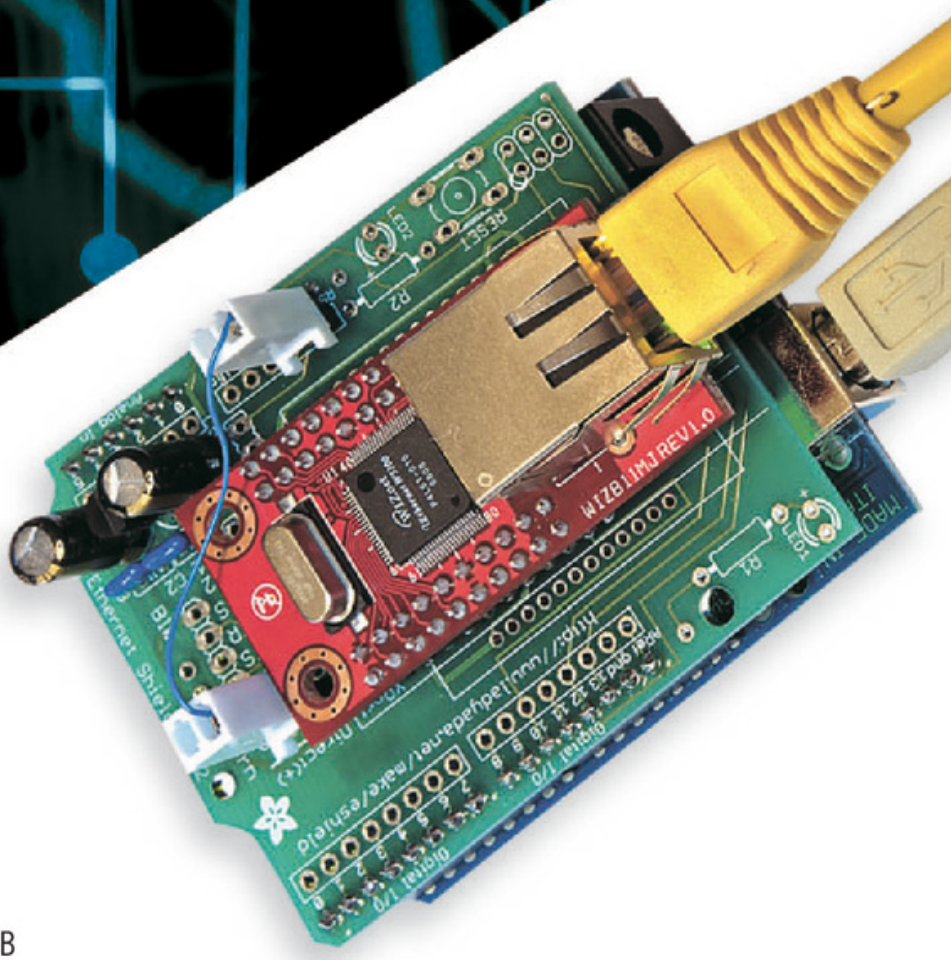
USERS

¡CONTIENE EJEMPLOS PRÁCTICOS PARA PONER MANOS A LA OBRA!

NETWORKING CON MICROCONTROLADORES

DESCUBRA CÓMO ACCEDER REMOTAMENTE A SUS EQUIPOS



ADEMÁS

- NETWORKING CON TCP/IP
- ETHERNET Y WI-FI CON MÓDULOS RABBIT
- MULTITAREA COOPERATIVO
- MONITOREO REMOTO MEDIANTE PÁGINAS WEB

COLECCIÓN USERS ELECTRÓNICA

FUNCIONAMIENTO, PROGRAMACIÓN Y APLICACIONES PRÁCTICAS



» HARDWARE
» 192 PÁGINAS
» ISBN 978-987-1773-22-0

Esta obra presenta las diferencias entre los sistemas analógicos y los digitales. Además, analizaremos los conceptos y los dispositivos fundamentales en los que se basa la electrónica digital, ingresaremos en el mundo de las memorias y aprenderemos cómo programar microcontroladores.

SOBRE LA COLECCIÓN: USERS ELECTRÓNICA

- Aprendizaje guiado mediante explicaciones claras y concisas
- Proyectos prácticos basados en necesidades reales
- Consejos de los profesionales
- Producciones fotográficas profesionales
- Infografías y procedimientos paso a paso



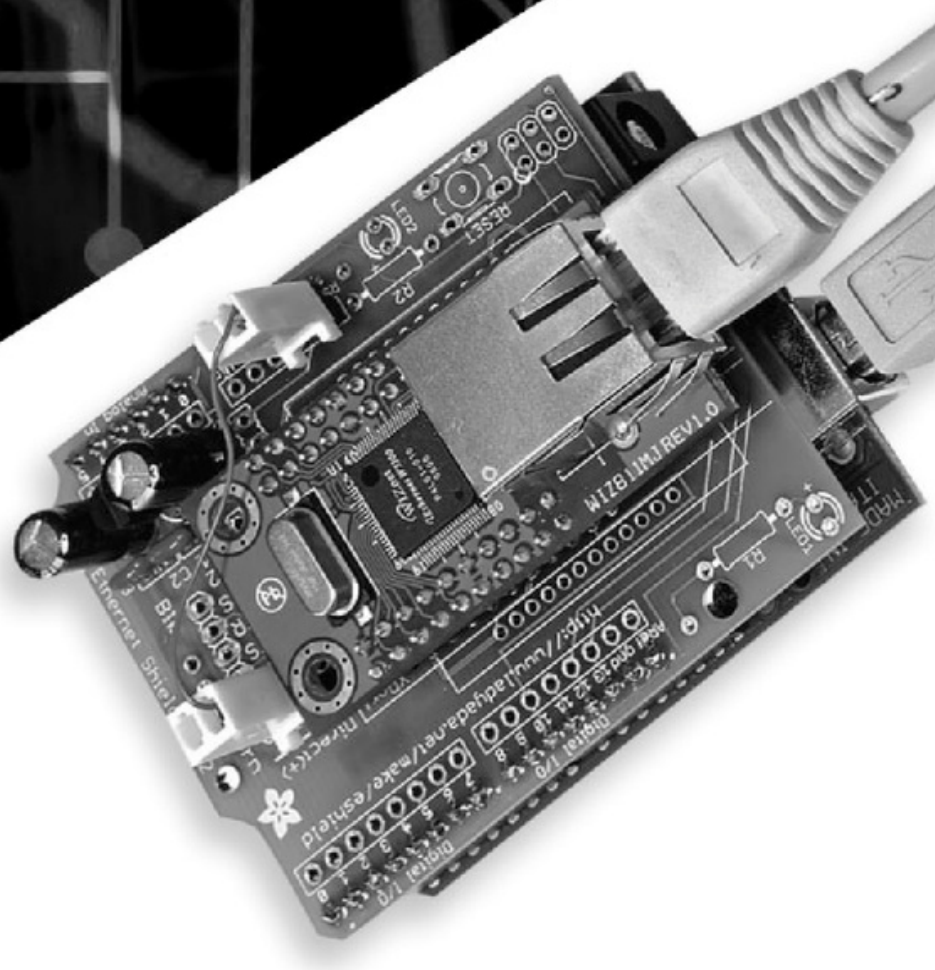
LLEGAMOS A TODO EL MUNDO VÍA OCA * Y DHL **

* SÓLO VÁLIDO EN LA REPÚBLICA ARGENTINA // ** VÁLIDO EN TODO EL MUNDO EXCEPTO ARGENTINA

usershop.redusers.com // usershop@redusers.com

NETWORKING CON MICROCONTROLADORES

DESCUBRA CÓMO ACCEDER REMOTAMENTE A SUS EQUIPOS





TÍTULO: Networking con microcontroladores
COLECCIÓN: desde Cero
FORMATO: 15 x 19 cm
PÁGINAS: 192

Copyright © MMXI. Es una publicación de Fox Andina en coedición con DALAGA S.A. Hecho el depósito que marca la ley 11723. Todos los derechos reservados. Esta publicación no puede ser reproducida ni en todo ni en parte, por ningún medio actual o futuro sin el permiso previo y por escrito de Fox Andina S.A. Su infracción está penada por las leyes 11723 y 25446. La editorial no asume responsabilidad alguna por cualquier consecuencia derivada de la fabricación, funcionamiento y/o utilización de los servicios y productos que se describen y/o analizan. Todas las marcas mencionadas en este libro son propiedad exclusiva de sus respectivos dueños. Impreso en Argentina. Libro de edición argentina. Primera impresión realizada en Sevagraf, Costa Rica 5226, Grand Bourg, Malvinas Argentinas, Pcia. de Buenos Aires en X, de MMXI.

ISBN 978-987-1773-24-4

Networking con microcontroladores / coordinado por Daniel Benchimol. -
1a ed. - Buenos Aires: Fox Andina; Dalaga, 2011.
192 p. ; 19x15 cm. - (Desde cero; 21)
ISBN 978-987-1773-24-4

1. Informática. I. Daniel Benchimol, coord.
CDD 005.3

Prólogo **al contenido**

Actualmente, estar conectados en red es fundamental para nuestra vida diaria. El objetivo principal de una red es que se puedan compartir recursos e información entre todos sus elementos y tener flexibilidad para así optimizar tareas o procesos que los usuarios realizan. Las redes de computadoras evolucionan para obtener mayor movilidad y rendimiento de las tareas.

La tecnología WiFi nos permite conectar dispositivos entre sí para formar una red inalámbrica. Las ventajas que trae aparejadas son muchas. La primera ventaja que aparece y una de la más importante es la movilidad que adquiere el usuario de estas redes. Una computadora o cualquier dispositivo puede acomodarse en cualquier punto dentro del área de cobertura de la red sin tener que preocuparse si es posible o no hacer llegar un cable de red hasta este lugar. La portabilidad es otro punto importante, ya que permite a los usuarios moverse junto con los dispositivos conectados a la red inalámbrica tales como notebooks, netbooks o similares sin perder el acceso a la red. La flexibilidad es otra ventaja de las redes sin cables. Podemos situar nuestra notebook sobre la mesa del escritorio para luego desplazarla hacia el dormitorio sin tener que realizar el más mínimo cambio de configuración de la red. Otras de las ventajas son el ahorro de costos y la facilidad de expandir la red luego de ser instalada.

En esta obra, entre otras cosas, aprenderemos cómo añadir conectividad WiFi a nuestros desarrollos, estudiando un pequeño módulo de la empresa ZeroG. Además, veremos cuáles son las precauciones y las consideraciones más importantes al momento de diseñar una placa donde vaya insertado este dispositivo. Es nuestro deseo que disfruten el libro y se convierta en una ayuda fundamental para explorar el mundo de la conectividad con microcontroladores.

El libro **de un vistazo**

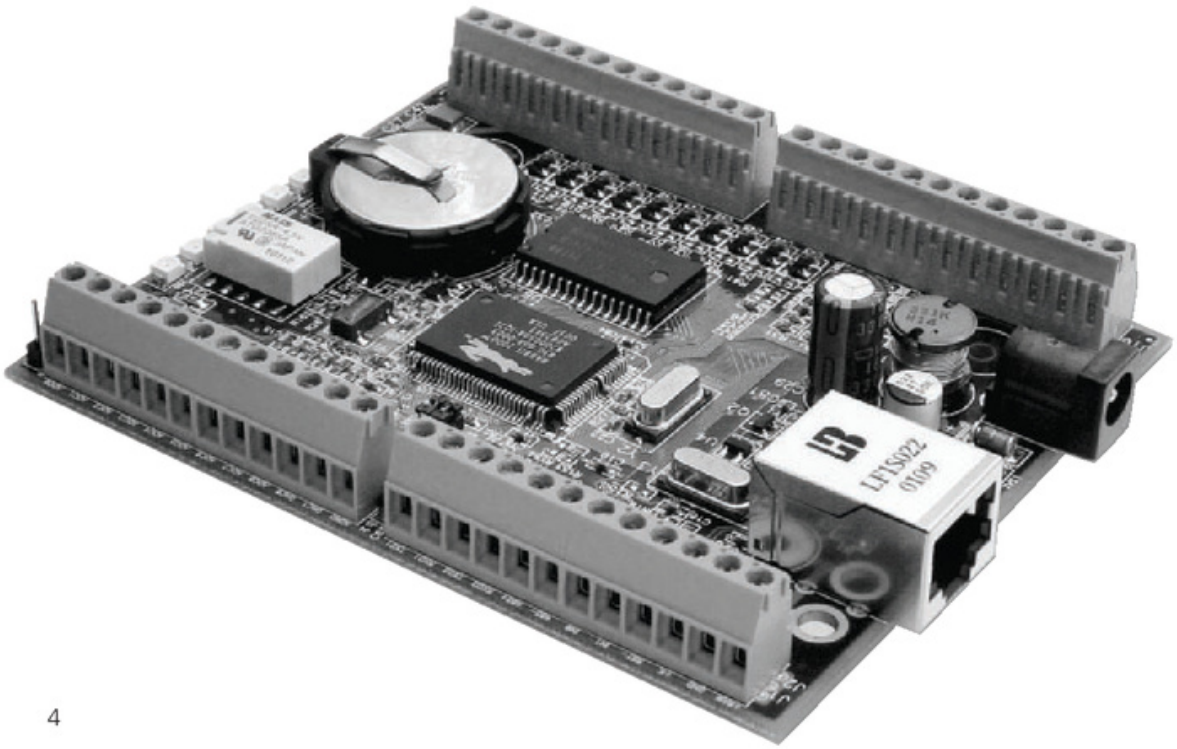
El objetivo de este libro es hacer un vasto recorrido por la conectividad entre los microcontroladores. Destinado a quienes quieran profundizar sus conocimientos sobre electrónica digital y realizar complejos proyectos, los cuales serán explicados paso a paso y complementados mediante ilustrativas infografías.

CAPÍTULO 1
NETWORKING CON TPC/IP

En este primer capítulo, analizaremos las diferentes redes, según su alcance y topología. Además, veremos de qué se trata el modelo OSI y explicaremos las distintas capas que posee. Conoceremos la forma en que nuestro navegador web se comunica con los servidores web a través del protocolo HTTP y el lenguaje HTML. Para finalizar el capítulo, veremos cómo conectarnos a un servidor web mediante Telnet, para obtener el código HTML que interpreta el navegador.

CAPÍTULO 2
NETWORKING CON RABBIT

Conoceremos la interfaz de programación que el fabricante brinda para desarrollar nuestras aplicaciones. Analizaremos RabbitWeb, un poderoso lenguaje script que permite la generación de páginas web dinámicas. Estudiaremos el sistema de conectividad WiFi y veremos las pequeñas diferencias para adaptar código de Ethernet a él. Veremos la forma de escribir código para varias tareas simultáneas y la alternativa que ofrece Rabbit.



CAPÍTULO 3
NETWORKING CON PIC

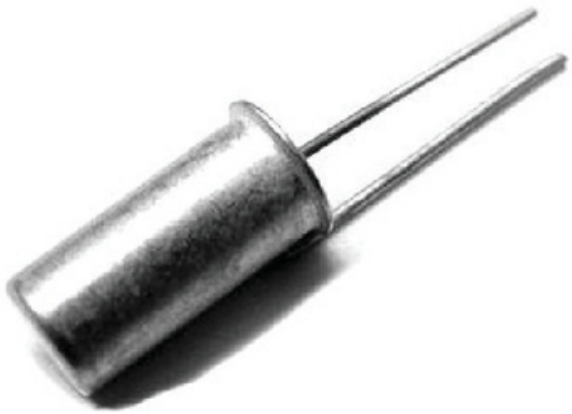
En este capítulo, veremos cómo configurar el servidor web para obtener conectividad a través de un router o una PC y controlar variables en forma remota. Además, conoceremos las opciones para presentar la información al usuario final de nuestra aplicación. Analizaremos la actualización automática y los métodos de almacenamiento.

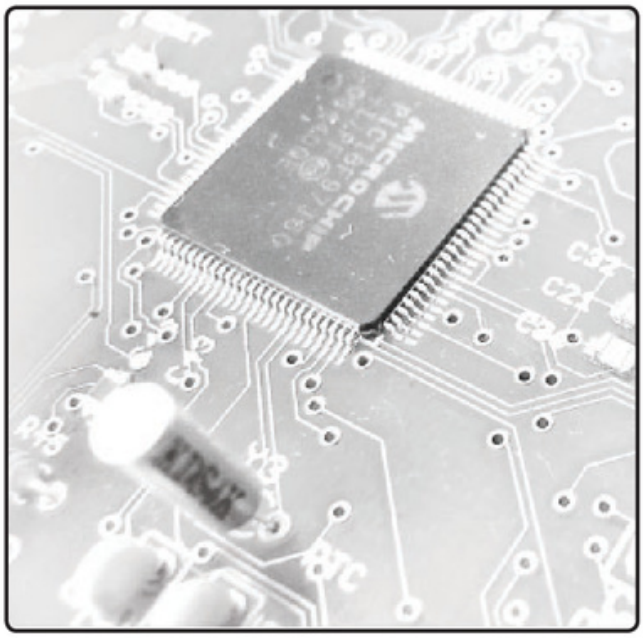
CAPÍTULO 4
TELEMETRÍA Y CONTROL REMOTO ETHERNET CON PIC

Aprenderemos las principales características del PIC18F97J60. También, conoceremos la teoría del monitoreo de magnitudes analógicas a distancia y realizaremos aplicaciones sobre Ethernet. Para completar lo aprendido, veremos algunas alternativas comerciales para desarrollar una comunicación a través de Ethernet con PIC.

CAPÍTULO 5
WIFI CON PIC Y ZEROG

Estudiaremos las principales características del módulo WiFi con ZeroG y su forma de conexión para comunicarse con los microcontroladores.





Además, conoceremos cuáles son las precauciones que debemos tomar al momento de agregar este módulo en nuestros desarrollos y daremos algunos consejos importantes a la hora de conectar un módulo con un router WiFi. Cerraremos el capítulo con un ejemplo de utilidad, donde accionaremos un actuador desde la página web del servidor.

APÉNDICE
ROBÓTICA

En este apéndice incursionaremos en el mundo de la robótica, aprendiendo las nociones básicas de esta excelente disciplina. Veremos la definición de lo que es un robot y las distintas clases que existen.

SERVICIOS
AL LECTOR

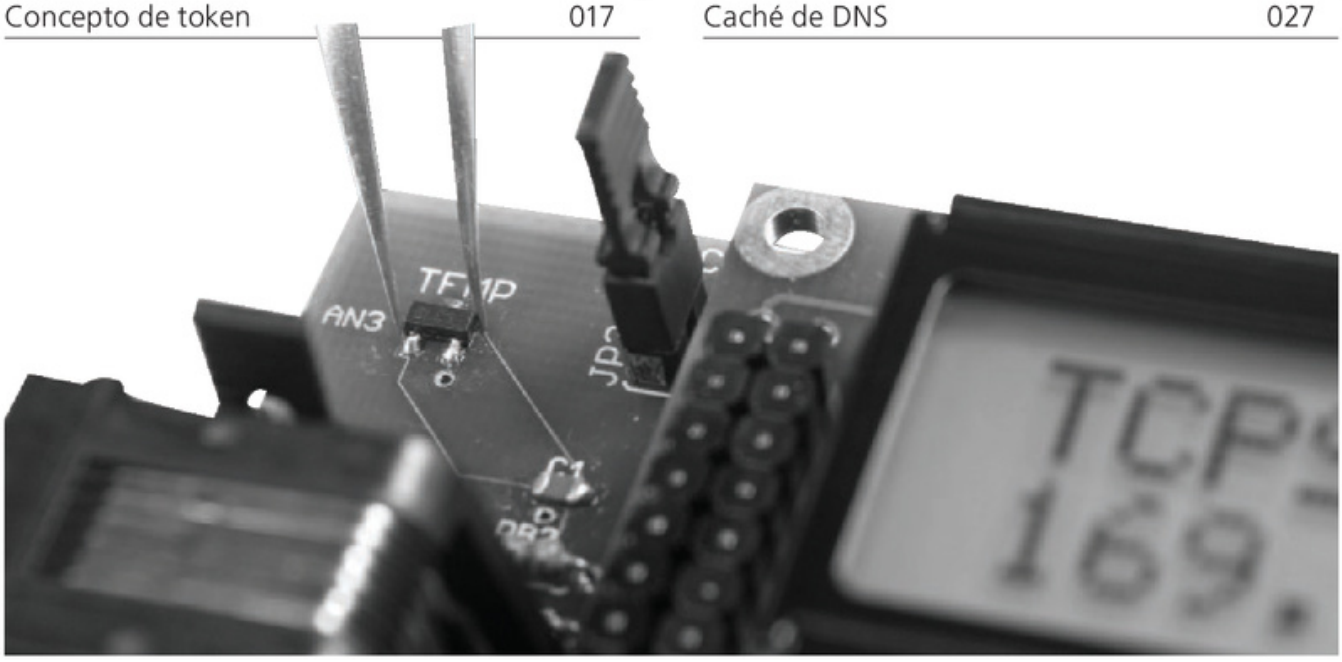
En este último apartado, presentamos un índice temático que nos ayudará a encontrar de forma más rápida y precisa los principales conceptos de la obra.

Contenido del libro

Prólogo al contenido	**003**
El libro de un vistazo	**004**
Introducción a Networking con microcontroladores	**010**

CAPÍTULO 1 NETWORKING CON TPC/IP 011

Clasificación de las redes	**012**
Redes LAN, MAN y WAN	012
Topología BUS	013
Topología Malla	013
Topología Estrella	014
Topología Anillo	014
Dispositivos de red	015
Concepto de token	017
Modelo OSI	**018**
Capa1: Física	018
Capa2: Enlace de datos	018
Capa3: Red	019
Capa4: Transporte	019
Capa5: Sesión	019
Capa6: Presentación	019
Capa7: Aplicación	020
Flujo de información entre capas	021
TCP/IP	**022**
Estructura TCP/IP	022
• Capa1: Acceso a red	022
• Capa2: Internet	023
• Capa3: Transporte	023
• Capa4: Aplicación	023
Protocolo IP	024
• Dirección IP	024
Resolución de nombres: DNS	**025**
Sistema de nombre de dominio	025
Servidores DNS	026
Caché de DNS	027



UDP y TCP	**028**
Protocolo UDP	028
• Datagrama UDP	028
Protocolo TCP	028
• Fiabilidad en TCP	029
• Conexión y desconexión en TCP	030
Conexión a servidores web	**031**
El protocolo HTTP	031
HTML	031
Cabecera HTTP	031
Web server vía Telnet	031
El lenguaje HTML	**036**
Ethernet con PIC	**036**
TCP/IP en Rabbit	038
La interfaz de programación	038
Los servicios	039
Otras alternativas	**039**
• Freescale	040
• Texas Instruments	040
• Atmel AVR	040
• WIZnet	040
• Los chips	040
• Los módulos	040
Multiple choice	**042**

CAPÍTULO 2
NETWORKING CON RABBIT 043

Las funciones de networking en Rabbit	**044**
Inicialización de la interfaz	044
Manejo del stack	044
Sockets	045



Resolución por DNS	046
Carga de archivos en memoria	046
Servidor HTTP	047
Servidores web SSI y CGI	**049**
En Rabbit	050
• Autenticación	050
• Entorno Rabbit	051
RabbitWeb	**052**
Funciones	052
Las extenciones a C	053
Utilización	054
Soportar AJAX	055
Servidor web con Rabbit	055
WiFi con Rabbit	**061**
Conexión y encriptación	062
Inicio del servidor web	063
Multitarea cooperativo	**064**
Multitarea	064
• Multitarea cooperativo	064
• Máquinas de estados finitos	064
Reutilización de código	067
Multitarea cooperativo con Rabbit	**068**
Costates (co-sentencias)	068
Cofunctions (co-funciones)	068
Sentencias de control	069

Reutilización de código I 069
Multiple choice 070

CAPÍTULO 3
NETWORKING CON PIC 071

Introducción a networking 072
Microchip TCP/IP Configuration Wizard 072
Agregar servicios al stack 074
Funciones especiales del stack 074
Configuración del Web Server 077
Ubicación del la página web 077
Compilar el stack 078
Acceder a la página web 079
Los archivos principales del stack 082
Funciones callback 084
Programación de la página web 085
Sistema de archivos MPFS 086
Cargar la web en la memoria interna del PIC 090
Microchip WebPreview 091
Envío de mails 092
Ejemplo completo 094
Puente Service-Ethernet 095
Acceso desde Internet 096
Multiple choice 100



CAPÍTULO 4
TELEMETRÍA Y CONTROL REMOTO ETHERNET CON PIC 101

Arquitectura del PIC18F97J60 102
Alimentación 102
Puertos I/O 102
Modelo Ethernet 104
Telemetría 108
Telemetría de valores analógicos 108
Variables dinámicas 109
- Cómo insertar una variable dinámica 110
- Cómo utilizar una variable dinámica 110
- Parámetros de las variables dinámicas 112

Monitoreo de la temperatura por Ethernet 113
Servidores web dedicados 114
Funciones de un servidor web dedicado para control 114
Alternativas comerciales 118
Control remoto 119
- Ejemplo de control remoto 120

Método GET para controlar variables 121
Formularios web 122
Manejo de los datos con el método GET 123
Encender un LED y un relay

por Ethernet **123**
Encendido de un LED por Ethernet 124
Función callback para encender el LED 124
Manejo de varios LEDs y relays 124
Encender una lámpara de 220V a través de Ethernet 129
Multiple choice **134**

CAPÍTULO 5 WIFI CON PIC Y ZEROG 135

Módulo ZeroG **136**
Módulo ZG2100M 137
- Características del módulo ZG2100M 138
- Alimentación del módulo 138
- Comunicación con el módulo ZG2100M 139
- Conexión del módulo ZeroG por SPI 140
- Interfaz SPI esclavo con interrupciones al host 143
- Control del módulo por el puerto SPI 144

Acceso al espacio de registros 144
Interfaz FIFO 145
Consideraciones de layout 145
Ejemplo de diseño del PCB 147
Microchip Configuration Wizard para WiFi 150
Asignación de la dirección MAC pregrabada 152
Prioridad de las direcciones 153
Conexión a una red WiFi existente 154
Canal de radio frecuencia 154
Seguridad y encriptación de datos 155
- Seguridad WEP 155
- Seguridad WEP/WPA2 156

Configuración del router **156**
Configuraciones básicas 157
- Configuración de la red inalámbrica 158
- Configuración de la seguridad 158

Módulos de entrenamiento 159
Placa MCE WebServer WiFi 159
Control remoto de un motor a través de WiFi 160
Multiple choice **170**

APÉNDICE ROBÓTICA 171

Anatomía de los robots **172**
Tipos y características de los robots 174
Aplicaciones de los robots **175**
Generaciones de robots 175
Inteligencia de un robot 176
Componentes de un robot **177**
El controlador 178
Otras clasificaciones 179

SERVICIOS AL LECTOR 180

Índice temático **181**
Catálogo **184**

Introducción a Networking con microcontroladores

Hoy en día, existen redes informáticas en todas partes: en el hogar y en la oficina, sin olvidarnos de Internet, la red de redes. Estas son un conjunto de dispositivos interconectados, que permiten el intercambio de información entre ellos. Se pueden clasificar por su alcance y por la disposición física de interconexión de los dispositivos que las integran.

