caso base derecha
        PORTB_BIT1=1;
        PORTB_BIT0=0;
    break;
case '3': //caso art1 arriba
        PORTB_BIT2=1;
        PORTB_BIT0=0;
    break;
 case '4': //caso art1 abajo
        PORTB_BIT2=1;
        PORTB_BIT0=1;
    break;
 case '5': //caso art2 arriba
        PORTB_BIT3=1;
        PORTB_BIT0=1;
    break;
case '6': //caso art2 abajo
        PORTB_BIT3=1;
        PORTB_BIT0=0;
    break;
case '7': //caso gripp abrir
        PORTB_BIT4=1;
        PORTB_BIT0=1;
    break;
case '8': //caso gripp cerrar
        PORTB_BIT4=1;
        PORTB_BIT0=0;
    break;
 case '9': //caso detener
       stop(); //detener motores
    break;
    }
```

Figura 7.19 Fragmento de código del lado servidor para la comunicación entre 2 tarjetas

La comunicación entre las dos tarjetas es de gran utilidad en algún proceso industrial en el que el cliente, basado en las mediciones recibidas, le indique al servidor que ejecute alguna acción o que le envíe el valor de alguna variable del proceso para tomar alguna decisión. Las aplicaciones que se presentaron en este tema son ejemplos de lo que se puede realizar con el sistema desarrollado. Como ya se había dicho anteriormente, la programación del lado cliente y del lado servidor dependen de las necesidades del proceso pero las capacidades del sistema son lo suficientemente amplias para el monitoreo o control de dicho proceso.