Para comenzar, aprenderemos sobre el modelo de referencia OSI, explicando sus capas. Posteriormente, analizaremos las diferencias con respecto al modelo TCP/IP. Veremos también qué son las direcciones IP, el concepto de DNS y de qué manera las direcciones de dominio se traducen en direcciones IP.

Estudiaremos los conceptos fundamentales del protocolo IP, como así también de UDP y TCP, con sus diferencias. Además, mostraremos en forma práctica cómo realizar una simple conexión con un servidor web mediante Telnet.

Además, analizaremos un esquema compacto de Rabbit con soporte para multitarea cooperativo. Conoceremos la interfaz de programación que el fabricante provee para desarrollar aplicaciones. Estudiaremos WiFi y veremos las pequeñas diferencias para adaptar código de Ethernet a él.

Realizaremos un servidor web basado en el microcontrolador PIC PIC18F97J60. Para complementar lo aprendido, veremos cómo cargar las páginas web en la memoria flash del PIC.

Estudiaremos la arquitectura del PIC18F97J60 y los detalles del hardware para su implementación. Con el fin de completar nuestras competencias, aprenderemos a conectar el microcontrolador PIC a una red inalámbrica WiFi por medio de los módulos de comunicación ZeroG. Para finalizar, incluimos un interesante apéndice sobre robótica, donde analizaremos las diferentes clases que existen.

Capítulo 1
Networking con TCP/IP



Analizaremos el funcionamiento de las redes, veremos cómo se clasifican y sus topologías.

Clasificación de las redes

Antes de abordar este tema, surge una pregunta clave: ¿qué son las redes informáticas? Pues bien, se trata de un conjunto de dispositivos interconectados, que permiten el intercambio de información entre ellos. Esta es una definición bastante amplia, pero es la base fundamental del tema que trataremos en este apartado.

Las redes pueden clasificarse de distintas maneras. Una de ellas es por el alcance: es así que tenemos redes tipo **LAN**, **MAN** y **WAN**. También podemos agruparlas según su topología, es decir, la disposición física de interconexión de los distintos dispositivos que las integran: **bus**, **estrella**, **árbol**, **malla**, **anillo**, **anillo doble** y **mixta**.

Otra posibilidad es hacerlo de acuerdo con la forma de transmisión de datos, en cuyo caso pueden ser **simplex** (de transmisión simple), que es unidireccional (un dispositivo transmite y otro recibe); **half duplex** (bidireccionales), donde ambos dispositivos pueden transmitir y recibir, pero no al mismo tiempo, y **full duplex** (bidireccionales), en las que los dos dispositivos transmiten y reciben simultáneamente.

REDES LAN, MAN Y WAN

Las redes tipo **LAN** (*Local Area Network* o red de área local) son de pequeña extensión (entre 10 metros y 1 kilómetro). Se las emplea generalmente en el ámbito de un edificio u organización (**Figura 1**). Las de tipo **MAN** (*Metropolitan Area Network* o red de área metropolitana) tienen una extensión mayor (hasta 10 kilómetros), en tanto que las **WAN** (*Wide Area Network* o red de área amplia) se extienden aún más que las anteriores (entre 100 y 1000 kilómetros).



TOPOLOGÍA ÁRBOL

Puede considerarse como varias topologías en estrella, en donde el dispositivo central de cada una se conecta con un dispositivo central de nivel superior, siguiendo la forma de un árbol de conexiones. La ventaja principal de esta modalidad es que resulta sencillo ampliar la red.

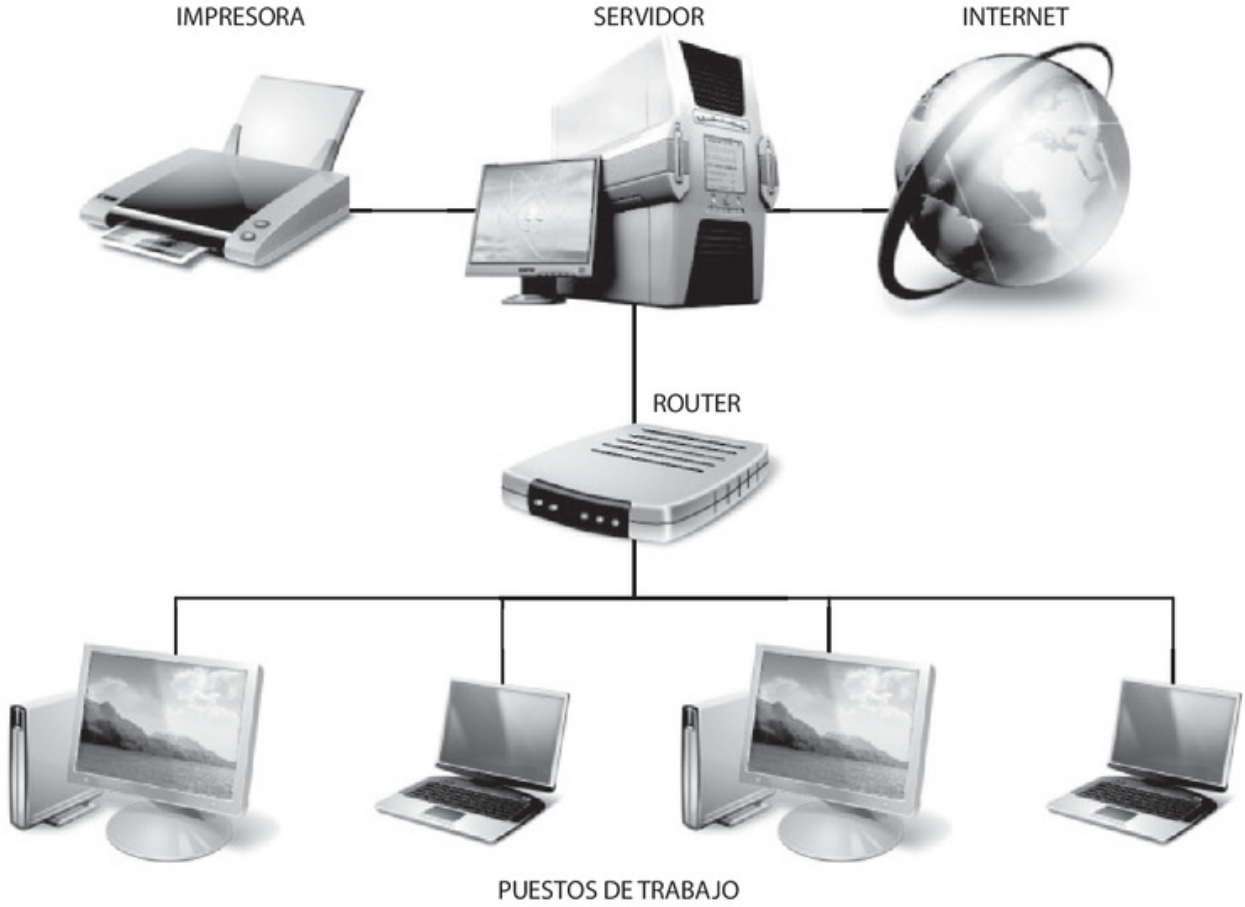


FIGURA 1. **Conexión de red LAN montada en una oficina. Hay un servidor principal, una base de datos, una impresora y cuatro puestos de trabajo.**

TOPOLOGÍA BUS

Consta de un bus principal, al cual se conectan todos los dispositivos de la red. El medio de transmisión típico es el cable coaxial. Para la construcción de esta red, se requiere que en ambos extremos del cable principal se coloque un terminador, que no es más que una resistencia cuyo fin es adaptar las impedancias.

La principal ventaja de esta topología es su facilidad de instalación y posterior ampliación. Como desventaja, podemos mencionar la siguiente situación: ante un problema físico en la red, como algún cable cortado, falla la red completa.

TOPOLOGÍA MALLA

En este caso, cada dispositivo de la red se conecta directamente con los demás (**Figura 2**). En una malla completa, todos los dispositivos están conectados con todos. Una de las ventajas es que los paquetes de datos pueden viajar por diferentes caminos. La desventaja es el alto costo de implementación, debido a la gran cantidad de conexiones que se requiere.

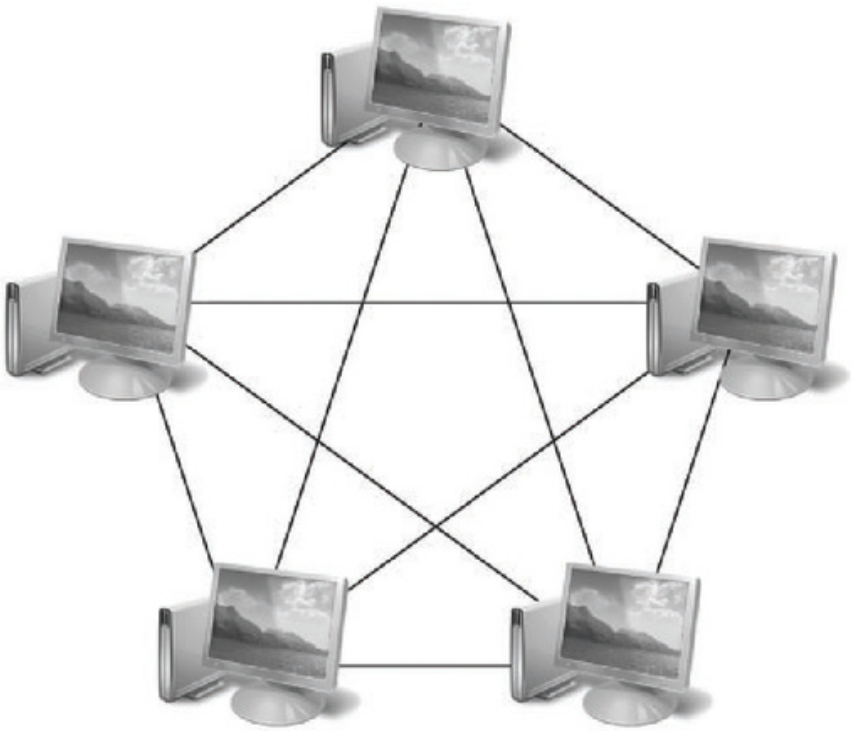


FIGURA 2. **En una red con topología** malla, **todos los dispositivos se conectan entre sí.**

TOPOLOGÍA ESTRELLA

En esta modalidad, todos los dispositivos de la red se conectan a un único dispositivo central, que puede ser un hub (concentrador), un switch (conmutador), un router (enrutador) o, directamente, otra computadora (Figura 3). De esta forma, todas las comunicaciones pasan a través de dicho dispositivo central.

Uno de los puntos a favor de esta topología es que, ante un problema en alguna de las estaciones de trabajo o en el medio de transmisión entre esta y el dispositivo central (algún cable cortado, por ejemplo), el resto de la red continúa funcionando normalmente. El punto negativo es que una falla en el dispositivo central afecta a toda la red.

Las redes son un conjunto de dispositivos interconectados para el intercambio de información

TOPOLOGÍA ANILLO

En una topología anillo, los dispositivos de la red se conectan uno a uno formando un anillo; es decir, cada uno se conecta con el siguiente, y el último, con el primero (Figura 4). En este caso, los paquetes de datos viajan de un dispositivo a otro en una única dirección a través del medio de transmisión.

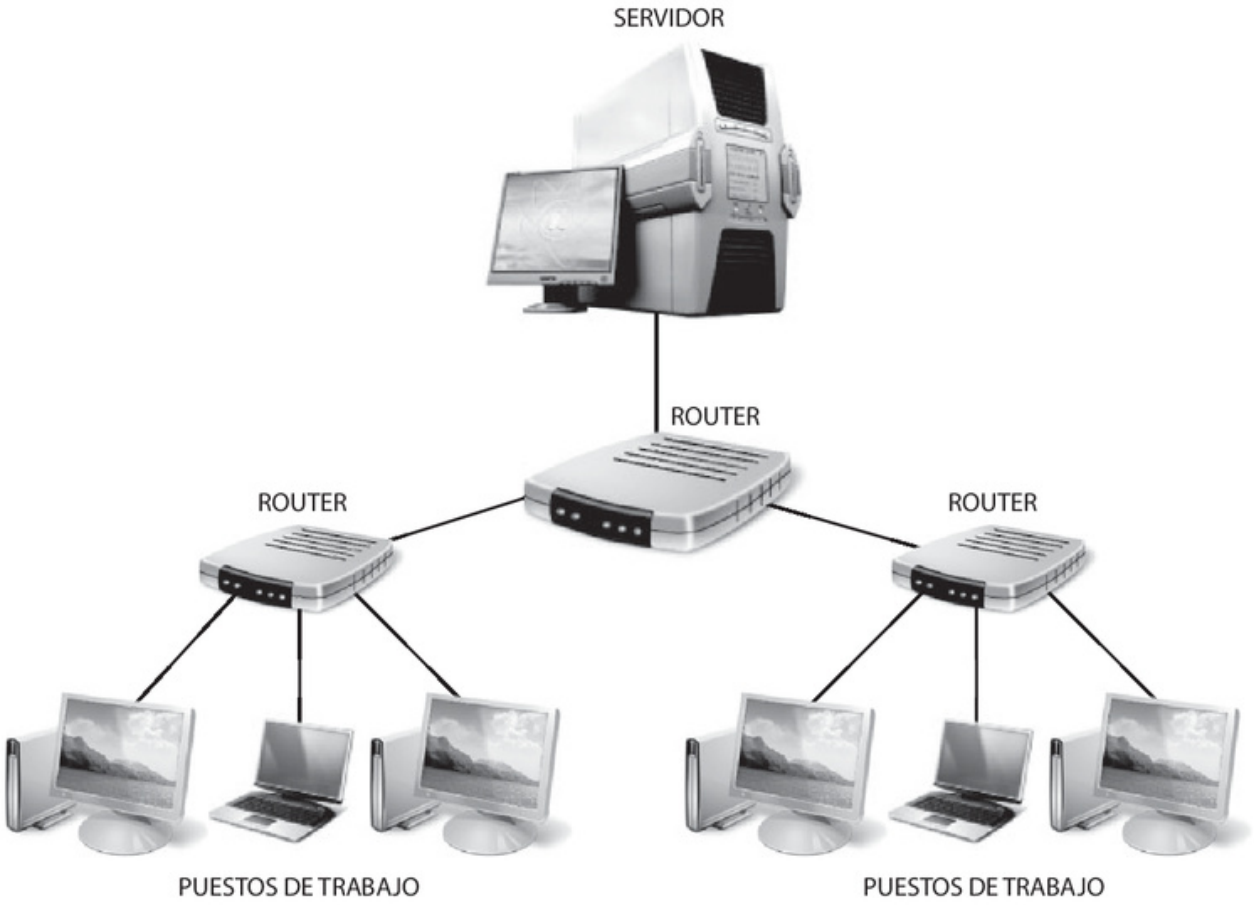
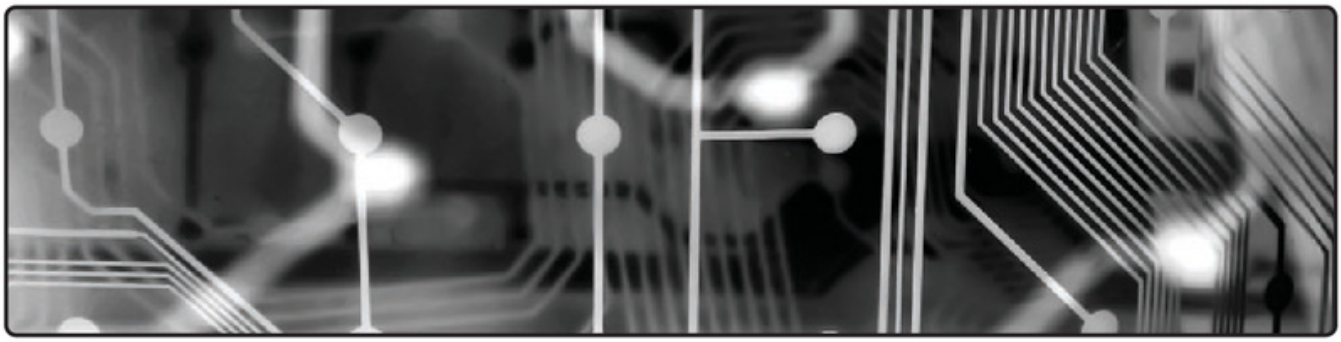


FIGURA 3. **En una red con topología** árbol, **podemos observar que el dispositivo central de cada nivel se comunica con el del nivel superior.**

Cada dispositivo verifica a quién va dirigido ese paquete. En caso de que no sea dirigido a él, lo retransmite al siguiente hasta llegar a destino (**Figura 5**).

DISPOSITIVOS DE RED

Entre los diferentes dispositivos de red existentes, podemos nombrar el hub o concentrador, el switch o



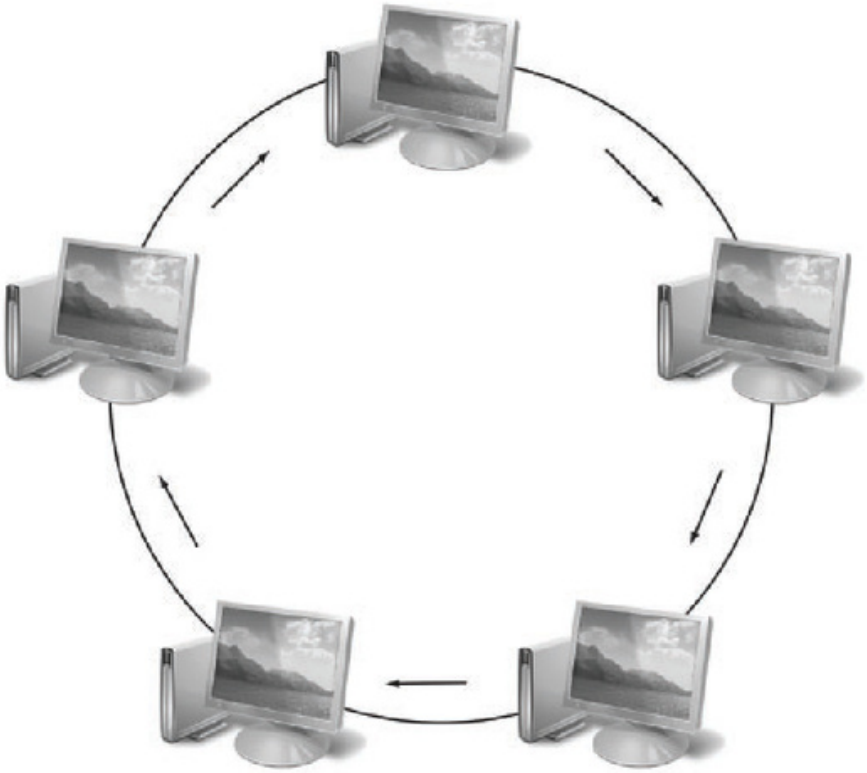


FIGURA 4. **Podemos observar la distribución de una topología** anillo, **en donde los dispositivos se conectan uno a uno para componer una estructura de esa forma.**

conmutador y el router o enrutador. Un **hub** es un dispositivo que trabaja en la capa física (capa 1 del modelo OSI, que veremos más adelante) y permite estructurar el cableado de una red para expandirla.

El **switch** trabaja en la capa de enlace de datos (capa 2 del modelo OSI). Se dice que es como un "hub inteligente", ya que elige el mejor camino para enviar los datos. Por último, el **router** actúa en la capa de red (capa 3 del modelo OSI) y una de sus funciones es brindar conectividad con Internet a los distintos dispositivos que la integran. Al trabajar en esa capa, maneja direcciones IP, las cuales detallaremos más adelante, en este mismo capítulo.

Una típica configuración de red hogareña con conexión a Internet de banda ancha se realiza conectando la salida del módem provisto por el proveedor de Internet a la entrada de un router. Las salidas del router se conectarán a las interfaces de red de cada PC o dispositivo de la red.

Una topología mixta permite combinar las ventajas particulares de otros tipos de topologías

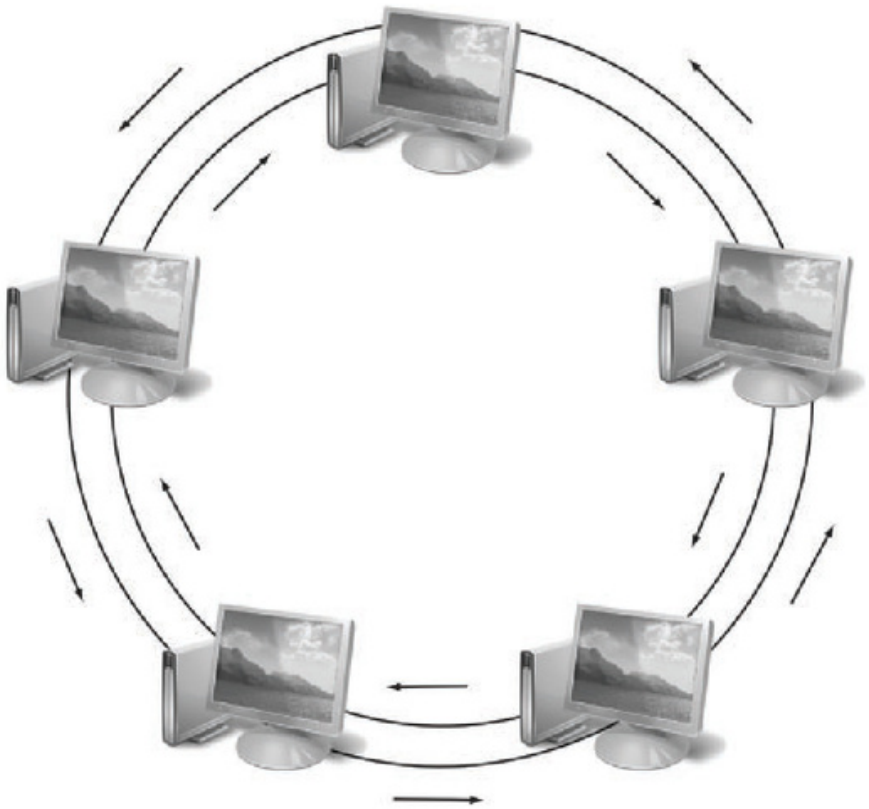
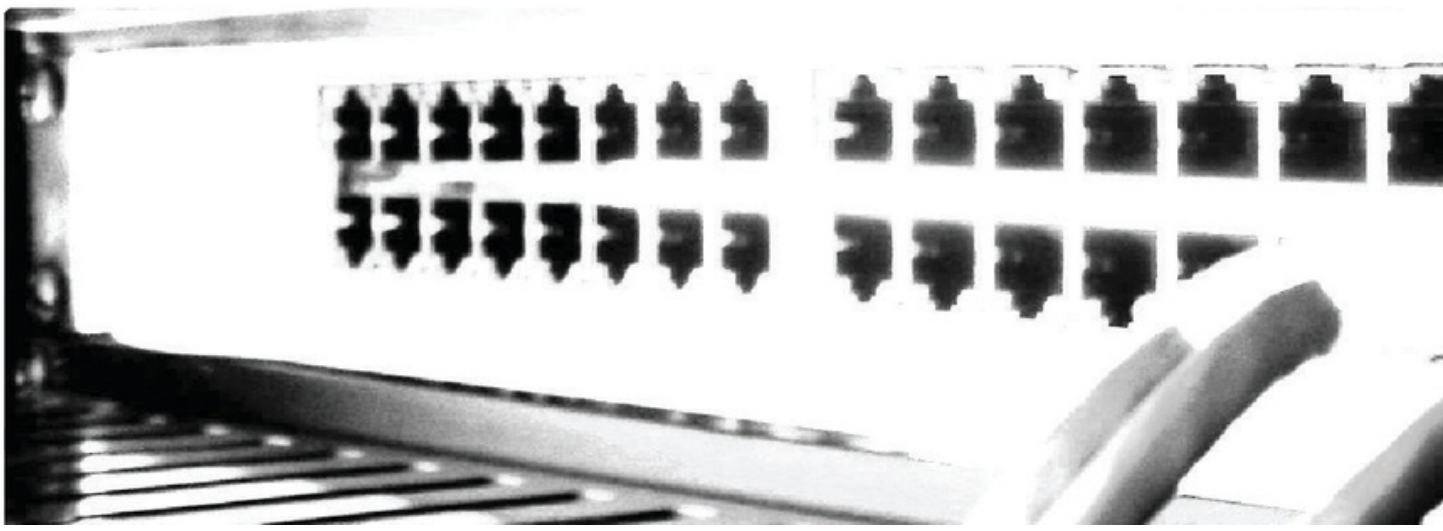


FIGURA 5. **Observamos una topología** anillo doble, **con la cual logramos redundancia en la transmisión de los datos y, por lo tanto, reducción de fallas.**

CONCEPTO DE TOKEN

Para evitar que varios dispositivos envíen datos a la red al mismo tiempo, surge el concepto de **token**, que no es más que una serie de bits que viajan por la red. El token es capturado por cada dispositivo, y le permite acceder al medio en ese momento. Cuando el dispositivo recibe el token, puede enviar datos por la red; mientras que cuando el dispositivo de destino recibe el dato, le envía una señal al de origen para indicar que la operación se realizó correctamente.



Modelo OSI

El modelo de referencia **OSI** (*Open System Interconnection* o interconexión de sistemas abiertos) fue creado en el año 1984 por la ISO (*International Organization for Standardization* u Organización Internacional para la Estandarización). El objetivo era solucionar los problemas de compatibilidad entre los sistemas provenientes de distintos fabricantes. Este modelo define una estructura de siete capas, en donde cada una es completamente independiente del resto y soluciona un aspecto particular del sistema de comunicación (**Figura 6**). Veamos cada una de ellas.

CAPA 1: FÍSICA

La capa física es la que se encarga de transmitir los bits (unos y ceros), ya sea por un cable (medios guiados) o en forma inalámbrica (medios no guiados). En esta capa se definen, entre otras cosas, los niveles de tensión en los cuales se representan los unos y los ceros, los conectores empleados, el tipo de medio guiado (por ejemplo, coaxial, par trenzado o fibra óptica), el tipo de medio no guiado (como infrarrojo, microondas o radiofrecuencia), la velocidad de transmisión, y otros parámetros. En definitiva, todos los aspectos físicos requeridos para transportar los bits de información.

CAPA 2: ENLACE DE DATOS

La función principal de esta capa es permitir la comunicación con el dispositivo inmediato, y que esta sea libre de errores. Esta capa delimita la cantidad de bits que maneja la capa física. Por lo tanto, envía los bits en forma de tramas, agregando una codificación al inicio (encabezado o *header*) y otra al final (cola o *tail*). De esta manera, se permite la detección y recuperación de errores. Ethernet ocupa tanto esta capa como la 1.

El modelo OSI solucionó los problemas de compatibilidad entre sistemas de distintos fabricantes



FIGURA 6.
Diagrama de las capas del modelo de referencia OSI. **La inferior es la física, y la superior, la de aplicación, la que está más cerca del usuario.**



CAPA 3: RED

La función de la capa de red es enrutar y enviar los paquetes de información entre las diferentes redes. Define el camino por donde viajarán los datos para llegar a destino. De esta forma, se encarga de controlar la congestión en la red. En esta capa se habla de los **protocolos IP** *(Internet Protocol o protocolo de Internet)*, **NetBEUI** *(NetBIOS Extended User Interface)* o interfaz extendida de usuario de **NetBIOS** *(Network Basic Input/Output System)*.

CAPA 4: TRANSPORTE

Es la capa **intermedia** del modelo, que actúa como nexo entre los primeros tres niveles (encargados, básicamente, de la comunicación) y los tres últimos (más orientados a la aplicación). En ella se define, por ejemplo, si los paquetes de información se entregarán en el mismo orden en el que fueron enviados o en cualquier orden. Esta capa agrega un **header**, que gestiona las distintas maneras de operación; también garantiza la integridad de la información y el control de flujo. En esta capa se habla de **TCP** (*Transmission Control Protocol* o protocolo de control de transmisión), **UDP** (*User Datagram Protocol* o protocolo de datagrama de usuario) y otros.

CAPA 5: SESIÓN

La capa de sesión es la encargada de comenzar, mantener y terminar la comunicación. Brinda servicios a la capa de transporte, como el agregado de puntos de sincronismo para la recuperación de transferencias extensas, en donde la comunicación es interrumpida por algún motivo.

CAPA 6: PRESENTACIÓN

El objetivo de esta capa es lograr que la información sea representada correctamente.

También, se ocupa de tomar la información proveniente de la capa de aplicación y darle un formato entendible para las demás capas. Permite, además, la encriptación y la compresión de datos.

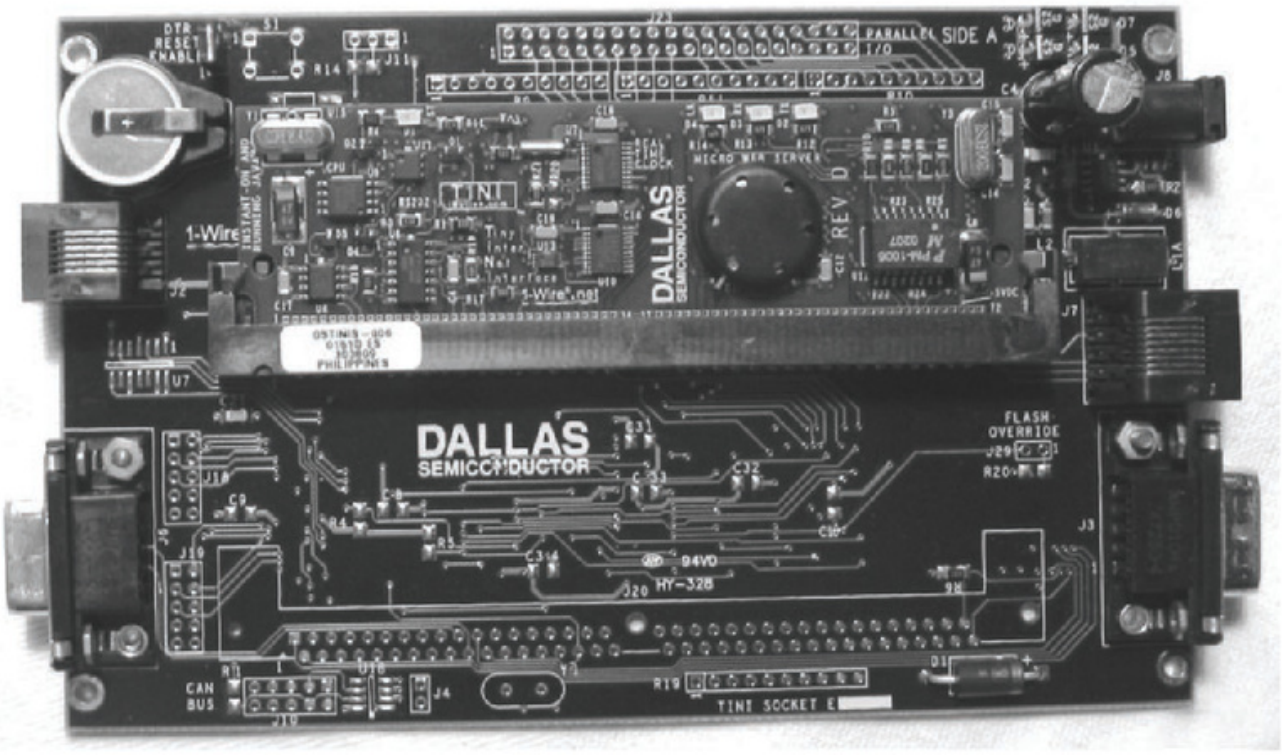
CAPA 7: APLICACIÓN

Es la **última** capa del modelo de referencia OSI, con lo cual, es la que se encuentra **más cerca del usuario**. Se ocupa de brindarles servicios a las aplicaciones de los usuarios. Cuando hablamos de aplicaciones, lo primero que se nos ocurre es un navegador web, un cliente de correo electrónico, etc.

Sin embargo, en el marco del modelo de referencia OSI, abarca a los **protocolos de red** que emplean las aplicaciones con las que interactúa una persona. Por ejemplo, cuando navegamos por

Cada capa del modelo OSI es independiente y soluciona un aspecto particular del sistema de comunicación

Internet, no leemos directamente el **código HTML** (*HyperText Markup Language* o lenguaje de marcado de hipertexto), sino que lo hacemos mediante el uso de un navegador web, que interpreta el código y lo traduce en un formato más amigable para el usuario. En esta capa se habla de **HTTP** (*Hypertext Transfer Protocol* o protocolo de transferencia de hipertexto), **SMTP** (*Simple Mail Transfer Protocol* o



protocolo simple de transferencia de correo) y **FTP** (*File Transfer Protocol* o protocolo de transferencia de archivos) entre otros.

FLUJO DE INFORMACIÓN ENTRE CAPAS

Imaginemos que queremos enviar un mensaje desde un **host 1** a un **host 2**, ambos ubicados en la misma red local. El mensaje se interpreta como un dato, que comienza su viaje por encima de la capa de aplicación del host 1. Este dato irá descendiendo por cada capa, y cada una de ellas le agregará una codificación al inicio (encabezado o header), antes de pasarlo a la capa inferior.

La capa de enlace de datos agrega, además, una codificación al final (cola o tail). La física toma esta secuencia de bits y la manda a través de la red, hasta que llega a la capa física del host 2. Una vez allí, la información va subiendo por cada capa, cada una de las cuales se encarga de quitarle el encabezado correspondiente, hasta que, finalmente, pasa a la capa de aplicación del host 2 (**Figura 7**).

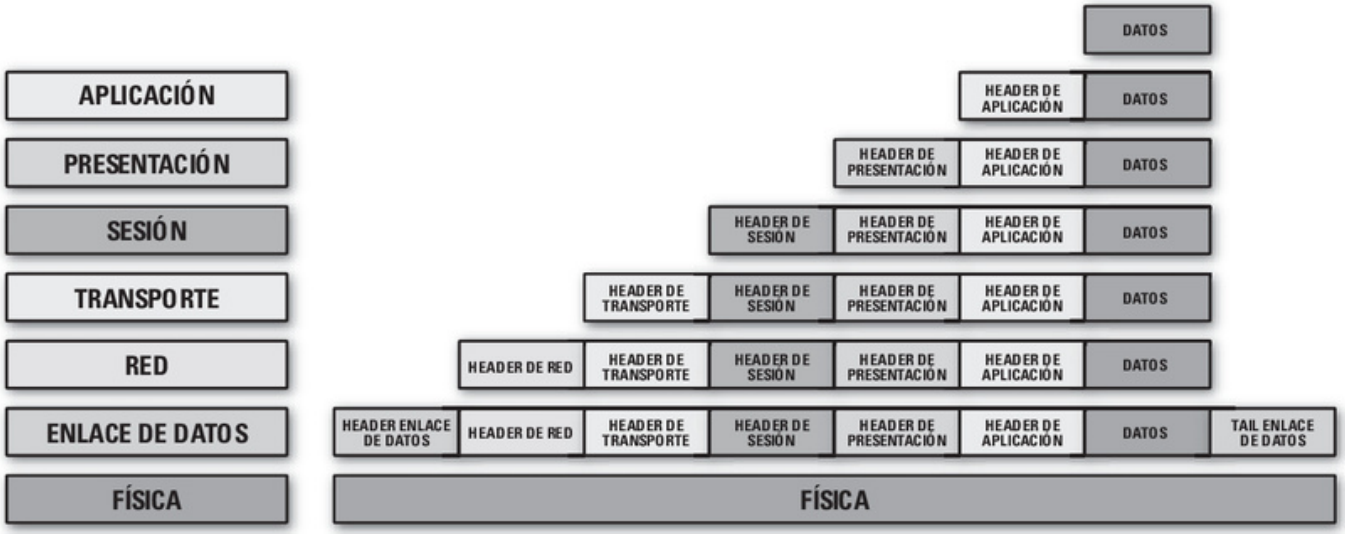
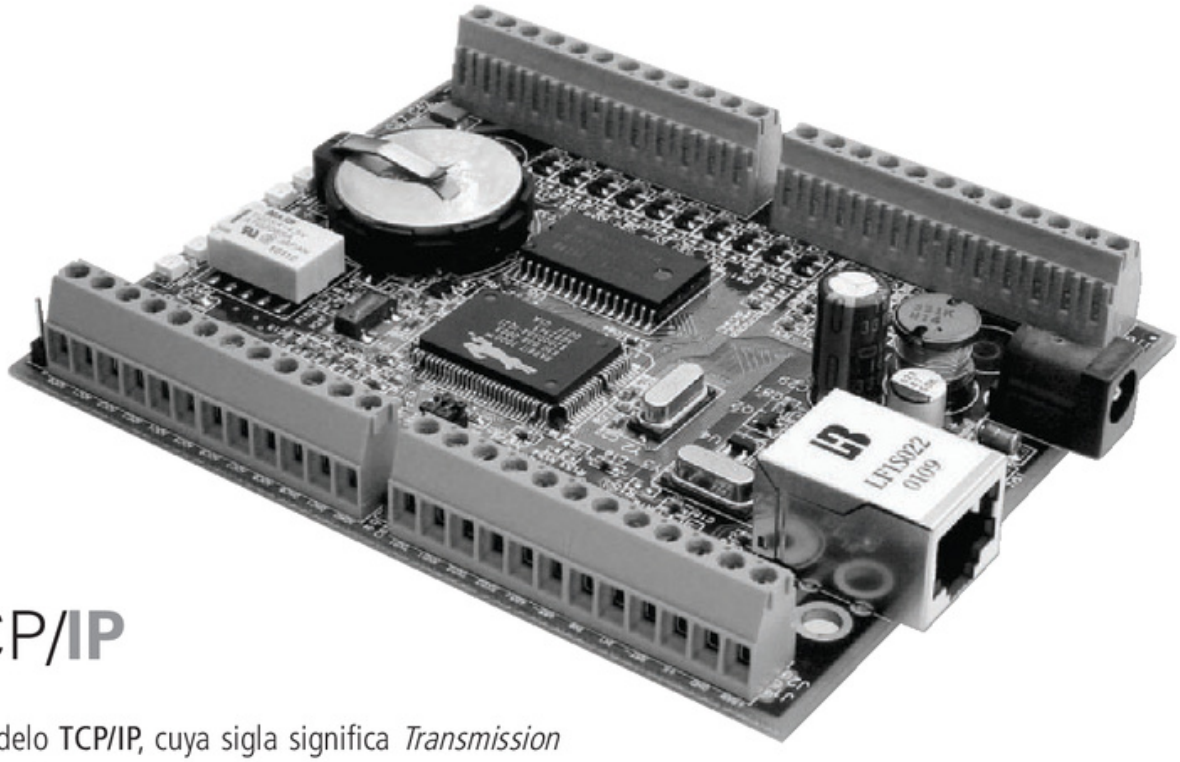


FIGURA 7. . **Flujo de información entre capas. Vemos cómo se va formando el mensaje en cada una, al agregarle el header/tail (encabezado/cola) correspondiente.**

IP ESPECIAL

Existen direcciones con significados especiales. Por ejemplo, la 0.0.0.0 (este host en esta red) es la que posee cada dispositivo cuando aún no se le ha asignado una IP. La dirección **loopback** es la 127.0.0.1, y es la de nuestro propio equipo.



TCP/IP

El modelo **TCP/IP**, cuya sigla significa *Transmission Control Protocol* (protocolo de control de transmisión) e *Internet Protocol* (protocolo de Internet), nació en los Estados Unidos a mediados de los años 70. Fue desarrollado por **DARPA** (*Defense Advanced Research Projects Agency* o Agencia de Proyectos de Investigación Avanzada), bajo el proyecto **ARPANET** (**Figura 8**).

ESTRUCTURA TCP/IP

TCP/IP es un conjunto de protocolos que permiten realizar el transporte confiable de datos por las redes. Al igual que el modelo OSI, el TCP/IP está dividido en capas, en este caso, cuatro: acceso a red, Internet, transporte y aplicación (**Figura 9**). Veamos en detalle cada una de ellas.

Capa 1: Acceso a red

Es la capa inferior del modelo TCP/IP. Si hacemos una analogía entre el modelo OSI y el TCP/IP, corresponde al conjunto de capas 1 y 2 del primero (capa física y de enlace de datos).

IP RESERVADA

Existen algunas direcciones reservadas para redes privadas, que no existen en Internet; ellas son: 10.0.0.0 a 10.255.255.255, 172.16.0.0 a 172.31.255.255 y 192.168.0.0 a 192.168.255.255. Estos números nunca deben ser utilizados en Internet.

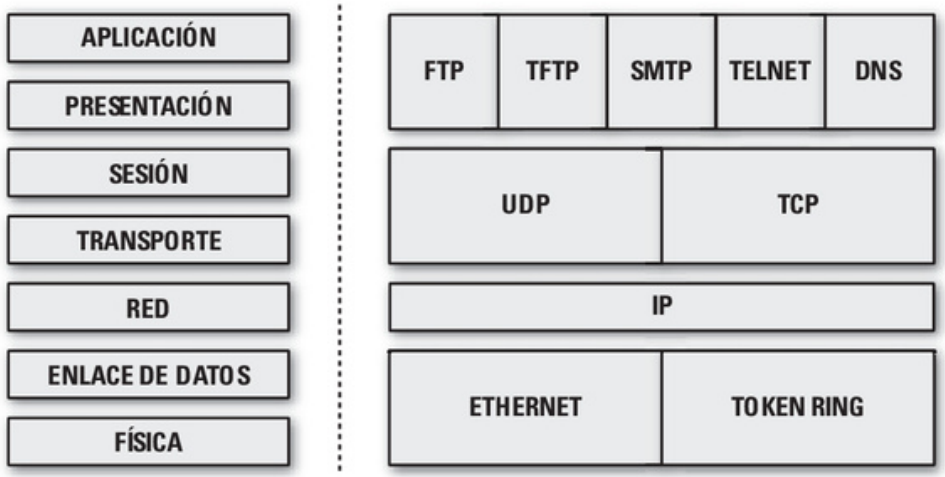


FIGURA 8. **Podemos observar algunos de los protocolos más importantes de** TCP/IP **y la capa del** modelo de referencia OSI **en que se encuentran.**

En esta capa se utilizan conceptos como Ethernet y token ring, entre otros.

Capa 2: Internet

Se encarga de enrutar los paquetes para enviarlos a otras redes. Es análoga a la capa 3 del modelo OSI (capa de red). En esta capa se habla de protocolo IP, ARP (*Address Resolution Protocol* o protocolo de resolución de direcciones) e ICMP (*Internet Control Message Protocol* o protocolo de mensajes de control de Internet).

Capa 3: Transporte

Corresponde a la capa 4 del modelo OSI (capa de transporte). Transporta los mensajes sin importar el tipo de red empleada. En esta capa se habla de TCP y UDP, dos protocolos que veremos más adelante.

Capa 4: Aplicación

Es la capa superior del modelo TCP/IP, correspondiente a las capas 5, 6 y 7 del modelo OSI (sesión, presentación y aplicación). Aquí se habla de SMTP y FTP, entre otros (**Figura 10**).

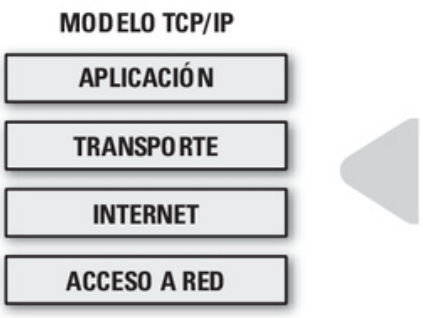


FIGURA 9.
Capas del modelo TCP/IP.
La inferior es la de acceso a red, y la superior, la de aplicación, la que está más cerca del usuario.

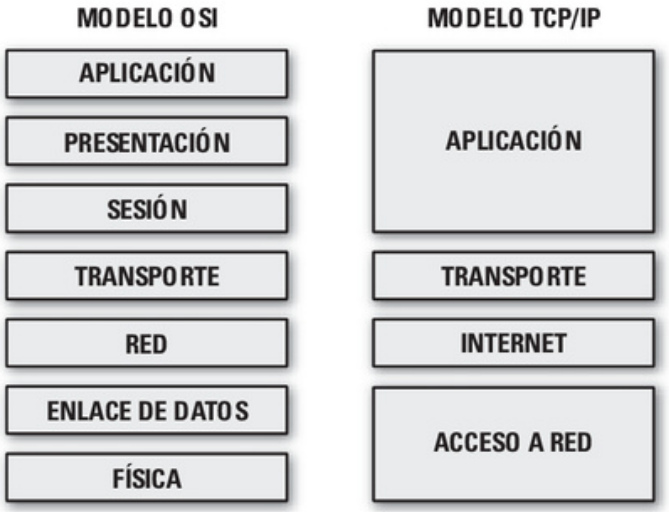


FIGURA 10. **Comparación entre las siete capas del modelo de referencia OSI y las cuatro del TCP/IP.**

PROTOCOLO IP

El protocolo IP está ubicado en la capa 3 del modelo OSI (capa de red). Es el principal de esta capa y permite el intercambio de datos en una red. Se trata de un protocolo **no orientado a la conexión**, ya que cada paquete que se envía puede tomar distintos caminos entre el origen y el destino, con lo cual pueden llegar desordenados. También es un protocolo **no fiable**, debido a que los paquetes de datos pueden dañarse o perderse.

Direcciones IP

Para identificar un dispositivo en una red, a cada uno se le asigna una dirección, llamada **dirección IP**, que si bien es única dentro de la red, puede no ser fija y modificarse de manera dinámica.

Las direcciones IP están formadas por una serie de **32 bits**, separados en **cuatro octetos** que se representan en formato decimal. Un ejemplo de esto, es 192.168.0.3.

El modelo TCP/IP está dividido en cuatro capas: acceso a red, Internet, transporte y aplicación

Ahora bien, estas direcciones se dividen en dos partes: **red** y **host**. La parte que indica la red define hacia qué red va dirigido el paquete de datos. Por otro lado, la parte de **host** señala hacia qué equipo dentro de esa red se lo dirige. Se dice que es un direccionamiento jerárquico, ya que existen diferentes niveles.

Las redes se dividen en cinco clases: **A**, **B**, **C**, **D** y **E**, identificadas con el primer octeto. Las de clase **A** son redes grandes, que cuentan con muchos dispositivos; las de clase **B** son medianas; las **C** son redes

pequeñas, con pocos dispositivos; en tanto que las D son multicast, en donde el envío de datos es recepcionado por múltiples dispositivos; y las E son para experimentación y prueba (Figura 11).

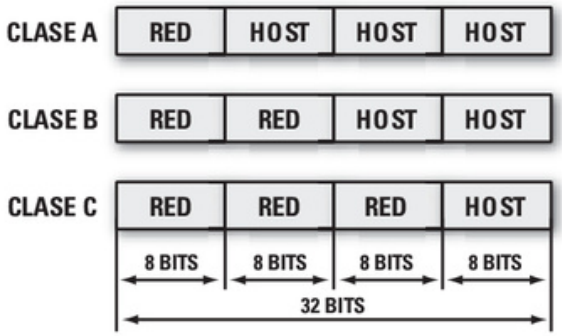


FIGURA 11. **Vemos cómo está formada una dirección IP. Cada octeto corresponde a la dirección de red o de host, según la clase de red.**

Resolución de nombres: DNS

Como mencionamos en secciones anteriores, cada dispositivo conectado a una red debe tener asignada una dirección IP, mediante la cual se podrán direccionar datos entre diferentes dispositivos. Con esta información, podría surgir la pregunta: ¿cómo puedo acceder a una página web sin saber su IP? La respuesta es muy sencilla: mediante el **DNS** (*Domain Name System* o sistema de nombre de dominio).

SISTEMA DE NOMBRE DE DOMINIO

El DNS es una base de datos distribuida y jerárquica en donde se almacena diferente información, entre la que se encuentra la asociación entre un nombre de dominio y una dirección IP.

Veamos un ejemplo. Supongamos que queremos acceder al sitio de **Google**. Para hacerlo, simplemente escribimos **www.google.com** en la barra de dirección del navegador. Ahora bien, también podríamos acceder a la misma página ingresando 209.85.195.99 en la barra, que corresponde a la IP del servidor web de Google (Figura 12).

Cuando decimos que el DNS es una base de datos distribuida, nos referimos a que no está ubicada en un servidor en particular, sino que se encuentra dispersa en diversos servidores. Por otra parte, es jerárquica porque el sistema de nombre de dominio

CLASE DE RED	BITS	1ER. BYTE	FORMATO	RANGO
A	0xxxxxxx	0 – 127	red.host.host.host	1.0.0.0 – 126.255.255.255
B	10xxxxxx	128 – 191	red.red.host.host	128.0.0.0 – 191.255.255.255
C	110xxxxx	192 – 223	red.red.red.host	192.0.0.0 – 223.255.255.255
D	1110xxxx	224 – 239	-	224.0.0.0 – 239.255.255.255
E	1111xxxx	240 – 255	-	240.0.0.0 – 255.255.255.255

TABLA 1. **Detalle de cada una de las cinco clases de redes.**

tiene una estructura de árbol, dividido en dominios y subdominios, cada uno de ellos se encuentra separado por un punto.

En un nombre de dominio completo, la jerarquía debe leerse de derecha a izquierda. Para el dominio **google.com**, el dominio de jerarquía más alta es **.com**. La raíz de este árbol se representa con un punto al final del nombre **google.com** (**Figura 13**).

SERVIDORES DNS

La función principal de los servidores DNS es traducir los nombres de dominio en direcciones IP. Existen distintos tipos de servidores DNS: **primarios** y **secundarios**. Cada dominio y subdominio posee un servidor primario y varios secundarios como respaldo ante una eventual falla. El primario es el que cuenta con la información original, en tanto que los secundarios obtienen la información solicitándola en forma periódica al primario.

La función de los servidores DNS es traducir los nombres de dominio en direcciones IP

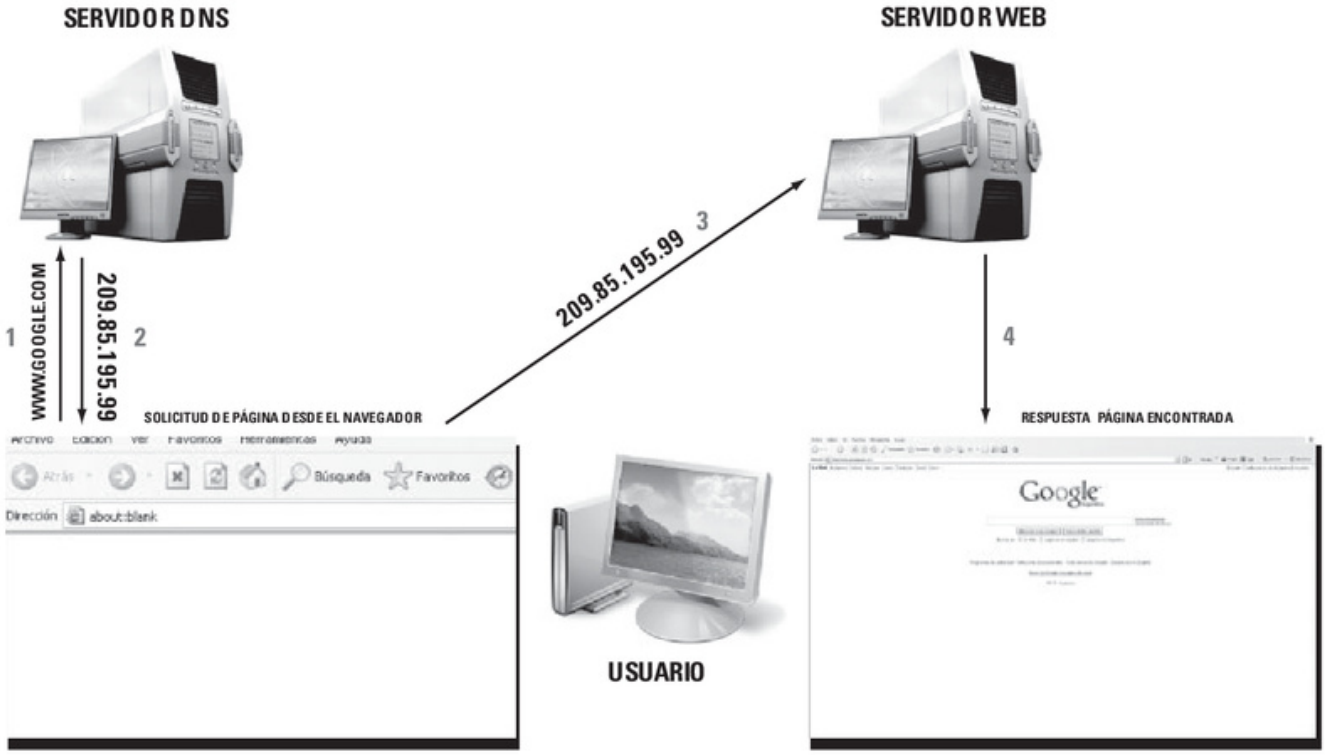


FIGURA12. **Podemos ver, en forma simplificada, lo que ocurre desde que ingresamos** www.google.com **en la barra de dirección del navegador hasta que logramos ver la página web correspondiente.**

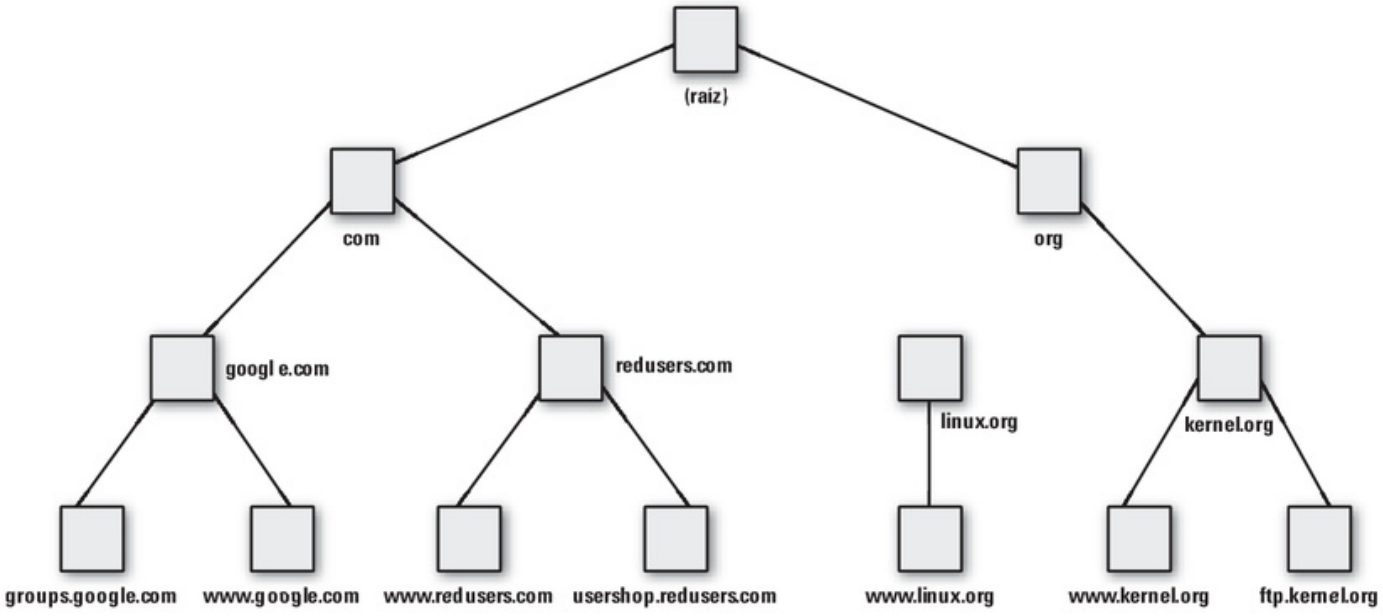


FIGURA 13. **Ejemplo de la estructura jerárquica del sistema de nombre de dominio. Vemos que la raíz se representa con un punto.**

CACHÉ DE DNS

Existe un mecanismo de caché mediante el cual se agiliza la búsqueda en servidores de DNS. El funcionamiento es de la siguiente manera. Cuando intentamos acceder a una dirección web, el primer servidor DNS al que se acude es al de **nuestro propio ISP** (*Internet Service Provider* o proveedor de servicios de Internet). Si este no la localiza en ese servidor, mediante una búsqueda jerárquica intenta ubicarla en otros servidores DNS hasta hallarla.

Ahora, supongamos que otro usuario del mismo ISP quiere acceder a la misma dirección web. En ese caso, el servidor DNS del ISP ya tiene almacenada esa dirección gracias a la búsqueda efectuada previamente; es decir que, a medida que un usuario intenta acceder a distintas direcciones, los servidores DNS van guardando esa información, mecanismo llamado **caché de DNS**. La información guardada en la caché del servidor DNS no queda allí para siempre, sino que se elimina después de un cierto tiempo, denominado **TTL** (*Time To Live* o tiempo de vida); este es un parámetro asignado por el administrador a cada servidor.



UDP y TCP

Los protocolos UDP y TCP están ubicados en la capa 4 del modelo OSI (de transporte). Ambos utilizan el protocolo IP para el transporte de los mensajes. La diferencia fundamental entre ellos es que UDP es un protocolo **no orientado** a la conexión, mientras que TCP está orientado a ella.

PROTOCOLO UDP

Su función es proporcionar comunicación entre las aplicaciones de dos equipos. Emplea el protocolo IP para el transporte de los mensajes y, al igual que este, es **no orientado a la conexión** y **no fiable**. En el primer caso, debido a que no se establece una conexión previa entre los dos equipos para la transmisión de los mensajes, con lo cual existe la posibilidad de que estos no lleguen ordenados al destino.

En el segundo caso, porque los mensajes pueden llegar dañados o perderse. Tampoco existe confirmación de llegada por parte del receptor, con lo cual no hay manera de saber si alcanzaron el destino correctamente.

Las aplicaciones que empleen este protocolo deberán prever la forma de asegurar que la información que llegue sea la adecuada.

Datagrama UDP

El datagrama UDP cuenta con cinco campos: cuatro de ellos corresponden a la cabecera, y el quinto, a los datos. La cabecera está compuesta por puerto de origen, puerto de destino, longitud del mensaje y suma de verificación. Los **puertos** son números de **16 bits** que permiten asociar las diferentes aplicaciones con un mismo host. Así, por ejemplo, **TFTP** (*Trivial File Transfer Protocol* o protocolo trivial de transferencia de archivos) emplea el puerto 69 de UDP. La longitud del mensaje se mide en bytes, incluyendo la cabecera (**Figuras 14 y 15**). Este protocolo es muy utilizado para la transmisión de video y sonido, en donde se necesita una alta velocidad de respuesta, sin importar que se pierdan algunos datos.

PROTOCOLO TCP

Este protocolo, al igual que el UDP, emplea IP para el transporte de los mensajes, pero a diferencia de aquel, está **orientado a la conexión** y es **fiable**.

Los protocolos UDP y TCP están ubicados en la capa 4 de transporte del modelo OSI

PROCESOS DE RESOLUCIÓN I

Si deseamos acceder al sitio **groups.google.com**, en primer lugar, el sistema se contactará con un servidor de DNS local. Este interrogará a un servidor de DNS raíz, preguntando por la dirección indicada. La respuesta será una lista de servidores para el dominio .com.



FIGURA 14. **Podemos ver la manera en que un mensaje UDP viaja sobre el datagrama IP.**



FIGURA 15. **El datagrama UDP está compuesto por el header y los datos. El header consta de cuatro campos: puerto UDP de origen, puerto UDP de destino, longitud del mensaje y suma de verificación.**

En el primer caso, porque se necesita establecer una conexión previa entre los dos equipos para la transmisión de los mensajes, con lo cual se garantiza que estos lleguen ordenados al destino; luego de la comunicación, se debe cerrar la conexión.

En el segundo caso, debido a que garantiza que los mensajes lleguen correctamente al destino. De esta forma, las aplicaciones que empleen este protocolo no deberán preocuparse por la integridad de los datos, porque de eso se encarga el propio protocolo TCP.

Fiabilidad en TCP

Dijimos anteriormente que el protocolo TCP es fiable. Quizá nos preguntemos cómo es posible esto, considerando que TCP viaja sobre el protocolo IP, que es no fiable. Pues bien, la forma de lograr esto es muy simple, y es que el receptor envíe un mensaje de confirmación (ACK o *acknowledgement*) al emisor cada vez que reciba un mensaje de él. En caso de que el emisor no reciba esta confirmación pasado un cierto tiempo, volverá a enviar el mismo mensaje.

PROCESOS DE RESOLUCIÓN II

Posteriormente, nuestro servidor interrogará a esos nuevos servidores por el nombre de dominio completo. De esa forma, obtendremos la lista de servidores correspondiente al dominio google.com para finalizar el proceso de resolución.

Conexión y desconexión en TCP

El proceso para una conexión se realiza en tres pasos. En primer lugar, el emisor manda un mensaje SYN (configuración dentro de la cabecera TCP), y si el receptor está en condiciones de establecer la comunicación, le envía un mensaje SYN + ACK al emisor. Luego, el emisor manda un mensaje ACK al receptor. Recién en ese momento se establece la comunicación (Figura 16).

Por otra parte, el proceso para realizar una desconexión se efectúa en cuatro pasos, en donde cada uno deberá enviar un mensaje FIN y recibir el ACK correspondiente (Figura 17).

UDP es un protocolo no orientado a la conexión; TCP, sí está orientado a ella

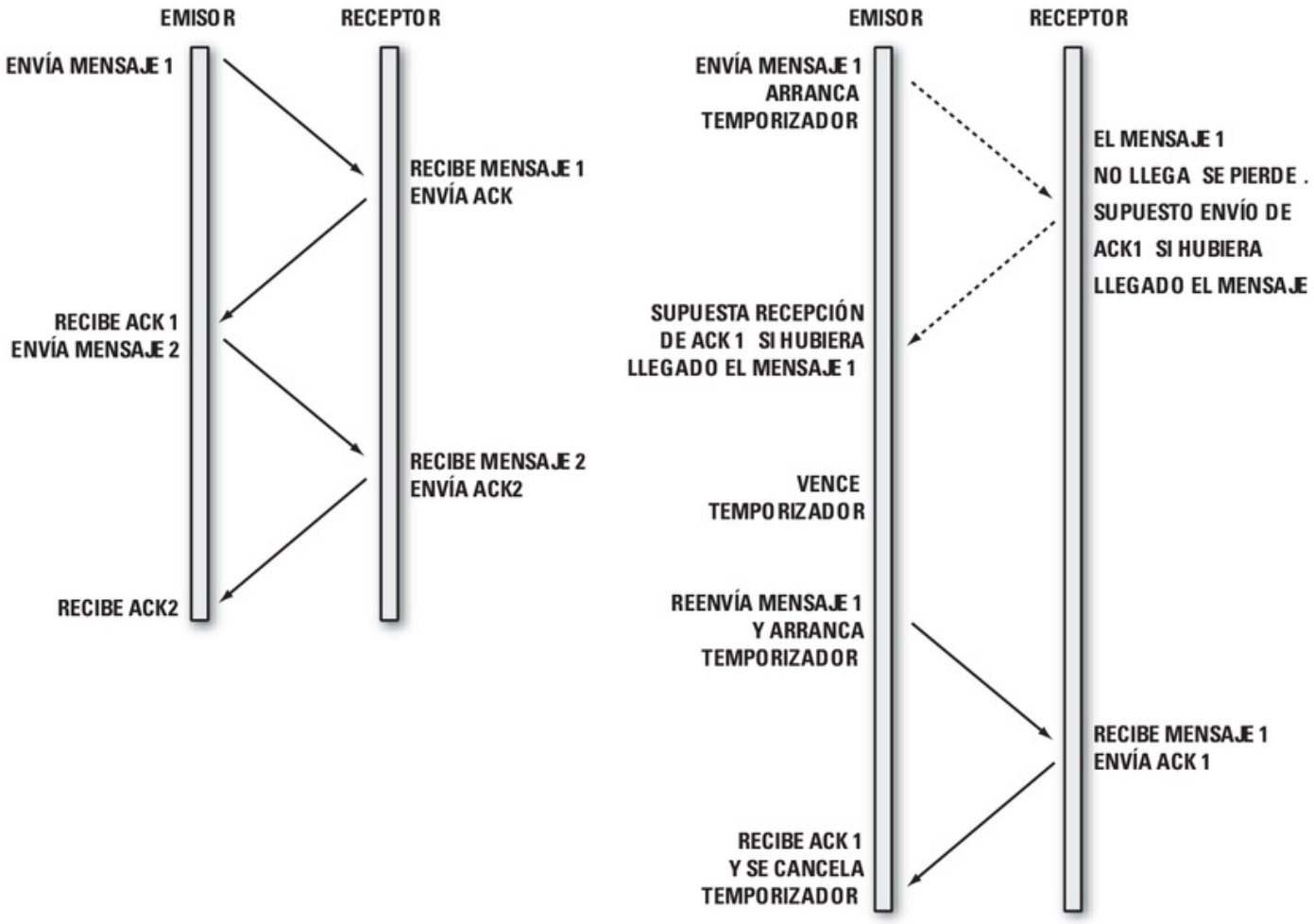


FIGURA 16. En el diagrama de la izquierda, vemos el envío de mensajes sin errores. En el de la derecha, vemos qué ocurre en caso de que el mensaje no llegue correctamente a destino.

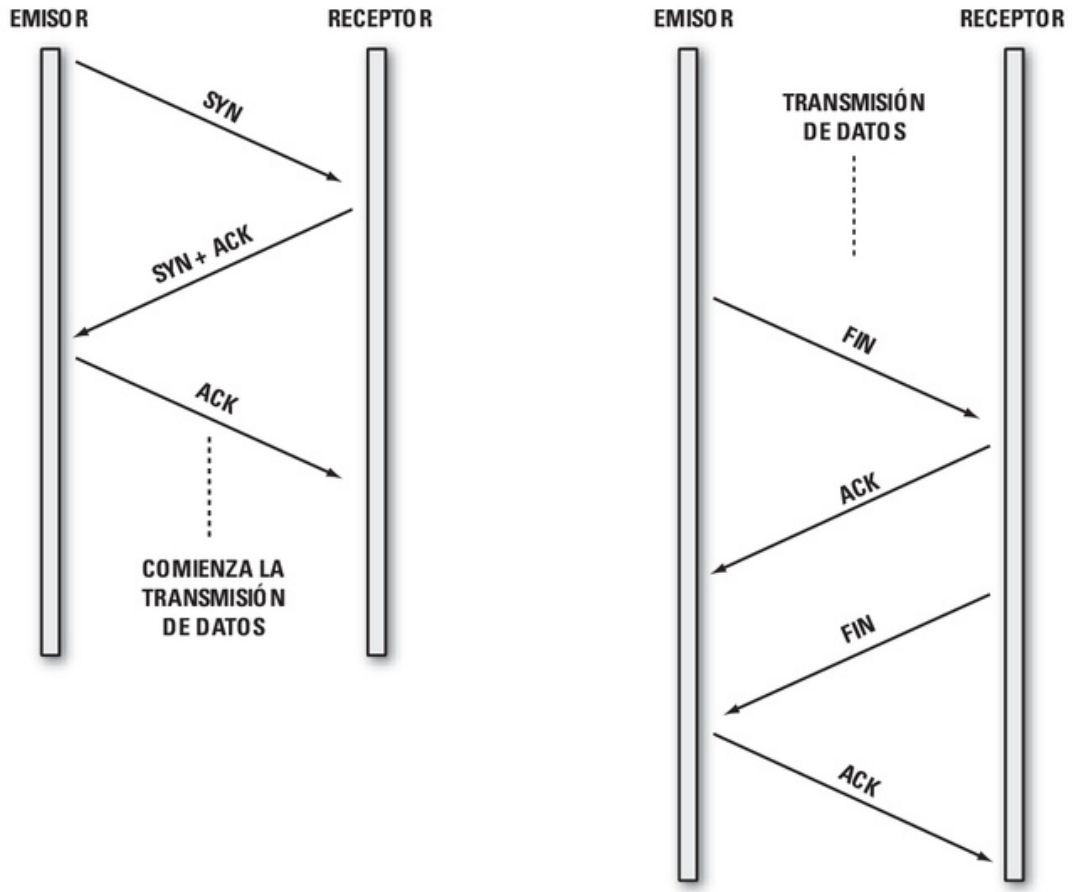


FIGURA 17. **En el esquema de la izquierda, observamos, paso a paso, el proceso para realizar una conexión en TCP (*three way handshake*). A la derecha, el proceso de desconexión (*four way handshake*).**

Conexión a servidores web

Cada vez que ingresamos en una página web desde nuestro navegador, estamos accediendo a un servidor web; se trata de programas que se ejecutan en una computadora y están permanentemente a la espera de que algún cliente realice una conexión. Estos clientes son los navegadores web que utilizamos.

EL PROTOCOLO HTTP

El **protocolo HTTP** está ubicado en la capa 7 del modelo OSI (capa de aplicación). Es un protocolo simple, basado en texto, utilizado por los navegadores y los servidores web para comunicarse.

Cuando desde nuestro navegador intentamos acceder, por ejemplo, al sitio **www.google.com**, estamos realizando una petición HTTP al servidor de Google, que responderá enviando la página solicitada.

HTML

El **lenguaje HTML** es el más empleado para la realización de páginas web. Con él es posible darle formato a un texto: modificar tipo de fuente, color y tamaño; así como también variar su estructura, como párrafos, tablas, etc. Para lograrlo, este lenguaje emplea etiquetas, que deben agregarse al texto cuyo formato deseamos cambiar. De esta forma, el navegador web las interpreta y permite mostrar en la pantalla un texto con un formato determinado (**Figura 18**).

Por ejemplo, para que el navegador vea un determinado texto en **negrita**, la página web debe contener ese texto dentro de la etiqueta **** de la siguiente manera: ****texto en negrita****. El navegador web, entonces, mostrará: **texto en negrita**.

CABECERA HTTP

Veamos un ejemplo de una cabecera HTTP correspondiente a la respuesta de un servidor web ante una petición de nuestro navegador:

```
HTTP/1.1 200 OK
Date: Fri, 02 Apr 2010 02:55:57 GMT
Server: Apache
Last-Modified: Tue, 28 Apr 2009
16:03:52 GMT
Accept-Ranges: bytes
Content-Length: 1386
Keep-Alive: timeout=15, max=100
Connection: Keep-Alive
Content-Type: text/html; charset=utf-8
```

La respuesta HTTP/1.1 200 OK nos indica que la petición fue correcta.

WEB SERVER VÍA TELNET

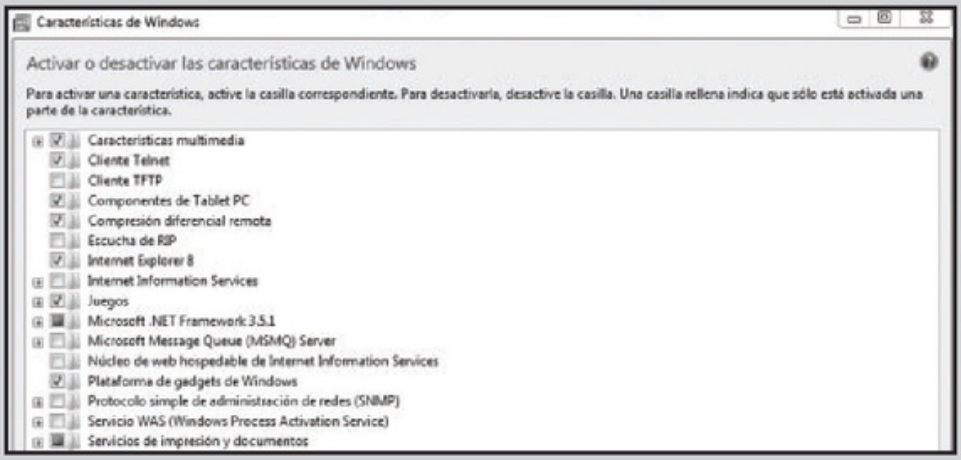
Telnet es un protocolo de la capa 7 del modelo OSI (capa de aplicación), que permite conectar un cliente con un servidor y enviarle comandos a este último a través del teclado. Utiliza una conexión TCP para mandar información. En el **Paso a paso 1**, realizaremos una conexión con el servidor web de Nic Argentina (**www.nic.ar**).



FIGURA 18. **Desde Internet Explorer, podemos ver el código fuente (HTML) de una página web haciendo clic en** Página/ Ver código fuente. **En este ejemplo, vemos el de** Nic Argentina (www.nic.ar).

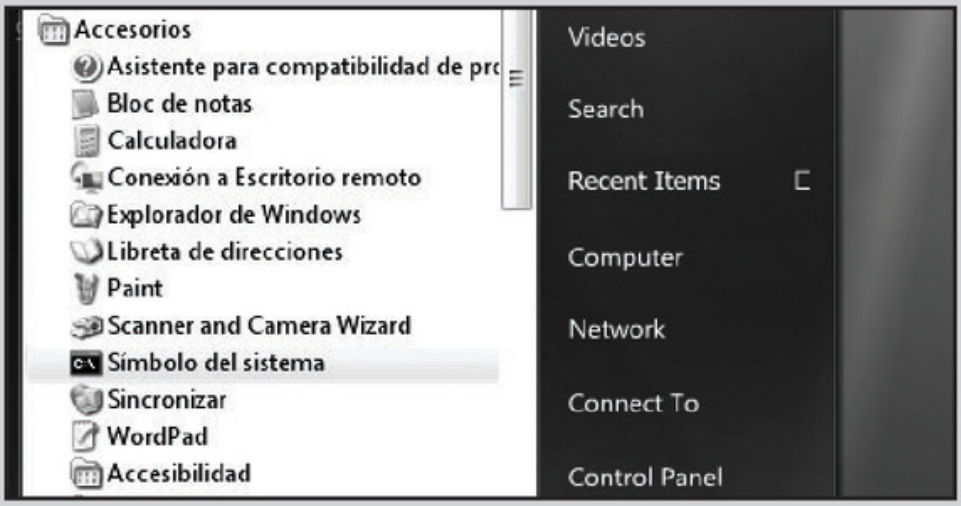
PASO A PASO /1
Conexión a un Web Server vía Telnet

1



Debe verificar que esté habilitada la aplicación Telnet. Para hacerlo, vaya a **`Inicio/Panel de control`** y haga clic en **`Desinstalar un programa`**. Posteriormente, presione **`Activar o desactivar las características de Windows`**. Aparecerá una ventana en la cual debe asegurarse de que estén marcadas las opciones **`Cliente Telnet`** y **`Servidor Telnet`**. Por último, pulse **`Aceptar`**.

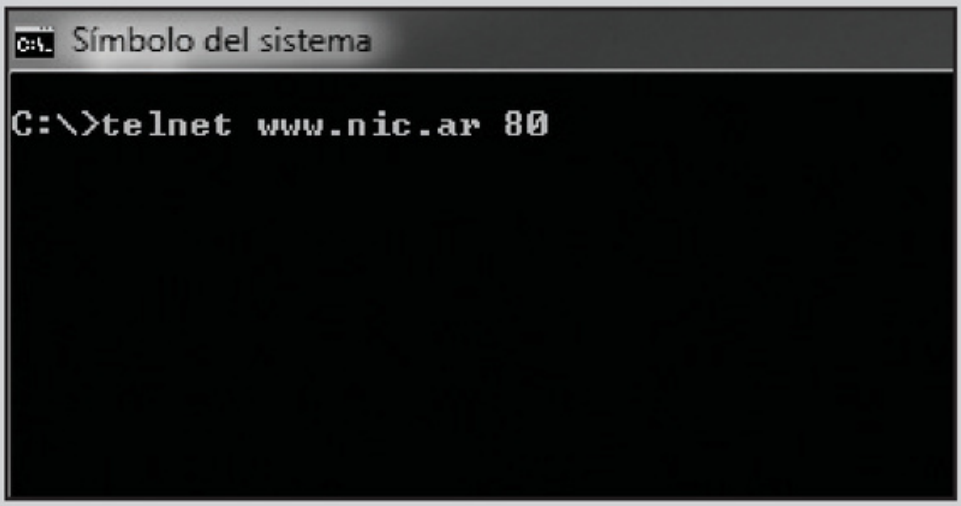
2



A continuación, abra una terminal, desde **`Inicio/Todos los programas/Accesorios/Símbolo del sistema`**. Para ver mejor el resultado del Telnet, es conveniente maximizar la ventana.

PASO A PASO /1 (cont.)

3



Está en condiciones de realizar la conexión al servidor web mediante Telnet. Escriba lo siguiente: **`telnet www.nic.ar 80`** y presione **`Enter`**. Lo que está haciendo es abrir una conexión TCP con el puerto 80 (HTTP) del servidor web de Nic Argentina.

4

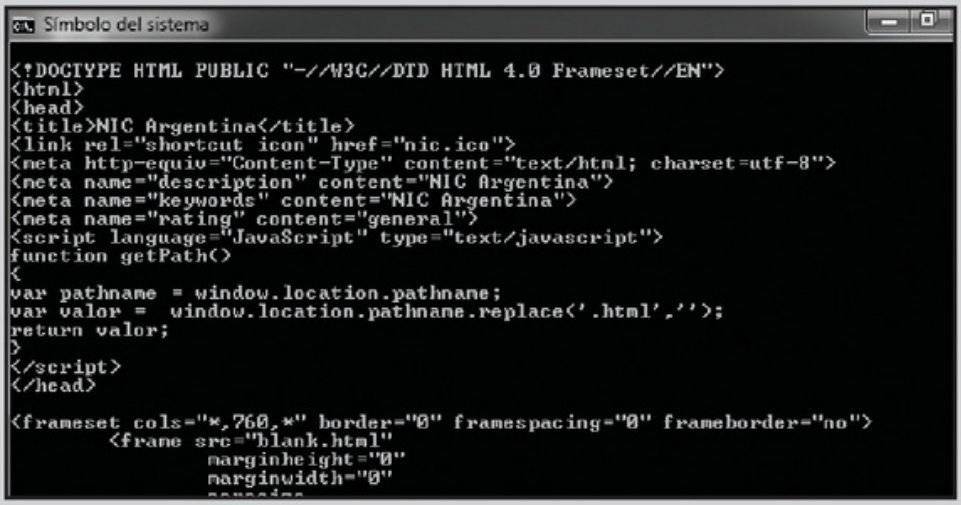
Verá que nos queda la terminal vacía, sin ningún texto. Debe darle la instrucción al servidor web para que envíe la página correspondiente. Escriba: **`GET /`** y presione la tecla **`Enter`**. Debe tener cuidado al hacerlo, ya que probablemente no aparezca en pantalla el texto ingresado. Esto se debe a que, por defecto, el eco local está desactivado.

NIC.AR

Es el organismo en Argentina que se dedica a regular el registro de los dominios terminados en **.com.ar**. Al registrar el sitio, la validez de los datos tiene valor de declaración jurada. Además, en su sitio web podemos consultar los dominios ya registrados.

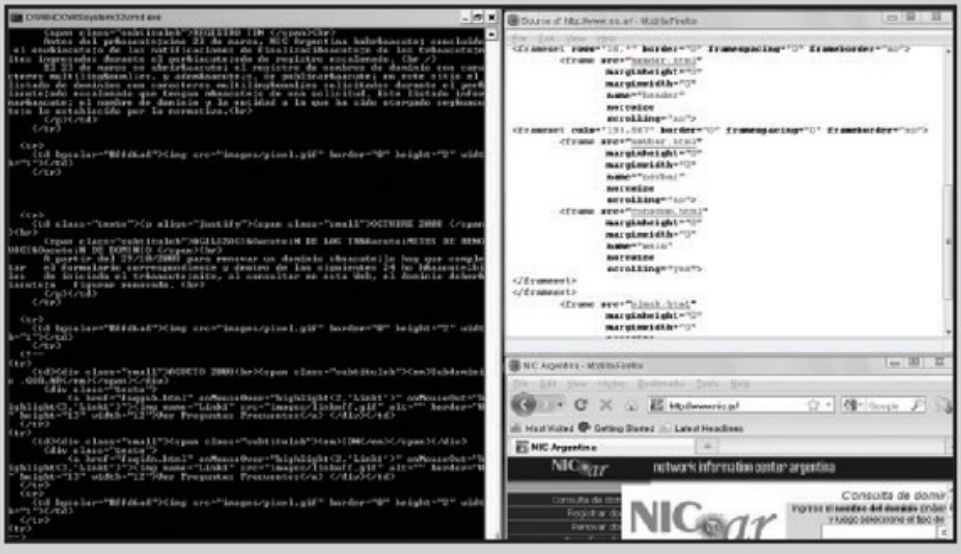
PASO A PASO /1 (cont.)

5



Verá en pantalla el código HTML crudo que entrega el servidor de Nic Argentina. Los navegadores se encargan de interpretarlo y mostrarlo en pantalla con el formato adecuado.

6



Vuelva a conectarse y, esta vez, pida, por ejemplo, uno de los framesets: **`GET / consdom.htm`** y pulse **`Enter`**. Obtenga más código HTML, de donde extraerá información sobre imágenes y texto.

El lenguaje HTML

En el **Paso a paso 1**, pudimos conocer cómo se comporta un navegador web; es decir, nos conectamos a un servidor **HTTP** iniciando una conexión **TCP** al **puerto 80** y le pedimos una página web.

Procedimos a analizarla sintácticamente y descubrimos que no podíamos mostrar nada porque necesitábamos otros objetos. Finalmente, comenzamos un proceso iterativo y recursivo para traer uno por uno los objetos, analizarlos y seguir así hasta obtener todos.

Pudimos observar que los objetos se indican dentro de **tags**, encerrados de esta forma: **<tag>**. El código HTML está encerrado entre los tags **<HTML>** y **</HTML>**. Algunos de los objetos que podemos encontrar son:

- Encabezado de página: entre **<HEAD>** y **</HEAD>**; contiene, entre otras cosas, el título de la página, encerrado entre los tags **<TITLE>** y **</TITLE>**.
- Cuerpo de la página: entre **<BODY>** y **</BODY>**.

Por otro lado, en el cuerpo de la página podemos hallar los siguientes:

- Framesets: ubicados entre los tags **<FRAMESET>** y **</FRAMESET>**; contienen otras páginas HTML incrustadas en porciones de la página actual.
- Imágenes, identificadas por el tag **<IMG SRC = "nombre.ext>**.
- Encabezados: identificados entre los tags **<Hx>** y **</Hx>**, donde la x es un número del 1 al 6 que corresponde al orden de encabezado, según su importancia y tamaño de fuente.
- Tablas: se definen entre los tags **<TABLE>** y **<TABLE>**. En el cuerpo de la tabla, cada fila se define entre **<TR>** y **</TR>**, y cada columna, entre **<TD>** y **</TD>**.
- Párrafos: entre los tags **<P>** y **</P>**.
- Separadores horizontales: indicados como **<HR>**.
- Saltos de línea: indicados como **
**.
- Links a otras páginas u objetos. El link se indica dentro del tag **<A HREF = "url">**, mientras que el texto comprendido entre este tag y **</A>** se muestra como un link (**Figura 19**).



Ethernet con PIC

Los microcontroladores PIC diseñados para Ethernet incorporan un módulo dedicado para esta función. En la mayoría de los casos, estos micros poseen gran cantidad de pines (64, 80 o 100) y son de montaje superficial. Sin embargo, podemos migrar una aplicación que originalmente no tenía conectividad Ethernet a través de módulos de Microchip que se conectan por SPI al microcontrolador.

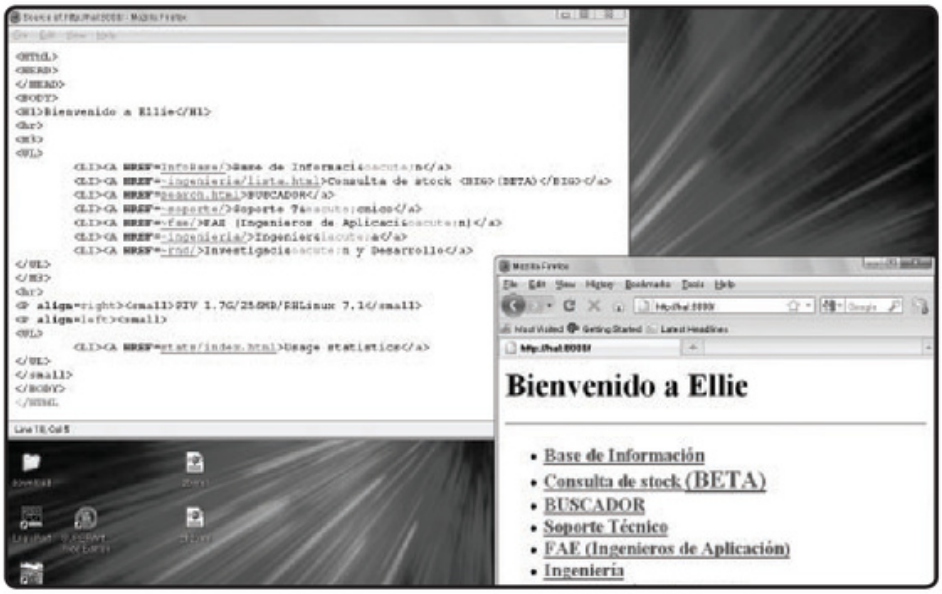


FIGURA 19.
Una página HTML muy simple, con listas y links, para servir de acceso en una intranet (red interna de una empresa).

Por ejemplo, podríamos conectar un módulo Ethernet al PIC18LF4620 y, de este modo, tener el sistema conectado a una red.

Para realizar una conexión Ethernet con PIC es posible recurrir al Microchip TCP/IP Stack, que se descarga de forma gratuita desde el sitio oficial del fabricante: **www.microchip.com/tcpip**.

A diferencia de la capa física, este conjunto de protocolos establece la conexión lógica entre el otro host y el microcontrolador. Además, incluye varios ejemplos, lo que permite optimizar el tiempo de desarrollo. Es importante destacar que, en los **Capítulos 4** y **5**, implementaremos el stack de Microchip en un micro de montaje superficial.

Una ventaja de estos micros es su reducido tamaño y el hecho de que, al incorporar el módulo Ethernet, los costos se reducen considerablemente. Además, veremos de qué forma añadir conectividad WiFi, y realizar telemetría y control a través de Internet (**Figura 20**).

Los microcontroladores PIC con módulo Ethernet se destacan por su robustez y facilidad de uso. Además, las librerías gratuitas de Microchip permiten una rápida implementación.



FIGURA 20.
Web Server comercial implementado con un microcontrolador PIC de 8 bits.

TCP/IP EN RABBIT

Vamos a explicar brevemente la implementación del modelo TCP/IP que realiza Rabbit, con miras a su estudio detallado en el **Capítulo 2**. En el año 2000, poco después de la salida al mercado del **Rabbit 2000**, Dynamic C ya contenía un stack TCP/IP completo y sin regalías. Por entonces, los módulos utilizaban un chip externo como el **RTL8109**, de la firma Realtek, para encargarse de la interfaz Ethernet.

En la actualidad, el **Rabbit 4000** incluye Ethernet integrada de 10 Mbps, y el **5000** provee de Ethernet integrada de 10/100 Mbps y hardware de Wi-Fi 802.11b/g.

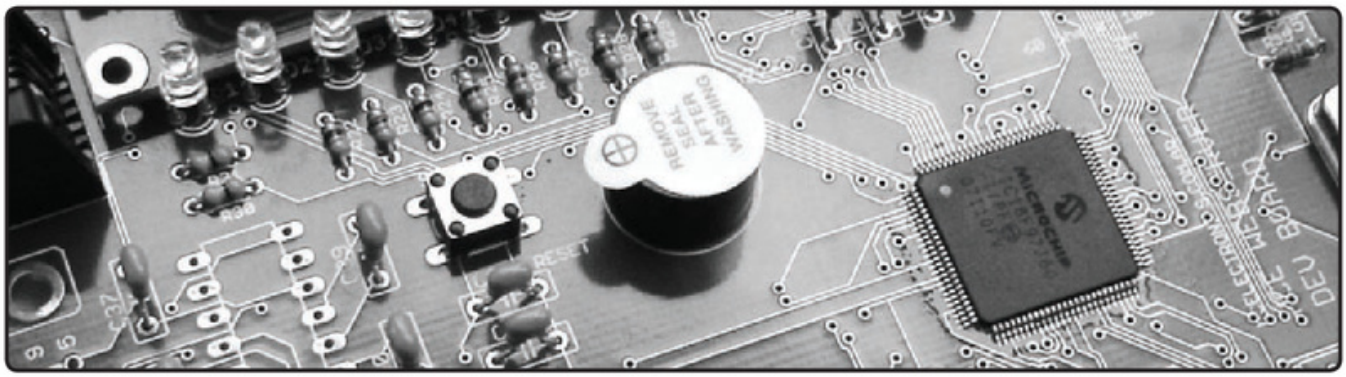
LA INTERFAZ DE PROGRAMACIÓN

La interfaz que provee **Dynamic C** guarda una cierta relación con la de **Berkeley sockets**, lo suficiente como para que el programador no deba aprender desde cero un nuevo entorno. Si bien muchas llamadas son diferentes, la arquitectura es muy similar y se domina con rapidez. Todo el stack corre cuando el usuario cede el control mediante una llamada a la función `tcp_tick()`. La ocupación de memoria se administra al momento de compilar mediante macros que definen, por ejemplo, la cantidad de sockets y servidores.

LOS SERVICIOS

Las bibliotecas de funciones de Dynamic C incorporan código fuente para muchas aplicaciones, veamos cuáles son:

- **Servidor HTTP con SSI** (*Server Side Includes*) y un lenguaje script con sintaxis similar a C, llamado **RabbitWeb**.
- **Cliente POP y SMTP** para recibir y enviar correo, respectivamente.



ESPECIFICACIONES HTML

La especificación corriente de HTML puede obtenerse de la página de estándares de W3C: **www.w3.org/TR**. Mientras se trabaja en la versión 5, el estándar vigente es HTML 4.01. Existen muchos más objetos, como botones, selectores y listas.

- Cliente DHCP.
- **Cliente y servidor FTP** para transferir archivos.
- Resolución de nombres (**DNS**).
- Soporte para **DDNS** (DNS dinámico, mediante un **parser HTML**).
- **SSL** y **AES** para conexiones seguras.

Estos son solo algunos de los **códigos fuente** que incorporan las bibliotecas de Dynamic C.

RabbitWeb es un lenguaje script para armar páginas web dinámicas

Otras **alternativas**

En las páginas anteriores analizamos la propuesta de **Microchip** y la de **Rabbit**, pero estos no son los únicos fabricantes de microcontroladores. Afortunadamente, la oferta es muy amplia y hay productos para todos los gustos. Si bien algunos desarrolladores lideran en ventas, otros lo hacen en innovación, y hay quienes marcan tendencias en los inicios, por lo que todos tienen algo de lo que podemos aprender.



Freescale

El coloso heredero de Motorola, por el momento, no parece interesarse en networking para 8 o 16 bits. Sus soluciones están orientadas a su línea de **32 bits Coldfire**, mediante un stack que provee sin cargo y que fue desarrollado por la firma **InterNiche**.

Texas Instruments

Este gran innovador tiene el foco de networking en sus líneas de 32 bits, entre ellas, **OMAP**, basada en núcleos **ARM** de alta gama; y **Stellaris**, la línea **ARM Cortex-M3**. Sin embargo, provee una nota de aplicación que plasma un hardware y un stack de código abierto basado en el controlador **CS8900A** de Crystal Semiconductor, para ingresar en el networking de la mano del **MSP430**. La firma Rowley Associates, fabricante de un excelente entorno de desarrollo y compilador **C/C++**, brinda, además, el hardware y un stack basado en **uIP** (stack simplificado para microcontroladores).

Atmel AVR

Este micro, caracterizado por su fuerte impacto en los aficionados –que terminan proveyendo gran parte del soporte adicional–, no posee controlador Ethernet. Existe en la Web una gran cantidad de proyectos, muchos basados en el controlador **RTL8019** de Realtek, en los que el stack TCP/IP está basado otra vez en **uIP**. El foco de Atmel en networking está puesto en la línea **AVR32**, una versión extendida del core AVR a 32-bits, y sus micros con núcleo **ARM**.

WIZnet

Hay una opción adicional para quienes no quieren abandonar su micro, y es utilizar módulos de networking. Hemos visto lo que todos tienen para ofrecer y, seguramente, tenemos el nuestro preferido.

Si ninguna de las opciones anteriores nos satisface, todavía podemos ingresar en el mundo del networking, dado que el fabricante coreano **WIZnet** ofrece una serie de chips y módulos prearmados que no sólo se encargan de la interfaz Ethernet, sino que, además, nos permiten delegarles el **stack TCP/IP**.

Los chips

El servicio provisto por los chips consta de una interfaz **socket level** a TCP o UDP para algunos sockets (**4** en el **W3150A+** y **8** en el **W5300**).

El chip W5300 soporta Ethernet 10/100, hasta ocho sockets y PPoE para conectarse a Internet

EL STACK TCP/IP

Un stack completo requiere mucha RAM, particularmente, para **packet buffers**, recuperar datagramas IP fragmentados y ensamblar el máximo de un segmento TCP. Los módulos Rabbit han incorporado lo suficiente como para que quepa el **stack completo**.

No existen servicios como Web Server o cliente DNS, solo comunicación por un socket. El fabricante ofrece notas de aplicación y **código C** para, por ejemplo, **ATmega128**, con el objetivo de implementar algunas de estas funcionalidades. La implementación de **ICMP** es bastante reducida, y los chips no parecen soportar fragmentación **IP**.

La forma de configurar los chips es lo que esperaría un desarrollador acostumbrado a **SPI** y periféricos, pero sin demasiados conocimientos sobre networking. Esto resulta más atrayente para quien piensa en su conexión TCP/IP como una forma de reemplazar un cable, en oposición a programar una aplicación utilizando una interfaz similar a **Berkeley sockets** como, por ejemplo, **Rabbit**.

Los módulos

Existen, fundamentalmente, dos tipos de módulos: unos proveen una interfaz **RS-232**, y los otros, acceso al chip:

- **Módulos con acceso al chip**: presentan un conector **RJ-45** y tiras de pines para su conexión a la PCB de control. No existe I/O, solo una interfaz para ser controlada por un micro. El módulo por sí mismo únicamente sirve como interfaz de networking para un microprocesador.
- **Módulos con interfaz RS-232**: tienen un conector **DB-9** y otro para la fuente de alimentación, además del **RJ-45**. Incorporan un micro para brindar cierta funcionalidad en la configuración de servicio.

Algunos de los módulos disponibles para conexión inmediata son el **WIZ811MJ**, basado en el chip **W5100** (**Figura 21**); y el **WIZ830MJ**, basado en el chip **W5300**. El primero provee de una interfaz **SPI**, mientras que el segundo ofrece un bus de 8 o 16 bits. Ambos contienen el conector **RJ-45** y una tira de pines para conexión a un proyecto.

FIGURA 21.
El módulo WIZ811MJ, basado en el chip W5100, provee de una interfaz SPI para conexión con un micro.



ALTERNATIVAS

Existen algunos fabricantes que han desarrollado alternativas basadas en clones del 8051; entre ellos, el TINI, de Dallas Semiconductor (hoy Maxim). Fabricantes como Silicon Labs ofrecen cores 8051 compatibles de hasta 100 MIPS y kits para agregar networking.

Multiple **choice**

▶1 ¿Cuál de las siguientes redes puede tener una extensión de hasta 100 km?

a- LAN.
b- MAN.
c- WAN.
d- Ninguna de las anteriores.

▶2 ¿Cuál de las siguientes redes suele emplearse, generalmente, en el ámbito de un edificio u organización?

a- LAN.
b- MAN.
c- WAN.
d- Ninguna de las anteriores.

▶3 ¿Cuál de las siguientes topologías de redes puede considerarse como varias topologías en estrella?

a- Malla.
b- Bus.
c- Anillo.
d- Árbol.

▶4 ¿Cómo se llama la red en la que ambos dispositivos pueden transmitir y recibir, pero no al mismo tiempo?

a- Simplex.
b- Half duplex.
c- Full duplex.
d- Ninguna de las anteriores.

▶5 ¿Cómo se llama la red en la que los dos dispositivos transmiten y reciben simultáneamente?

a- Simplex.
b- Half duplex.
c- Full duplex.
d- Ninguna de las anteriores.

▶6 ¿Cómo se llama la red que envía sus datos de manera unidireccional?

a- Simplex.
b- Half duplex.
c- Full duplex.
d- Ninguna de las anteriores.

Respuestas: 1-c, 2-a, 3-d, 4-b, 5-c, 6-a.

Capítulo 2

Networking con Rabbit



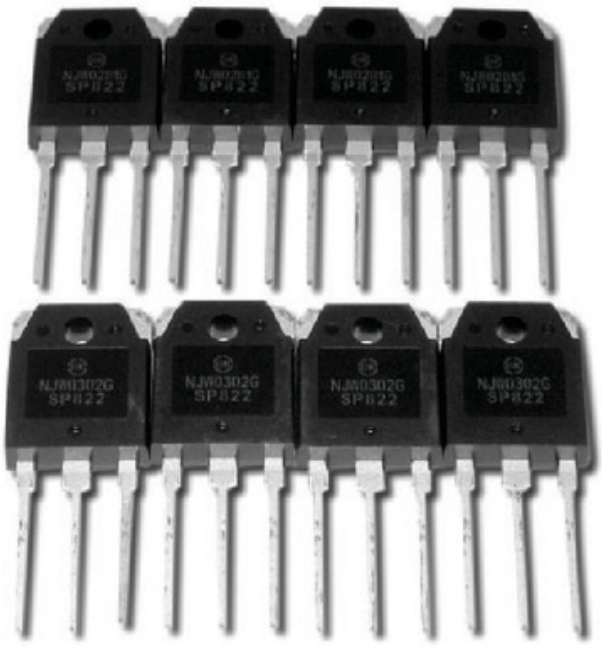
Conoceremos la interfaz de programación que el fabricante provee para desarrollar aplicaciones.

RedUSERS.com

Las funciones de networking en Rabbit

Todo el proceso de inicialización y operación del hardware de networking se maneja a través de una serie de **libraries** (librerías), de modo que el usuario no deba interactuar con el hardware. La principal ventaja de este conjunto es **DCRTCP.LIB**, que contiene todas las funciones esenciales del **stack TCP/IP**.

Incluiremos esta library mediante una directiva, pero antes de hacerlo, deberemos definir nuestra dirección IP, máscara, router y DNS. Por supuesto que es posible recibir toda la configuración mediante **DHCP** (*Dynamic Host Configuration Protocol*), pero para mantener las cosas simples y bajo control, iremos paso por paso. Un ejemplo se ve en el siguiente código.



```
#define TCPCONFIG        0
#define USE_ETHERNET     1

#define MY_IP_ADDRESS   "192.168.1.54"
#define MY_NETMASK     "255.255.255.0"

#define MY_GATEWAY       "192.168.1.1"
#define MY_NAMESERVER    "200.42.0.108"

#use "dcrtcp.lib"
```

INICIALIZACIÓN DE LA INTERFAZ

Para inicializar la interfaz –en este caso, Ethernet–, debemos realizar un llamado a la función **sock_init()**. Esta vuelve inmediatamente y, como asignamos una dirección fija, ya podemos comenzar a trabajar. En el caso de DHCP o de WiFi, como veremos más adelante, es necesario esperar a que termine el proceso de inicialización (**Figura 1**).

MANEJO DEL STACK

Como adelantamos en el **Capítulo 1**, todo el stack corre cuando el usuario cede el control mediante una llamada a la función **tcp_tick()**. Esta función deberá ser llamada periódicamente para poder atender al buen funcionamiento del stack TCP/IP. Para todas las otras tareas que vayamos a realizar, disponemos de funciones específicas.

Por ejemplo, existe una que nos permite demorar la aceptación o el rechazo de una conexión TCP a la espera de sockets libres, **tcp_reserveport()**, que se emplea para evitar que las conexiones sean rechazadas en el tiempo entre que se cierra una y se está abriendo otra.

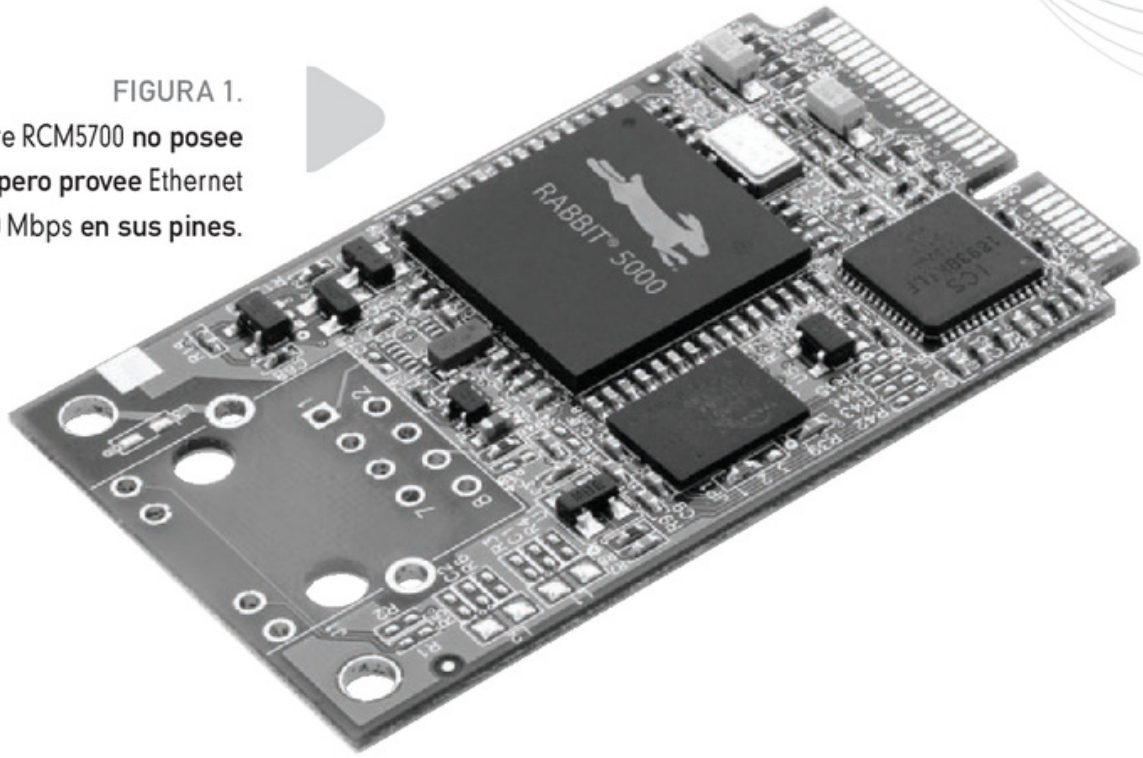


FIGURA 1.
El minicore RCM5700 no posee conector RJ-45, pero provee Ethernet de 10/100 Mbps en sus pines.

SERVICIOS

En el Capítulo 1, vimos que Dynamic C provee de código fuente para varios servicios, como los siguientes:

- Servidor HTTP con SSI (*Server Side Includes*) y un lenguaje de script con sintaxis similar a C llamado RabbitWeb.
- Cliente POP y SMTP para recibir y enviar correo, respectivamente.
- Cliente CDP (*Dynamic Host Configuration Protocol*).

Todo el stack corre cuando el usuario cede el control mediante una llamada a la función tcp_tick()

- Cliente y servidor FTP para transferir archivos.
  -Resolución de nombres (DNS).
  -Soporte para DDNS (DNS dinámico, con un un parser HTML).
  -SSL y AES para conexiones seguras.

SOCKETS

Para cualquier otra actividad que no sea resuelta por estos servicios o para UDP, se brinda acceso directo a los sockets:

- Podemos abrir sockets TCP llamando a la función `tcp_open()` y sockets UDP con `udp_open()`.
- Podemos crear servidores que esperen una conexión TCP en determinado port llamando a `tcp_listen()`.
- Establecido el socket TCP, enviamos y recibimos información con `sock_read()` y `sock_write()`.
- En UDP, utilizamos `udp_send()` y `udp_recv()`.

El entorno, en general, es similar al que encuentra alguien familiarizado en programar aplicaciones de networking (Berkeley sockets). La cantidad de conexiones y el tamaño de los buffers se limitan operando sobre algunas macros, previamente a la inclusión de DCRTCP. Por ejemplo, para setear el número máximo de sockets a seis hacemos lo siguiente:

```
#define MAX_TCP_SOCKET_BUFFERS 6
```

RESOLUCIÓN POR DNS

Si queremos iniciar una comunicación y solo disponemos del nombre de host, deberemos resolver este nombre a una dirección IP. El proceso de resolución es una búsqueda iterativa, como vimos en el Capítulo 1, pero en este caso todo lo que debemos hacer es utilizar la función `resolve()`.

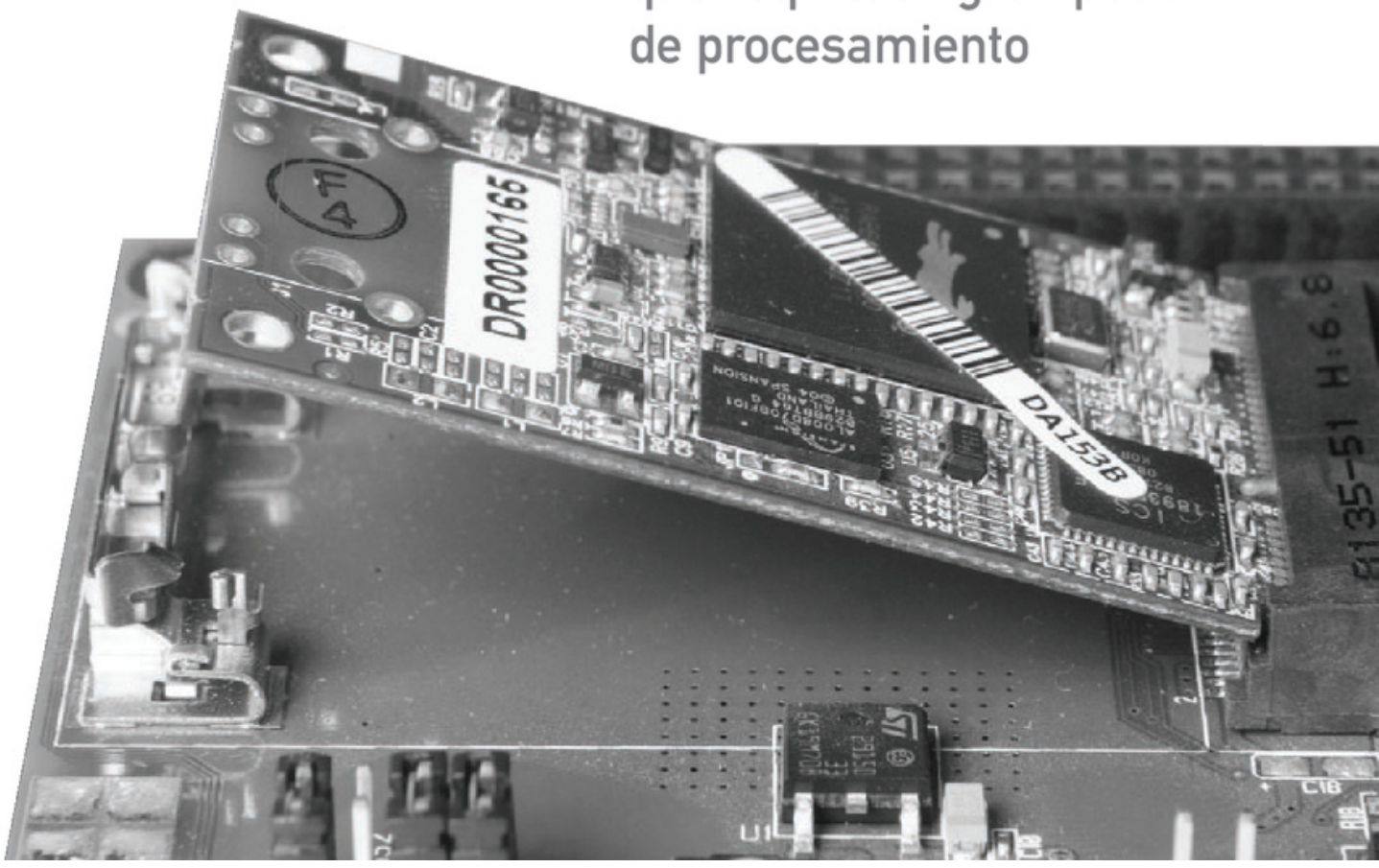
Esta devuelve una dirección IP en un formato que puede ser utilizado por las demás funciones del stack TCP/IP. Un ejemplo se ve en el código siguiente:

```
resolve("www.ldir.com.ar");
```

CARGA DE ARCHIVOS EN MEMORIA

Para algunos de los servicios, como servidor FTP o web (HTTP), necesitaremos referirnos a archivos. Como estamos programando en un micro sin un sistema operativo que nos respalde, el concepto de file system o sistema de archivos no existe.

SSL y AES son operaciones que requieren gran poder de procesamiento



Si bien podemos implementarlo, existen maneras más simples y productivas de trabajar. Lo que los servicios en realidad necesitan es una referencia a un bloque de memoria donde esté determinado un contenido específico. Después, asociaremos este contenido con un nombre mediante un directorio.

Así, al momento de compilar, Dynamic C toma un archivo del disco de la PC, lo aloja en un espacio de la memoria flash de Rabbit, y genera un puntero a esa posición. Esto se realiza mediante una directiva, que vemos en el siguiente código:

```
#ximport "index.html"   index_html
```

El primer elemento es el nombre del archivo en la PC y el segundo, el del puntero. Así, `index_html` apunta al contenido de **index.html**, copiado a la memoria flash de la PC al momento de compilar (**Figura 2**).

SERVIDOR HTTP

El servidor HTTP es el encargado de servir los elementos de una página web. Como tal, requerirá de un directorio donde tendrá referencias para traducir los nombres de los archivos (URLs) a su posición en la flash. De modo similar, precisará una tabla para indicar al navegador el tipo de archivo (tipo MIME) que va a servir, cuando el navegador lo interrogue y, en algunos casos, un tercer elemento que le señale al servidor mismo cómo procesar ese

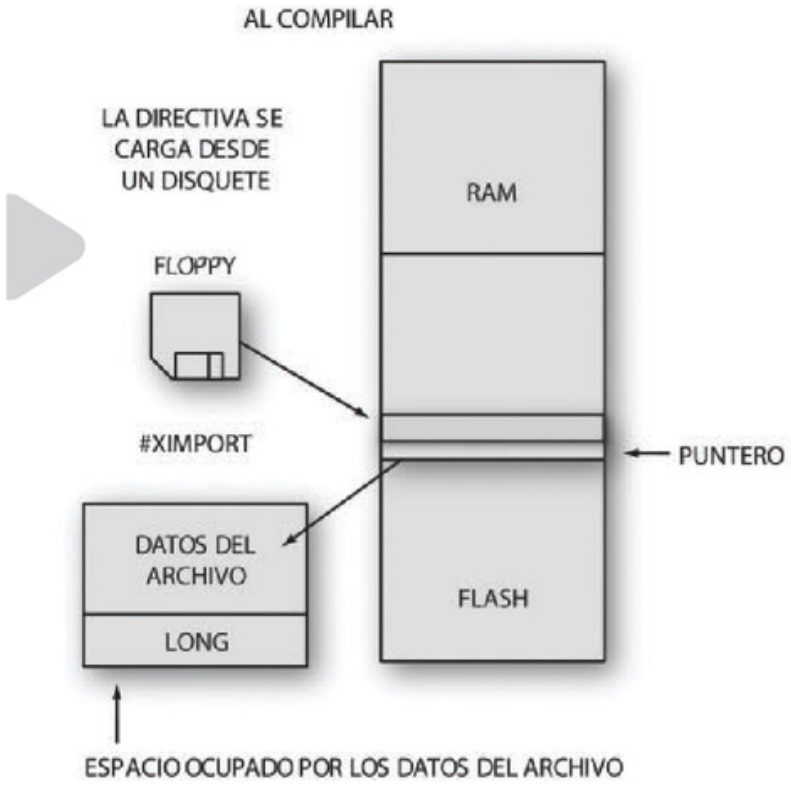


FIGURA 2.
La directiva `#ximport` copia el contenido del archivo de la PC a la flash de Rabbit.

archivo (qué handler utilizar). Esto se resuelve mediante dos estructuras, en las cuales se cargan los valores necesarios. Para facilitar esta operatoria, Dynamic C incluye una serie de macros que logran la simplificación del proceso.

Por ejemplo, para definir la tabla de tipos MIME, procedemos del modo que muestra el código:

```
SSPEC_MIMETABLE_START
        SSPEC_MIME_FUNC(".shtml",
"text/html", shtml_handler),
        SSPEC_MIME(".html", "text/html"),
        SSPEC_MIME(".gif", "image/gif"),
        SSPEC_MIME(".cgi", "")
SSPEC_MIMETABLE_END
```

Vemos que **SSPEC_MIME_FUNC** permite definir el handler por utilizar para este tipo de archivos, mientras que **SSPEC_MIME** es una entrada normal de la lista de tipos MIME.

Para definir el directorio de un servidor HTTP, realizamos lo que se muestra en el código siguiente:

```
SSPEC_RESOURCETABLE_START
        SSPEC_RESOURCE_XMEMFILE("/",
index_html),
        SSPEC_RESOURCE_XMEMFILE("/mues-
tra.shtml", muestra_html),

        SSPEC_RESOURCE_
ROOTVAR("mivar",&mivar, INT16,"%d"),

        SSPEC_RESOURCE_FUNCTION("/mifun.
cgi", setup)
SSPEC_RESOURCETABLE_END
```

Aquí, **SSPEC_RESOURCE_XMEMFILE** es un "archivo" en xmem, **SSPEC_RESOURCE _ROOTVAR** identifica a una variable y **SSPEC_RESOURCE_FUNCTION** identifica a una función **SSI** o **CGI**. Ambas se aplican en un contexto que veremos más adelante. También es posible agregar archivos en tiempo de ejecución y servirlos, por ejemplo, desde una tarjeta mini-SD.

La cantidad de servidores HTTP dispuestos a atender pedidos es dos; podemos modificar este dato agregando una macro antes de incluir la library de HTTP. Por ejemplo, para cuatro servidores tenemos el siguiente código:

```
#define HTTP_MAXSERVERS 4
#use "http.lib"
```

El servidor se inicializa llamando a la función **http_init()**. El manejo del servidor lo lleva adelante la función **http_handler()**, que deberemos llamar periódicamente. Esta a su vez llama a **tcp_tick()**, manteniendo activo el stack.

Para un ejemplo de esto ver el código siguiente:

```
http_init();
        tcp_reserveport(80);
        while(1)
                http_handler();
```

El directorio del servidor y los tipos MIME se arman como estructuras

Servidores web SSI y CGI

SSI (*Server Side Includes*) y CGI (*Common Gateway Interface*) son dos formas estándar de generar páginas dinámicas. La primera permite que el servidor identifique tags en el SHMTL (HTML con SSI) y los reemplace por el valor de una variable o la ejecución de una función (Figura 3). La segunda asigna una URL (*Uniform Resource Locator*, dirección de una página web) para que se ejecute una función a pedido. Esto es soportado por todo servidor web que se precie de tal; solo debemos utilizar la extensión SHTML para el archivo y, en Rabbit, indicar el handler como vimos anteriormente.

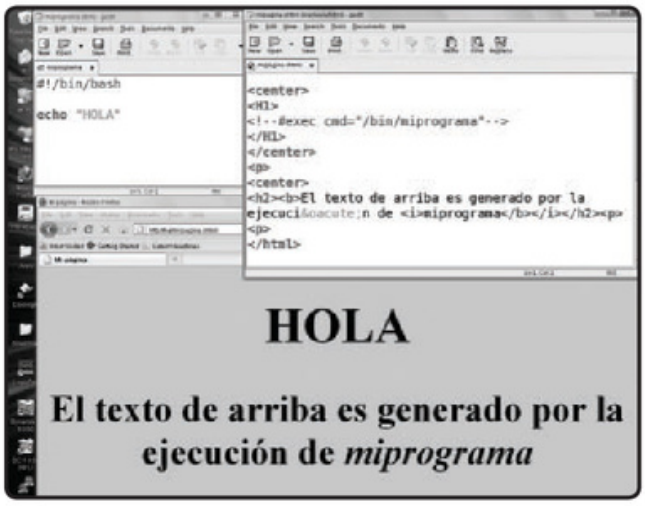


FIGURA 3. Ejecución de un programa mediante SSI en Apache. A la izquierda, el programa; a la derecha, el SHTML.

Los tags se introducen como comentarios HTML; por ejemplo, para mostrar una variable y ejecutar una función veamos el siguiente código:

```
<!--#echo var="mivar"-->
        <!--#exec cmd="mifun"-->
```

La forma en que el servidor web vincula los nombres dentro del SHTML con el resto del sistema depende de cada implementación en particular. Para la ejecución de funciones CGI, un servidor muy utilizado, Apache, permite definir un directorio en el cual se alojarán los archivos que podrán ser ejecutados. Estos, a su vez, pueden ser cualquier tipo de ejecutable que maneje el sistema donde reside el servidor. Este directorio es, generalmente, `/cgi-bin`.

SERVIDOR HTTP

Un servidor HTTP es una tarea que abre un socket TCP en el puerto 80 y espera una conexión. Cuando la recibe, la acepta y aguarda un tiempo por un pedido. Si no recibe nada, cierra la conexión; y si recibe un pedido, lo cumplimenta y luego la cierra.

Datos útiles para tener en cuenta

UTILIDAD

Desde el punto de vista de un sistema dedicado, podemos mostrar en una página web las variables de proceso de nuestra aplicación y ejecutar procesos a pedido. Nuestro sistema puede no solo tener una interfaz de control, sino que también es capaz de mostrar su estado de operación. El usuario obtiene, entonces, una presentación más amigable y profesional.

AUTENTICACIÓN

El proceso de autenticación consiste en solicitar un nombre de usuario y una clave. Uno de los métodos para hacerlo, `basic`, se centra en el envío en limpio de la información pedida.

Otro, más avanzado y seguro, se denomina `digest` e intercambia información generada a partir de la que se quiere autenticar, sin que esta se envíe realmente por la red. Esto evita que esa información sea leída en tránsito.

EN RABBIT

En este entorno, los nombres de la **función** y la **variable** deben existir en el directorio del servidor y apuntar al objeto respectivo, como vimos al analizar la estructura de directorio anteriormente.

Autenticación

Cuando queremos proteger el acceso a una página o limitar la ejecución de una función, necesitamos identificar a quién intenta acceder y establecer de algún modo cómo se relaciona esto con los archivos.

Una forma utilizada en el servidor Apache es leer un archivo de nombre `.htaccess`, que reside en el mismo directorio que los archivos protegidos. En él se identifican los usuarios o grupos de usuarios que tienen permiso para acceder a los archivos de ese directorio. Usuarios y grupos pueden, a su vez, definirse en el mismo `.htaccess`, en otro archivo de configuración del servidor o, incluso, en uno del sistema. Cuando haya un intento de acceso a estos archivos, el servidor solicitará autenticación al navegador, y este mostrará una



ventana que pide identificarse e indica el **realm** (entorno) correspondiente.

Entorno Rabbit

En este entorno necesitamos definir usuarios y grupos. Los archivos se asocian a grupos, y los usuarios pertenecen a uno o varios de ellos. Cuando un archivo tiene permiso, ya sea de lectura o de escritura, asociado a uno o más grupos, solo quienes pertenezcan a alguno de ellos podrán acceder a esos archivos y realizar dicha acción.

Para indicar que un elemento requiere autenticación, contamos con una entrada especial en el directorio del servidor, como se muestra en el código:

```
SSPEC_RESOURCE_P_XMEMFILE("/secret.
shtml",secret_html,"Secreto",1,0
,SERVER_HTTP, SERVER_AUTH_ANY),
SSPEC_RESOURCE_P_FUNCTION("/mifun.
cgi",mifun,"Secreto",1,0,
        SERVER_HTTP,SERVER_AUTH_ANY)
```

Un script CGI es un programa que se ejecuta en respuesta a un pedido de página

El valor **1** (bit 0 seteado) indica que los usuarios del grupo **0** podrán leer **/secret.shtml** o ejecutar **/mifun.cgi**. El valor **0** señala que ningún grupo tiene permiso de escritura.

El **realm** (Figura 4), en este caso, es **Secreto**. A continuación, creamos el usuario **cone** con clave **jito**, y lo asociamos al grupo **0**.

```
uid=sauth_adduser("cone", "jito",
SERVER_HTTP);

        sauth_setusermask(uid, 1, NULL);
```

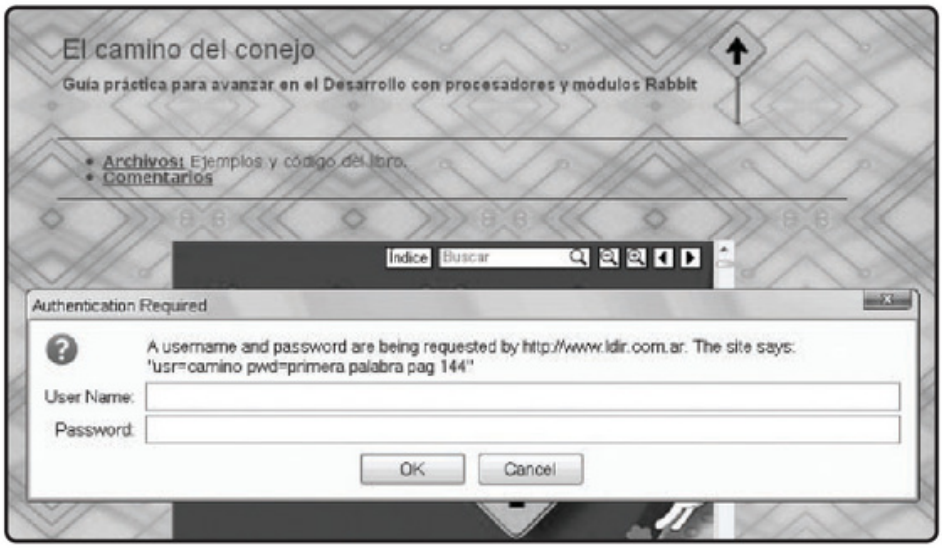


FIGURA 4.
Una página que solicita autenticación. El texto entre comillas es el `realm`.

RabbitWeb

El lenguaje de RabbitWeb, **ZHTML**, se encuentra embebido dentro de **tags**, de modo similar a **PHP** o **SHTML**, y permite realizar páginas dinámicas. El código se aloja en varias líneas dentro del envoltorio **<?z ?>**. Cada variable referida en el script está precedida por los signos **$** o **@**: el primero indica que estamos haciendo referencia al valor que se intenta introducir al enviar la página; en tanto que el segundo señala que nos referimos al valor que tiene la variable en el programa principal. La sintaxis es reducida y similar a C:

- ejecución condicional mediante **if()**.
- bloques iterativos mediante **for(; ;)**.
- llamadas a funciones internas de RabbitWeb.

Por ejemplo, el código que se presenta a continuación hace que el servidor muestre el valor de la variable **mivar**: en rojo si esta posee 1 y en azul si tiene otro valor.

```
<?z if($mivar==1) { ?>
       <FONT COLOR="#FF0000">
<?z } ?>
<?z if($mivar!=1) { ?>
       <FONT COLOR="#0000FF">
<?z } ?>
       <?z print($mivar) ?>
```

SELECTORES Y BOTONES

Existen funciones propias para soportar la generación de selectores y botones, y también podemos marcar la opción corriente.

El siguiente código genera un selector desplegable y una lista de botones.

```
<SELECT NAME="milista">
<?z print_select($milista) ?>
</SELECT><br>
<?z for ($A = 0; $A < count($blista);
$A++){ ?>
       <INPUT TYPE="radio"
NAME="blista" OPTION
       <?z if (selected($blista, $A) )
{ ?>
                  CHECKED
       <?z } ?>
       VALUE="<?z print_opt($blista,
$A) ?>"> <?z print_opt($blista, $A) ?>
<?z } ?>
```

Vemos aquí que la función **count($blista)** determina la cantidad de botones (opciones) en **blista**. Valiéndonos de **for(; ;)** armamos un bloque iterativo. Las variables de estos son internas, y sus nombres son de una sola letra (Figura 5).

FUNCIONES

También existen funciones para determinar si la página está siendo mostrada como resultado de un **GET** (pedido de página) o de un **POST** (envío de

RabbitWeb es un lenguaje de script interpretado por Rabbit al servir una página web

datos), y en este caso, si cometimos un error en la información enviada.

```
<?z if(update()) { ?>
        estoy respondiendo a un POST
        <?z if(error()) { ?>
                hay un error en alguno
de los valores
        <?z } ?>
        <?z if(error($mivar)) { ?>
        hay un error en el valor
de mivar en particular
        <?z } ?>
<?z } ?>
<?z if(!update()) { ?>
        estoy mostrando la página
por un GET
<?z } ?>
```

Los rangos de los valores se especifican al declarar las variables, como veremos más adelante. Estas son solo algunas de las prestaciones; vayamos ahora a las extensiones al lenguaje C.

LAS EXTENSIONES A C

Dynamic C incorpora una serie de extensiones al lenguaje C para soportar la comunicación con el script y los servicios adicionales. Por ejemplo, podemos definir grupos de usuarios en RabbitWeb, cuyos nombres usaremos en las funciones de autenticación vistas anteriormente en este capítulo.

Las variables que vamos a mostrar y/o cambiar vía web se registrarán después de declararlas. De manera opcional, al mismo tiempo o luego de hacerlo, podemos definir los grupos de usuarios y los permisos de acceso asociados a ellos. También podemos asociar una función de evaluación para comparar los valores que se quiere modificar con rangos permitidos.

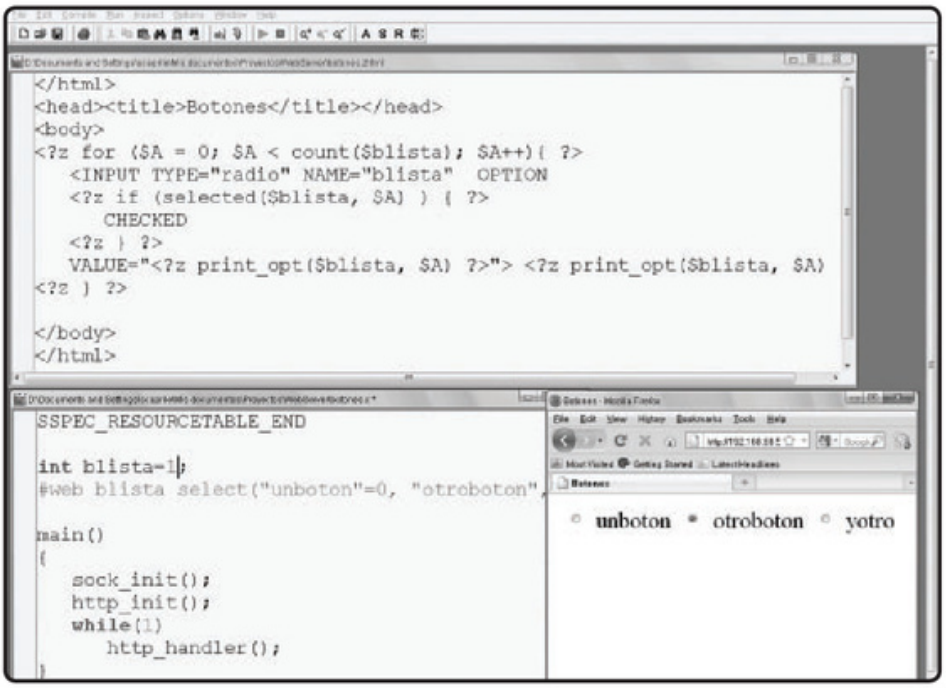


FIGURA 5.
Generación de botones con RabbitWeb. En la esquina inferior derecha, el navegador muestra el resultado.

```
#define USE_RABBITWEB 1
#web_groups GRUPO1, GRUPO2

#web mivar (($mivar >= -5) && ($mivar
<= 5)) auth=basic,digest \
groups=GRUPO1(rw),GRUPO2(ro)
```

Solo los usuarios pertenecientes a GRUPO2 y/o GRUPO1 pueden acceder a **mivar** y únicamente los de este último pueden modificarla. El valor por ingresar debe estar entre -5 y 5.

El signo **$** por delante del nombre hace referencia al valor que se quiere ingresar desde el navegador, en vez del que tiene la variable en el sistema.

Para trabajar con selectores y botones, asignamos los valores posibles en un entero y los textos del selector como trabajamos con un enumerador de C.

AJAX es un esquema que nos permite actualizar páginas web en forma dinámica

Cuando el navegador envía datos, estos son diferentes de los actuales, y cumplen con los requisitos especificados de acceso y rango de validez, podemos ejecutar una función que haga algo con todo eso. Lo llevamos a cabo de la forma con el siguiente código:

```
int milista;
#web milista select( "optuno" = 1,
"optdos", "opttres", "optcuatro" )

void cambio(void)
{
        // tarea a ejecutar si se modifica mivar
}

#web_update mivar cambio
```

UTILIZACIÓN

Al emplear RabbitWeb, no necesitamos declarar variables en la estructura de directorio; solo precisamos el tipo **MIME** para **ZHTML** con su **handler**.

```
SSPEC_MIME_FUNC(".zhtml", "text/html",
zhtml_handler)
```

RABBITWEB

RabbitWeb es introducido por Rabbit en el año 2005. Su intención es minimizar e integrar el trabajo de implementar interfaces de usuario. Con RabbitWeb, el usuario puede ver todo como una unidad, repartiendo la tarea entre un script y un programa principal.

Las variables declaradas mediante **#web** son accedidas dentro del script **ZHTML** en tiempo de ejecución de este y no es necesario incluirlas en el directorio del servidor web ni hacer nada adicional.

SOPORTAR AJAX

AJAX (*Asynchronous JavaScript And XML*) es una tecnología que permite actualizar elementos de una página web sin tener que recargarla. Un programa JavaScript se ejecuta en el navegador web y, periódicamente, solicita al servidor los elementos de interés. Por lo general, se trata de archivos XML que contienen los valores de las variables por actualizar. Como vemos, no se trata de algo relativo al servidor, sino que todo el trabajo está en el navegador; el servidor solo debe ser capaz de actualizar y entregar dichos objetos.

Utilizando RabbitWeb (Figura 6), podemos generar el archivo XML fácilmente mediante un script, empleando **<?z print (@mivar) ?>**. Un ejemplo de esto se ve en el código siguiente.

```
<?xml version='1.0'
encoding='ISO-8859-1'?>
<values>
        <analog>
        <ch> <?z print(@entradaAD)
?> </ch>
        </analog>
</values>
```

SERVIDOR WEB CON RABBIT

A continuación, observamos el código C, que incorpora todo lo visto hasta el momento, en el siguiente orden: configuración de la dirección IP, carga de archivos, tabla de tipos MIME, directorio del servidor, declaración de la variable que usaremos, función que se ejecuta ante un cambio y programa principal.

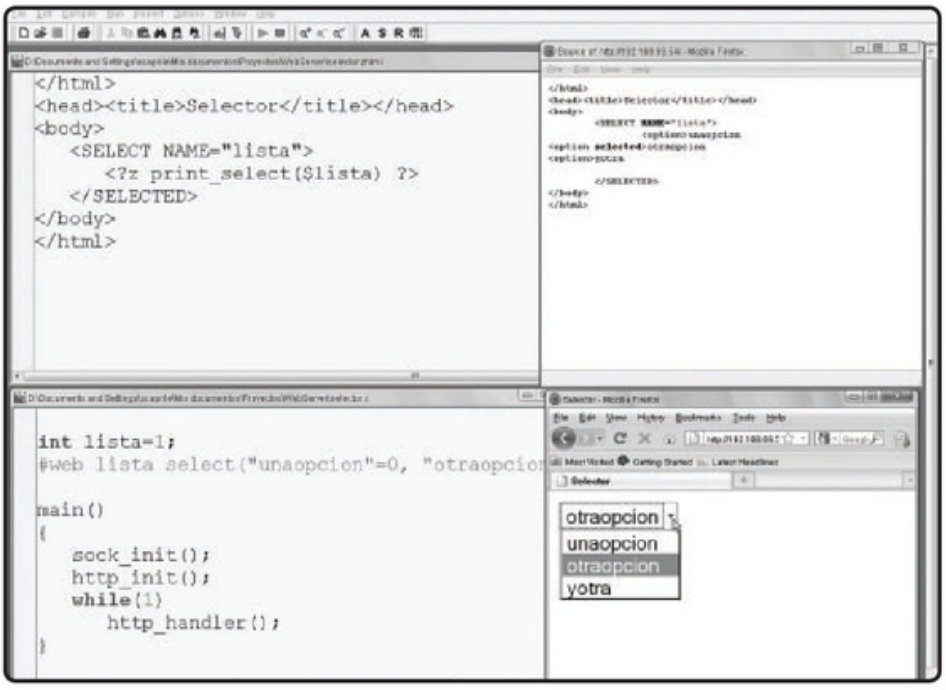


FIGURA 6.
Selector generado con RabbitWeb. A la derecha, el navegador muestra el código fuente generado (arriba) y el resultado (abajo).

```
#define TCPCONFIG        0
#define USE_ETHERNET     1

#define MY_IP_ADDRESS  "192.168.69.54"
#define MY_NETMASK    "255.255.255.0"

#define MY_GATEWAY       "192.168.69.1"

#use "dcrtcp.lib"
#define USE_RABBITWEB 1
#use "http.lib"

#ximport "index.zhtml" index_zhtml
#ximport "users.gif"     users_gif
SSPEC_MIMETABLE_START
        SSPEC_MIME_FUNC(".zhtml", "text/
html", zhtml_handler),
        SSPEC_MIME(".html", "text/html"),
        SSPEC_MIME(".gif", "image/gif"),
SSPEC_MIMETABLE_END

SSPEC_RESOURCETABLE_START
        SSPEC_RESOURCE_XMEMFILE("/",
index_zhtml),
        SSPEC_RESOURCE_XMEMFILE("/index.
zhtml", index_html),
        SSPEC_RESOURCE_XMEMFILE("/users.
gif", users_gif),
SSPEC_RESOURCETABLE_END

int mivar;
#web mivar ($mivar > mivar)

void cambio(void)
{
        printf("Cambio de valor: %d\n",
mivar);
}

#web_update mivar cambio

main()
{
        sock_init();
        http_init();
        tcp_reserveport(80);
        while(1)
                http_handler();
}
```

Luego, vemos el código HTML con el script embebido. Mostramos una imagen, la variable y un botón para cambiar el valor. Si el valor introducido cumple la condición que se le pide (mayor que el anterior), al aceptarlo, se mostrará **OK** en azul.

```
</html>
<head><title>Mi web server</title></
head>
<body>
<center><IMG SRC="users.gif" ></center>
<hr>
<H1>Felicitaciones, ha iniciado su pri-
mer web server</H1>
<hr>
<form ACTION="index.zhtml"
METHOD="POST">
<?z if(updating()) { ?>
        <?z if(!error()) { ?>
```

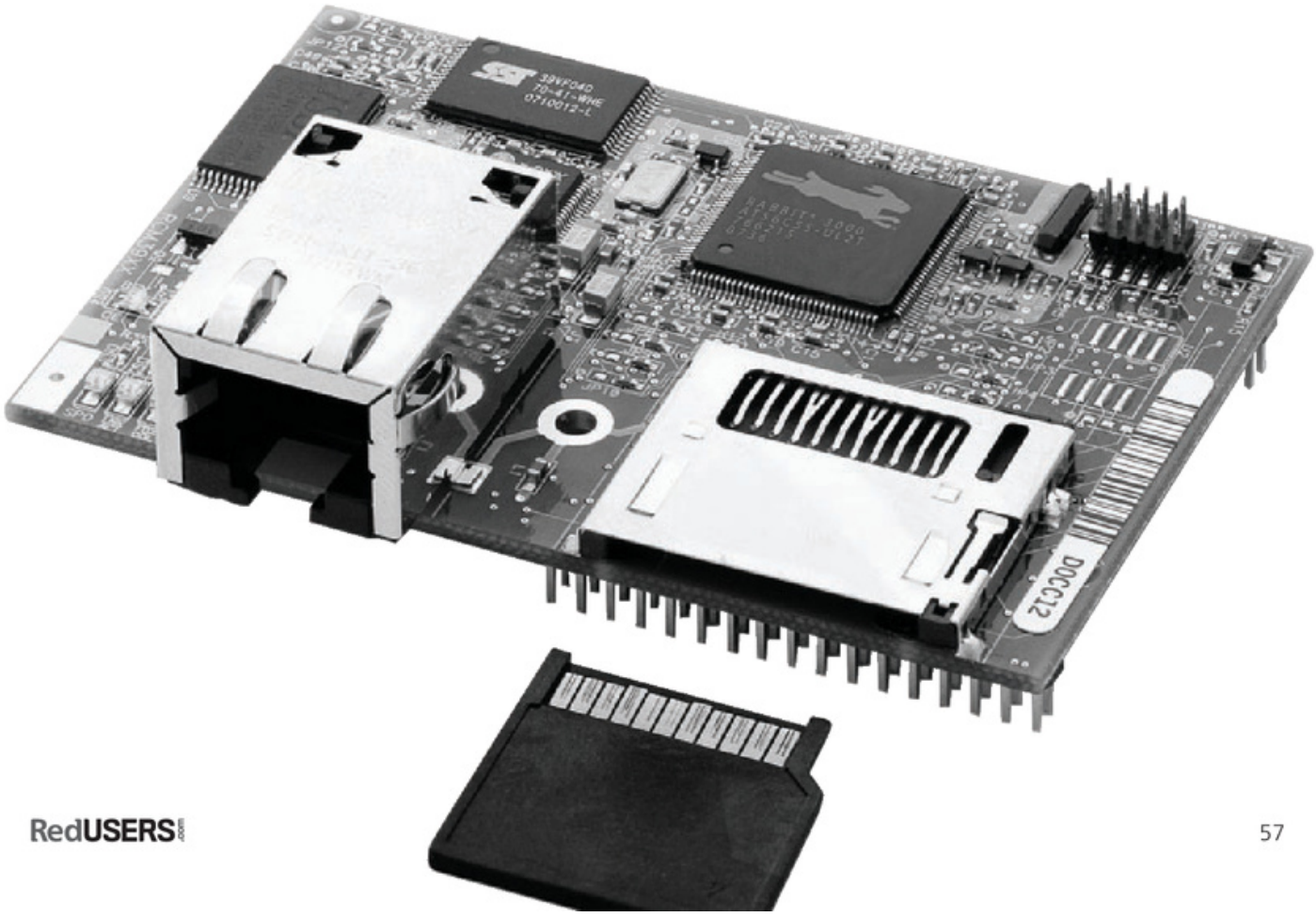
```
                <H2><FONT
COLOR="#0000FF">OK</FONT></H2><hr>
        <?z } ?>
<?z } ?>
<table>
<tr><td>Variable:
<td><input TYPE="TEXT" NAME="mivar"
SIZE=5 VALUE=<?z print(@mivar) ?>>
</table>
<input TYPE="SUBMIT" VALUE="Cambiar"></
form>
</body>
</html>
```

Esto es todo lo necesario para realizar un servidor web con Rabbit: el código del programa y la página en sí, sin conversiones ni pasos intermedios. En la próxima sección lo veremos en acción.

SERVIDOR WEB CON RABBIT

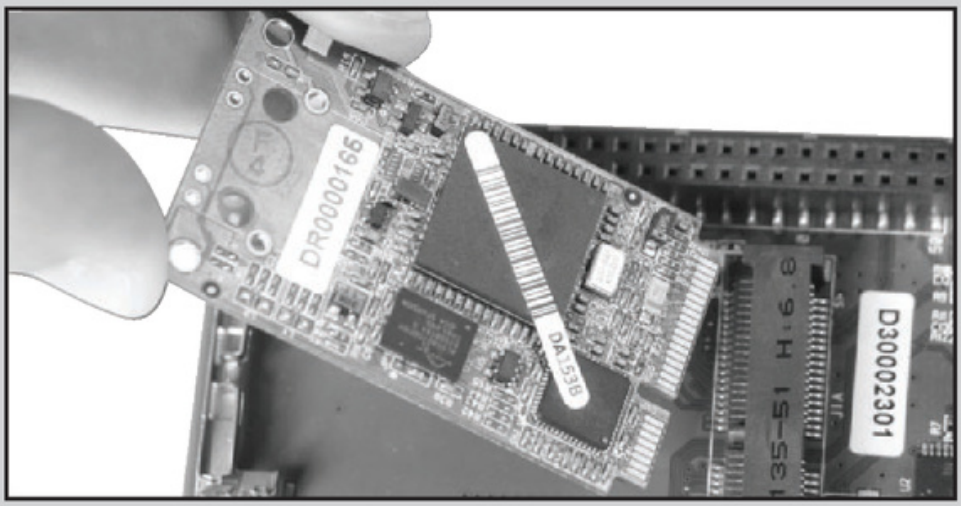
Llegó el momento de llevar a la práctica los conceptos de networking –en general– que hemos aprendido en el Capítulo 1 y los de Rabbit –en particular– vistos en este capítulo.

Para conectarnos con el módulo Rabbit, podemos usar un switch. Si disponemos de una red o podemos consultar a alguien, esta es la opción más aconsejable. Si no es así, en estas páginas sugerimos un cable cruzado para unir la PC y Rabbit directamente, de modo de poder realizar la práctica sencillamente aun para quienes no tienen conocimientos avanzados sobre networking.



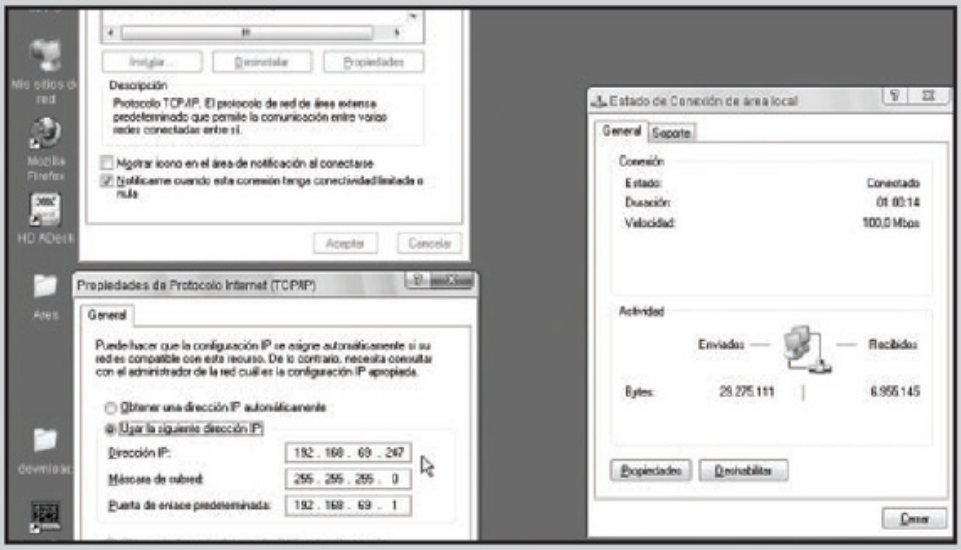
PASO A PASO /1
Servidor web con Rabbit

1



Mediante un patch-cord cruzado, una el conector de red de la PC (RJ-45) con el del módulo Rabbit. Si el módulo es un minicore, como el RCM5700, insértelo cuidadosamente en la placa de desarrollo.

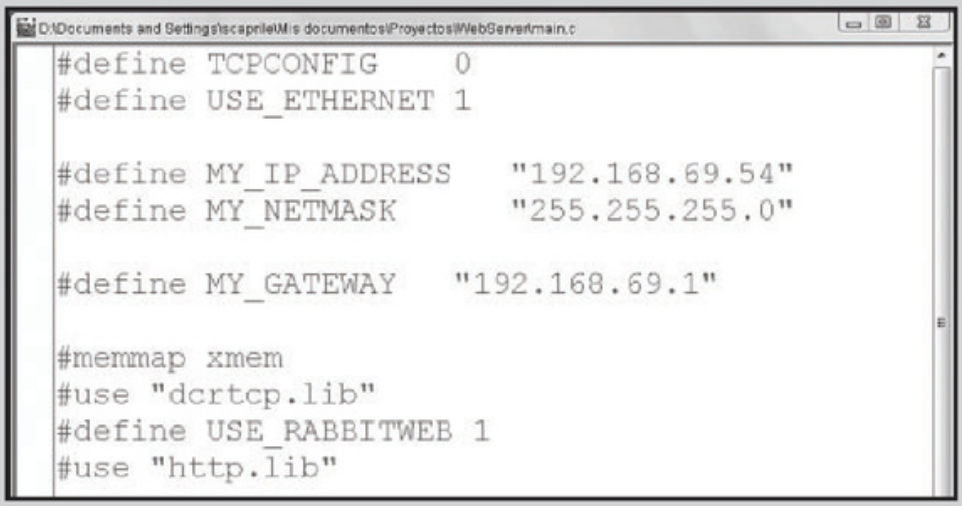
2



Debe configurar la dirección IP de la PC dentro de la misma red que Rabbit. Utilizará la IP 192.168.69.247, con máscara 255.255.255.0. Seleccione **`Conexiones de red`** y, luego, el port Ethernet correspondiente en el **`Panel de control`**.

PASO A PASO /1 (cont.)

3



La dirección IP de Rabbit está al principio del código del programa. Usará 192.168.69.54, máscara 255.255.255.0. Al hacer correr el programa, el módulo Rabbit estará en la red y podrá conectarse a él.

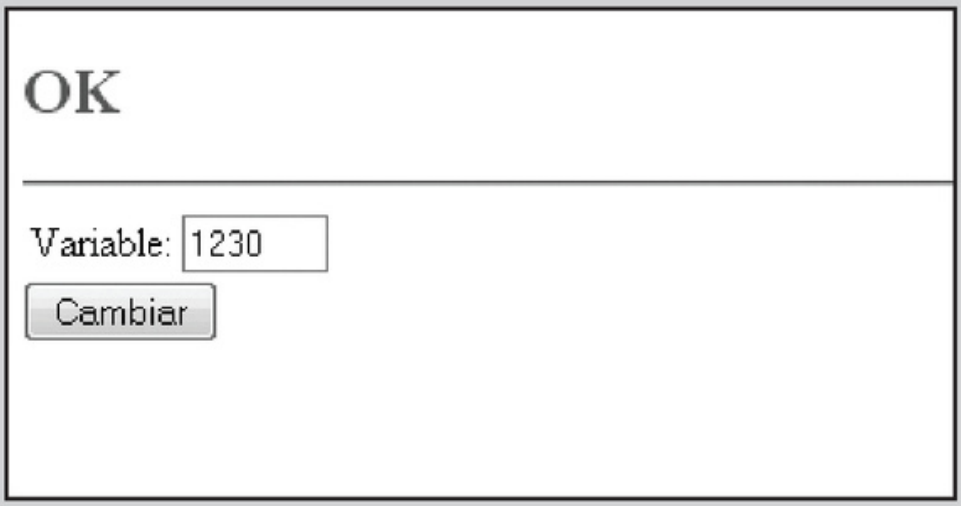
4



Abra el navegador web utilizado normalmente y coloque la URL de Rabbit: http://192.168.1.54/. Observará la página servida por él; de no ser así, tendrá que revisar los pasos anteriores.

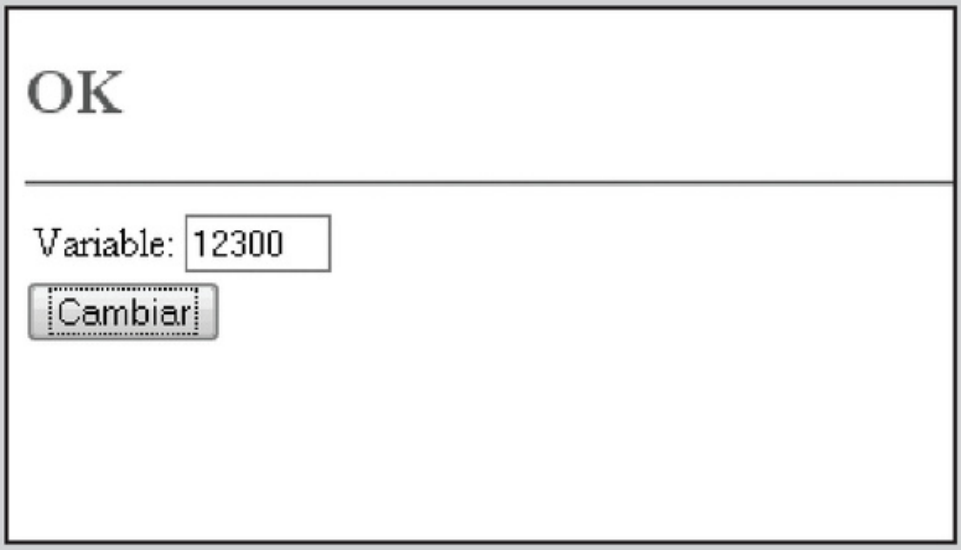
PASO A PASO /1 (cont.)

5



Modifique el valor mostrado, ingresando uno mayor, y presione el botón **`Cambiar`**.
El valor es aceptado y reciba la página nuevamente, con el texto **`OK`** en azul.
Visualice, una indicación en la ventana **`"stdio"`** de Dynamic C.

6



La acción anterior provoca que Rabbit ejecute una función al recibir un valor aceptable. Ponga un breakpoint en ella y repita la acción anterior.
Verá que la ejecución se detiene en esa función, y el navegador queda esperando.

WiFi con Rabbit

Wi-Fi es un nombre que engloba los estándares **IEEE 802.11**. Mediante esta tecnología, dos dispositivos pueden comunicarse entre sí **ad-hoc** o a través de un **access point** (punto de acceso) que provee la interconexión, en el modo infraestructura.
Una red **802.11** tiene un identificador **SSID** (*Service Set Identifier*), que se anuncia periódicamente, permitiendo que otros dispositivos puedan asociarse a la red. Para que los datos no sean vistos por aquellos que no deben hacerlo, se suele **encriptar** y autenticar a quienes quieren acceder a la red. Los métodos más utilizados son:

- **WPA/TKIP**: acceso protegido a WiFi usando el protocolo de integridad temporaria de claves.
- **WPA2/CCMP**: acceso mediante un protocolo basado en encriptación **AES**, el estándar avanzado.

Los métodos de encriptación dependen de una clave que puede ser **PSK** (clave pre-compartida) o adquirida mediante el esquema **EAP** (protocolo de autenticación extensible). El primer caso se conoce como esquema **residencial** y, el último, como **enterprise**. Por lo general, los equipos inalámbricos no utilizan una dirección IP fija, sino que la adquieren al momento de ingresar a la red de un servidor **DHCP**, que además les informa sobre la dirección del router y el o los servidores DNS. Es común que este servidor esté en el mismo access point.

El **Rabbit 5000** incorpora soporte para WiFi dentro del chip, incluyendo **WPA2** y autenticación de tipo enterprise. La forma de trabajar con un módulo WiFi es exactamente igual a lo ya visto para los módulos Ethernet (**Figura 7**).

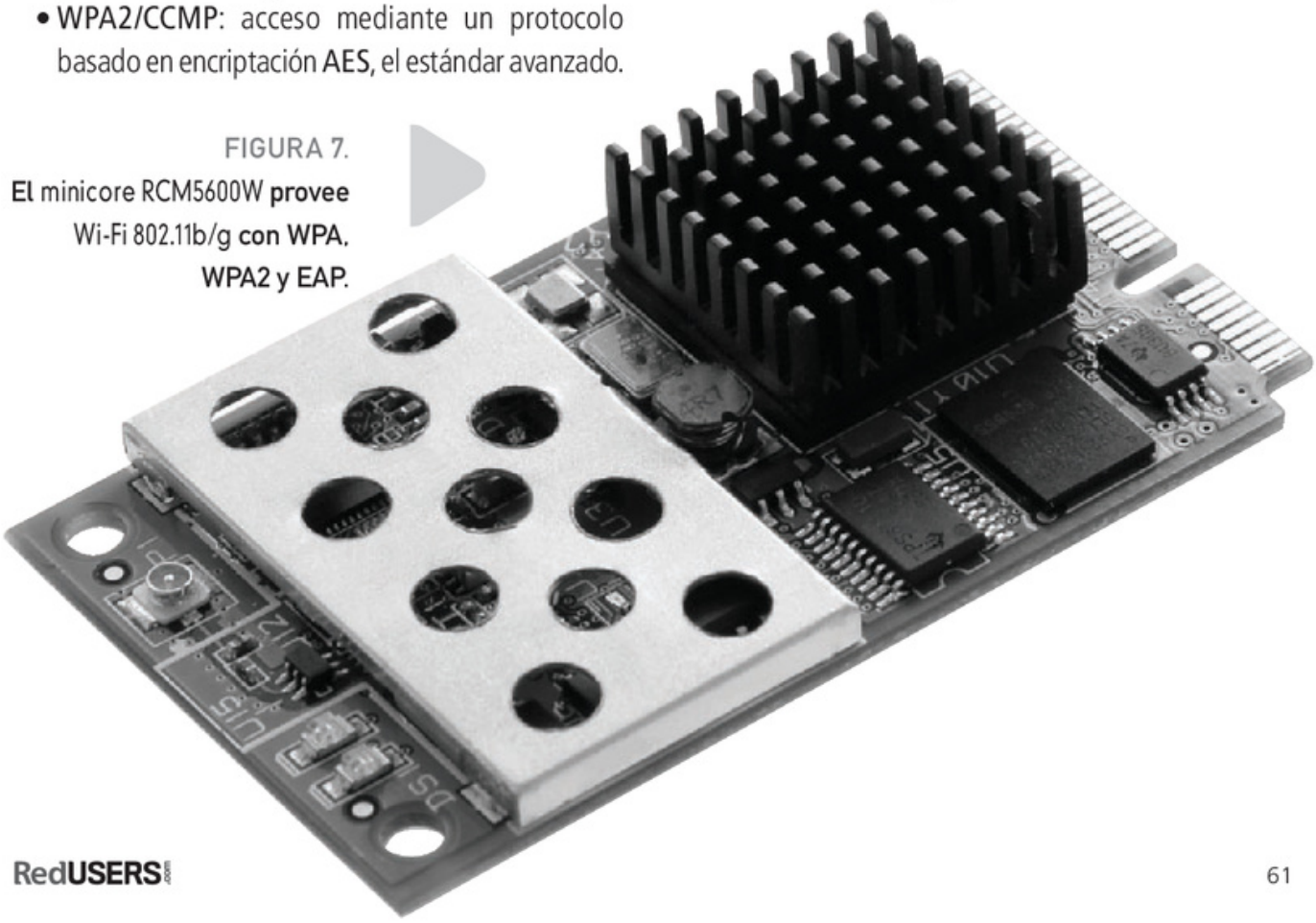


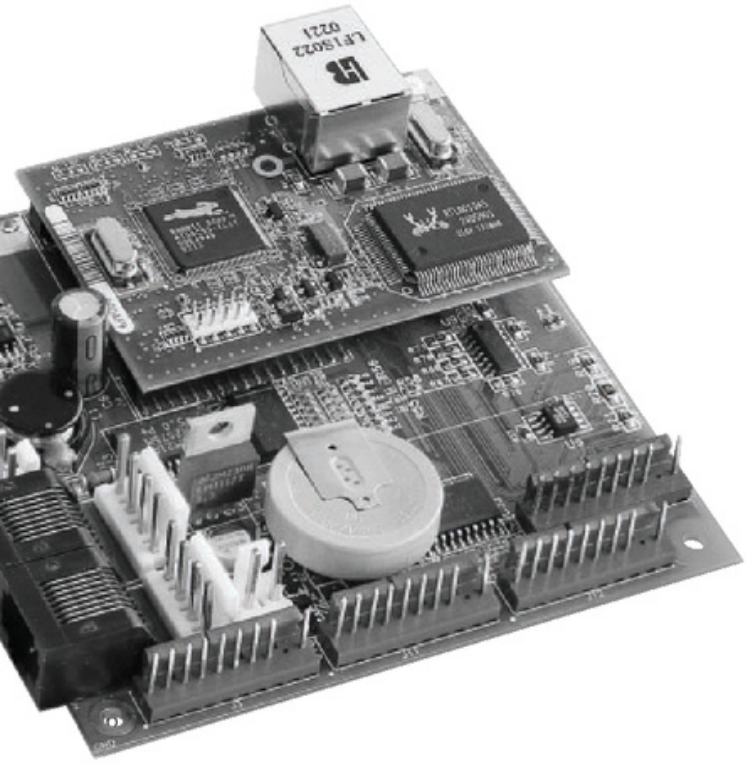
FIGURA 7.
El minicore RCM5600W **provee** Wi-Fi 802.11b/g **con WPA, WPA2 y EAP.**

A continuación, examinaremos las pequeñas diferencias que existen. En general, debemos especificar el uso de la interfaz Wi-Fi (y si empleamos DHCP), de la manera que se muestra en el código siguiente:

```
#define TCPCONFIG 5
// si utilizamos Wi-Fi

#define USE_WIFI 1
      #define IFC_WIFI_SSID "mired"
```

CONEXIÓN Y ENCRIPTACIÓN

Podemos conectarnos sin usar encriptación ni restringir el acceso, por ejemplo, en una red de desarrollo. De lo contrario, tanto para **WPA/TKIP** como para **WPA2/CCMP**, necesitamos compilar el código de soporte de **WPA** y especificar el tipo de encriptación. En el último caso, además, especificamos la necesidad de compilar el código de soporte de **AES**.



```
// red abierta
      #define IFC_WIFI_ENCRYPTION
IFPARAM_WIFI_ENCR_NONE

// red WPA
#define WIFI_USE_WPA
        #define IFC_WIFI_ENCRYPTION
IFPARAM_WIFI_ENCR_TKIP
#define WIFI_VERBOSE_PASSPHRASE

// red WPA2
#define WIFI_USE_WPA
#define WIFI_AES_ENABLED
        #define IFC_WIFI_ENCRYPTION
IFPARAM_WIFI_ENCR_CCMP
// clave
        #define IFC_WIFI_WPA_PSK_
        PASSPHRASE "miclave"
        #define IFC_WIFI_WPA_PSK_
        HEXSTR \
  "25BB6E6DE7937997D978A17932DD9ABC
  4C4F99E84FE95F8F5989964D8AC97004"
```

Como vimos, la clave PSK puede colocarse en ASCII. Esto provoca una demora importante (decenas de segundos) para la generación, por lo que es conveniente hacerlo solo la primera vez y, luego, cambiarla por el ingreso directamente en hexadecimal.

El Rabbit 5000 incorpora soporte para Wi-Fi dentro del chip, WPA2 y autenticación enterprise

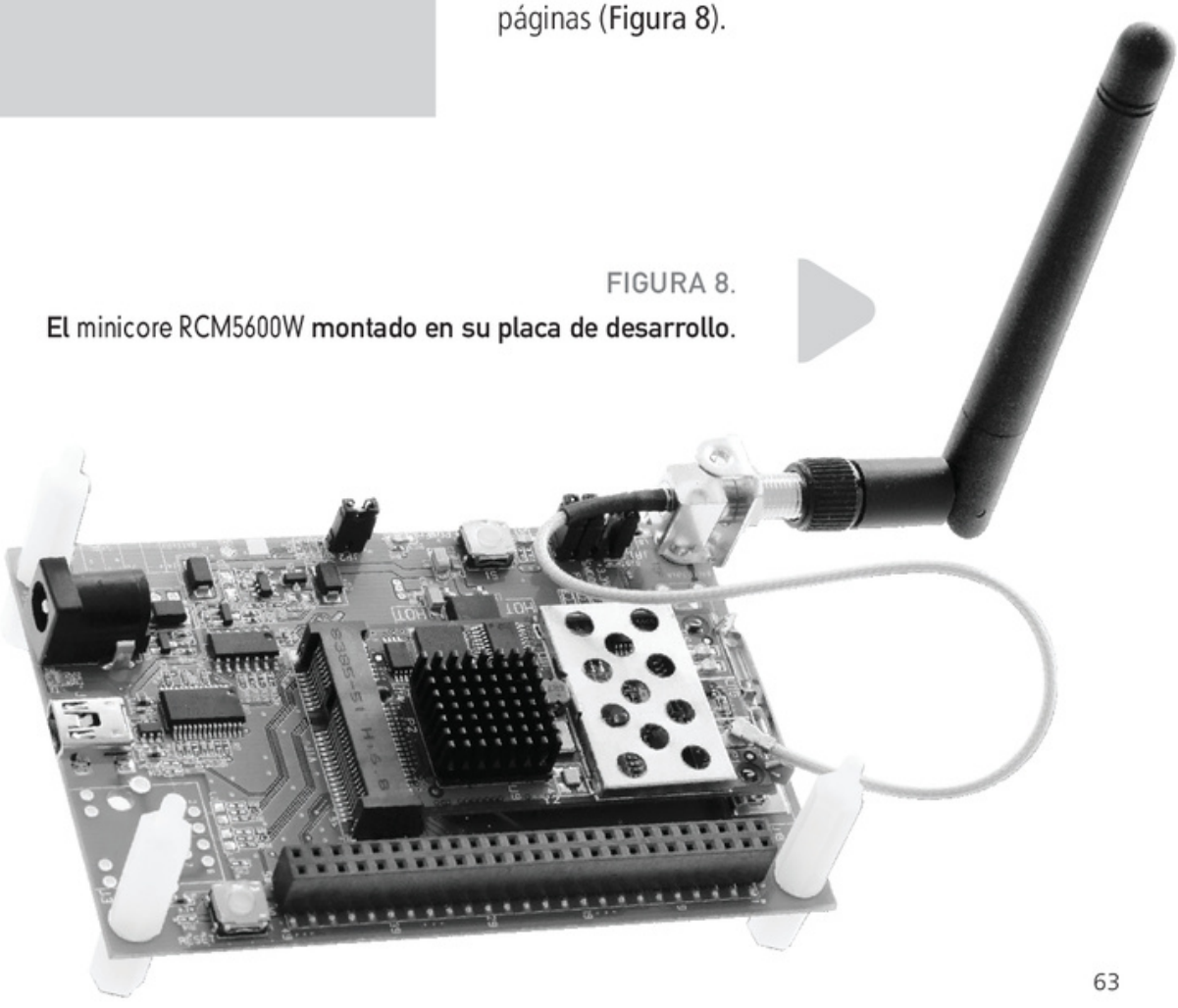
INICIO DEL SERVIDOR WEB

En el programa principal, luego de la llamada a **sock_init()**, debemos revisar el estado de la interfaz en espera de la conexión con el access point y la posterior asignación de la dirección mediante DHCP. Esto se realiza, por ejemplo, a través de un lazo de espera, a menos que debamos efectuar otra tarea.

```
#define TCPCONFIG 5
#define USE_WIFI                    1
     #define IFC_WIFI_SSID     "mired"
     #define IFC_WIFI_ENCRYPTION
IFPARAM_WIFI_ENCR_NONE

// directorio, MIME, etc

main()
{
        sock_init();

while(ifpending(IF_WIFI0)==IF_COMING_UP)
                tcp_tick(NULL);
        http_init();
        while(1){
                http_handler();
        }
}
```

Como vemos, el código del servidor web, modificado para un módulo Wi-Fi, difiere del utilizado en Ethernet solo en lo que hemos visto en estas últimas páginas (**Figura 8**).

FIGURA 8.
El minicore RCM5600W **montado en su placa de desarrollo.**



Multitarea cooperativo

Hasta ahora, nos enfocamos en realizar solamente una tarea. Pero si debemos ocuparnos de más de una operación al mismo tiempo, resultaría engorroso escribir el código que manejen ellas, porque sería tan complejo que solo lo entendería quien lo hubiera hecho (y a veces ni él mismo), y resultaría muy difícil de mantener y modificar. Aún más cuando la cantidad de tareas se incrementa.

MULTITAREA

Entendemos por **multitarea** (*multitasking*) a la aparente ejecución simultánea de varias acciones. Decimos que es "aparente" porque sabemos que un procesador solo puede ejecutar una instrucción (tarea) a la vez. Para que varias se ejecuten realmente al mismo tiempo, precisamos varias unidades de ejecución, en uno o varios procesadores. Cada cambio de tarea (*task switching*) requiere un intercambio del contexto de operación (*context switching*). La forma en que esto se realiza define al tipo de entorno. Los dos más comunes son el **preemptive**, en el que un administrador interrumpe y conmuta a la fuerza, y el cooperativo, que veremos a continuación.

Multitarea cooperativo

En este esquema, cada tarea es responsable de ceder el procesador a las demás cuando no tiene nada útil que hacer. El hecho de tener que estar esperando a que ocurra un evento no está considerado algo útil, por lo que el esquema se basa en la detección de los ciclos de espera y en la cesión de control en ellos. Entre las estructuras más utilizadas, encontramos:

- **Handlers** (manejadores), rutinas que atienden un evento y retornan inmediatamente.
- **Máquinas de estados finitos**, llamadas solo máquinas de estados, son rutinas que mantienen un estado interno, que cambia en función de determinados eventos y define la operación por realizar.

Máquinas de estados finitos

Durante los ciclos de espera, existe un estado que verifica la ocurrencia del evento esperado y cede el control. La máquina solo saldrá de ese estado cuando dicho evento se produzca (**Figura 9**).

Entendemos por multitarea a la aparente ejecución simultánea de varias acciones

MULTITAREA DE TIPO PREEMPTIVE

El sistema elige cuál de las tareas se ejecuta y el orden en el que las realiza. Se interrumpe la tarea en ejecución para ceder el control a otra. Las tareas no pueden saber en qué momento se las va a interrumpir, y el sistema debe guardar y restaurar el estado de ejecución de cada una.

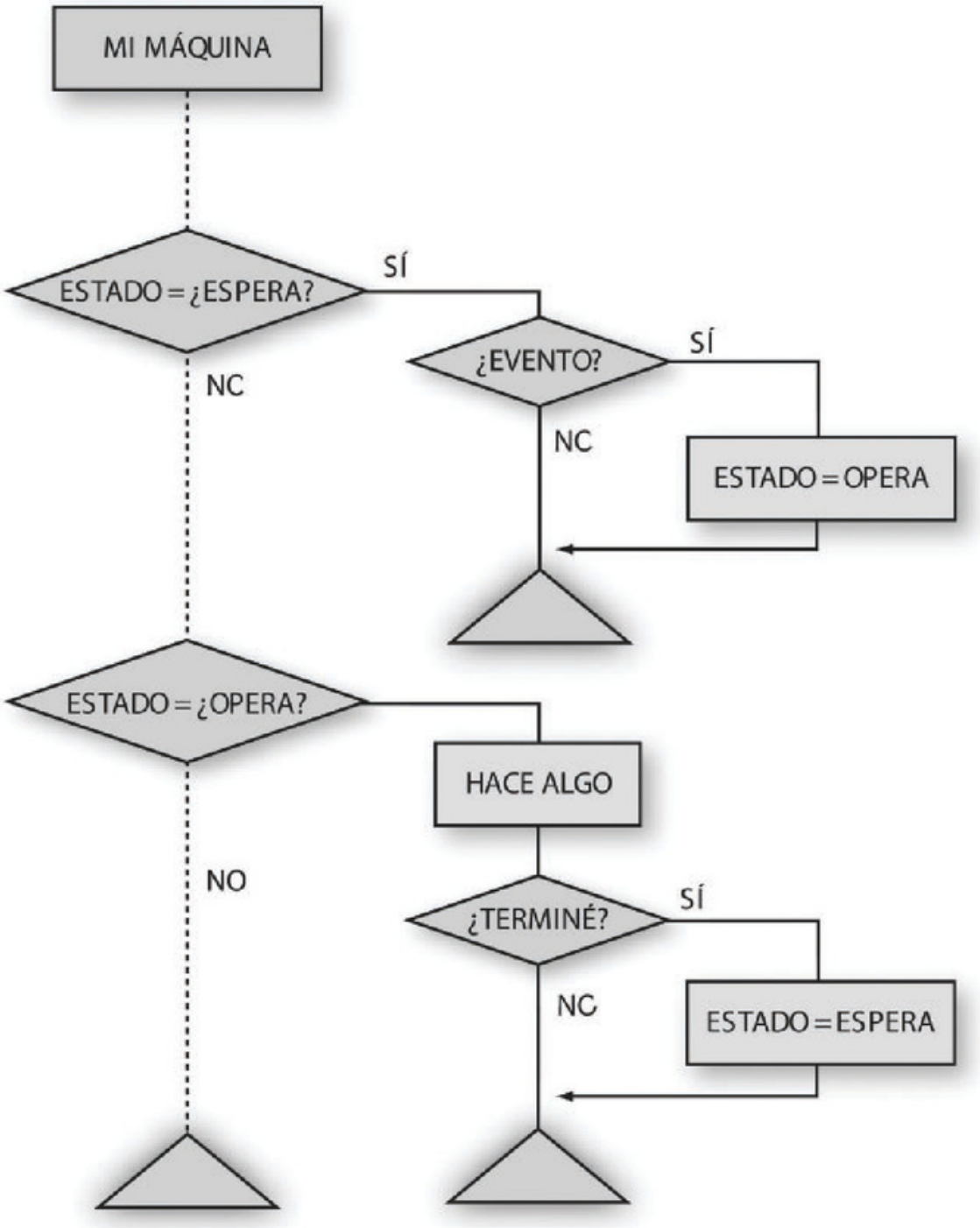


FIGURA 9. **Diagrama de flujo de una máquina de estados.**

El cambio de contexto es simple y solo hay un elemento que debe salvarse: el estado

El cambio de contexto es simple y solo hay un elemento que debe salvarse: el estado. Cada tarea cede el control voluntariamente en los estados de espera, y sabe que las variables compartidas con otras tareas (excepto interrupciones) solo podrán ser alteradas durante los tiempos en que ella lo permite. En caso de no existir ciclos de espera, el código tomará la forma de **handlers**.

Las tareas serán llamadas de a una por un loop principal (**Figura 10**). La única manera de que todas las tareas se ejecuten es que todas cedan el control periódicamente, como vemos en el código de la columna contigua.

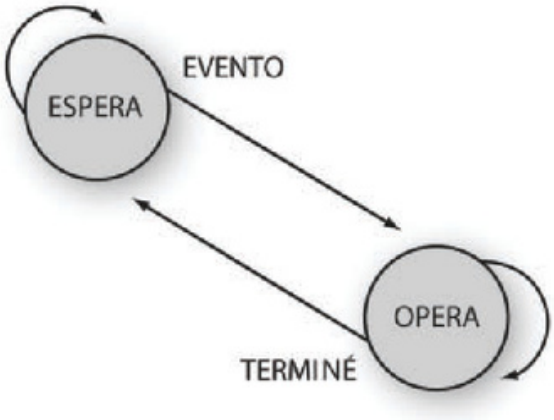


FIGURA 10. **Diagrama de estados de la máquina ejemplificada. Vemos cómo se indican los cambios.**

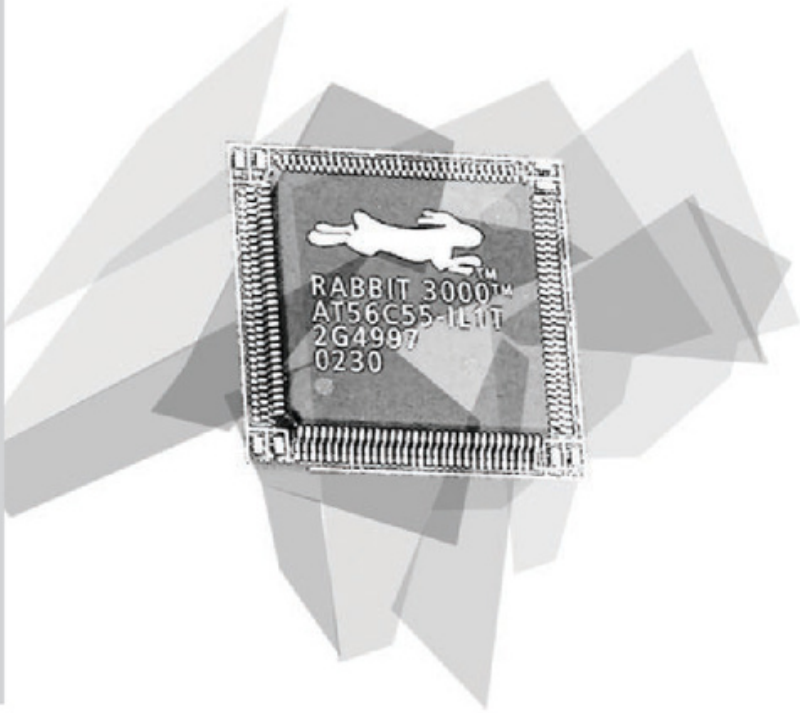
```
void mimaquina(void)
{
enum { ESPERA=0, OPERA};
static int estado=ESPERA;

        switch(estado){
        case ESPERA:
                if(evento)
                        estado=OPERA;
                break;
        case OPERA:
                // hace algo
                if(termine)
                        estado=ESPERA;
                break;
        }
}

void mihandler(void)
{
                if(evento) {
                        // hace algo
                }
}

main()
{
        while(1){
                miamaquina();
                mihandler();
        }
}
```

REUTILIZACIÓN DE CÓDIGO

En caso de realizar la misma operación sobre diferentes elementos, en vez de escribir código muy parecido varias veces, podemos reutilizarlo. Entonces, pasamos parámetros a la máquina (el estado y los elementos), por ejemplo, dentro de una estructura.

```
typedef struct {
        int estado;
        void *elemento;
} mistruct;

void mimaquina(mistruct *ptr)
{
        switch(ptr->estado){
                ...
                // hace algo con
*ptr->elemento
        }
}

main()
{
mistruct maq1,maq2;

        // incializa estructuras
        while(1){
                mimaquina(&maq1);
                mimaquina(&maq2);
        }
}
```

Por el contrario, si varias tareas operan sobre el mismo elemento, tendremos que implementar un esquema que permita tener uso compartido de él. Una alternativa común es usar un **semáforo**.



SEMÁFORO

Permite repartir un recurso entre varias tareas. Puede ser representado, por ejemplo, por un bit. Todas las tareas que deseen utilizar el recurso deberán esperar hasta que el semáforo esté libre (bit=0). La tarea que obtiene primero el semáforo (setea el bit), utiliza el recurso.

Multitarea cooperativo en Rabbit

Dynamic C incorpora una serie de extensiones al lenguaje C que simplifican el proceso de generación y el mantenimiento de código. Si bien es posible utilizar las técnicas de programación comunes –y, de hecho, las funciones del stack TCP/IP de Rabbit lo hacen–, y nada obliga a utilizar estas extensiones propietarias, seguramente el lector las encontrará interesantes.

COSTATES (CO-SENTENCIAS)

Mediante la palabra clave **costate** identificamos un bloque que se asocia con una tarea. Internamente, el sistema maneja cada **costate** como si fuera una máquina de estados.

Cada **costate** tiene su propio puntero de programa (*statement pointer*) para determinar qué ítem en la lista debe ejecutarse cuando le corresponde. La cesión de ejecución ocurre en los puntos donde existen sentencias de control a tal fin, las cuales veremos más adelante. El esquema se ve en el código siguiente.

```
while(1){
        costate {
                // tarea 1
                ...
                sentencia de control
                // cede el control a
otros costates
                ...
        }
        costate {
                // tarea 2
                ...
                sentencia de control
                // cede el control a otros
costates
                ...
        }
  }
```

COFUNCTIONS (CO-FUNCIONES)

Se trata de funciones que pueden tener dentro sentencias de control. Así como en un programa en C tomamos una porción de código y la convertimos en una función, en Dynamic C podemos hacer lo mismo con un **costate** y convertir una parte en una **cofunction**. Es posible pasarles argumentos, y devuelven un resultado.

```
cofunc int micof(int miparam)
{
        ...
        sentencia de control
        // cede el control a otros costates
        ...
        return(miresult);
}
```

Las **cofunctions** se llaman desde un **costate** mediante una sentencia de control especial denominada **waitfordone**.

```
waitfordone {micof}
        // puede abreviarse wfd micof;
```

SENTENCIAS DE CONTROL

El esquema se basa en el uso de sentencias de control para ceder el procesador a otras tareas.

- **waitfor(expr)**: cede el control hasta tanto expr sea verdadero (1). Al retomar la ejecución, se reevalúa la expresión.
- **yield**: cede el control. Al retomarse la ejecución, se realiza desde la sentencia siguiente.

Por ejemplo, el código de la máquina de estados, explicada anteriormente, se muestra en el código:

```
cofunc void mimaquina(void)
{
        waitfor(evento);
        do {
                //hace algo
                yield;
        } while (!termine);
}
```

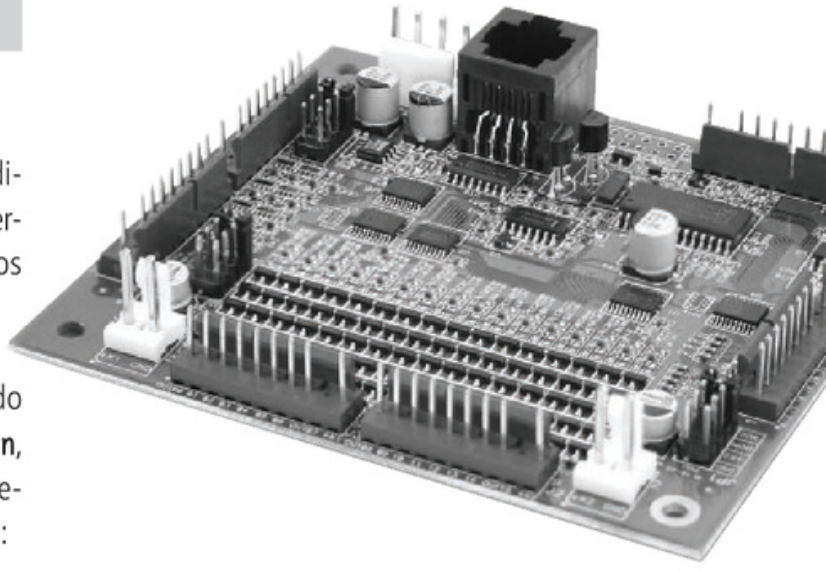
REUTILIZACIÓN DE CÓDIGO I

En este caso también podemos reutilizar el código, pero como los "estados" son manejados internamente por Dynamic C, disponemos de algunos recursos adicionales.

Uno de estos es el equivalente a pasar el estado como parámetro en una **indexed cofunction**, que se declara de manera similar a un array. El segundo ejemplo se muestra en el siguiente código:

```
cofunc void mimaquina[2](void *elemento)
{
        // hacer algo con *elemento y ceder
el control con waitfor y yield
}

main()
{
        while(1){
                mimaquina[0]
(&elemento1);
                mimaquina[1]
(&elemento2);
        }
}
```

El análogo al uso de semáforos son las **single user cofunctions**, que se declaran utilizando **scofunc** en vez de **cofunc**. En este caso, existen dos funciones adicionales: una se emplea para inicializar el esquema interno de semáforos, y la otra se llama al inicio del loop.



Multiple **choice**

1 ¿Quién es el encargado de servir los elementos de una página web?

a- El servidor FTP.
b- El cliente POP.
c- El cliente SMTP.
d- El servidor HTTP.

2 ¿Cuál de las siguientes formas de generar páginas dinámicas permite que el servidor identifique tags en el SHMTL?

a- SSI.
b- CGI.
c- FTP.
d- SMTP.

3 ¿Cuál de las siguientes formas de generar páginas dinámicas asigna una URL para que se ejecute una función?

a- SSI.
b- CGI.
c- FTP.
d- AJAX.

4 ¿Cuál de las siguientes opciones es una tarea que abre un socket TCP en el puerto 80 y espera una conexión?

a- El servidor FTP.
b- El cliente POP.
c- El cliente SMTP.
d- El servidor HTTP.

5 ¿En qué año fue introducido RabbitWeb?

a- 1975.
b- 1985.
c- 1995.
d- 2005.

6 ¿Cómo se llama la tecnología que permite actualizar elementos de una página web sin tener que recargarla?

a- SSI.
b- CGI.
c- FTP.
d- AJAX.

Respuestas: 1-d, 2-a, 3-b, 4-d, 5-d, 6-d.

Capítulo 3

Networking con PIC



Conoceremos las opciones
para presentar la información
al usuario final de nuestra aplicación.

RedUSERS.com

Introducción a networking

Para comenzar a trabajar, necesitamos tener instalado el programa **MPLAB**, el compilador C18 y Microchip TCP/IP Stack. Vamos a estudiar Microchip TCP/IP Stack. Siempre es recomendable consultar la Web del fabricante para descargar la última versión: **www.microchip.com/tcpip**.

Para Microchip TCP/IP Stack, basta con ejecutar el programa **Microchip Libraries**. En verdad, al hacerlo estaremos instalando una serie de librerías que Microchip provee para el manejo de USB, **mTouch** (sensor capacitivo), pantallas gráficas y Ethernet. Todas son libres, siempre y cuando se utilicen en microcontroladores de la misma firma. Microchip TCP/IP Stack está formado por un conjunto de capas que dan acceso a distintos protocolos: ARP, IP, ICMP, UDP, TCP, DHCP, SNMP, HTTP, FTP y TFTP (**Figura 1**).

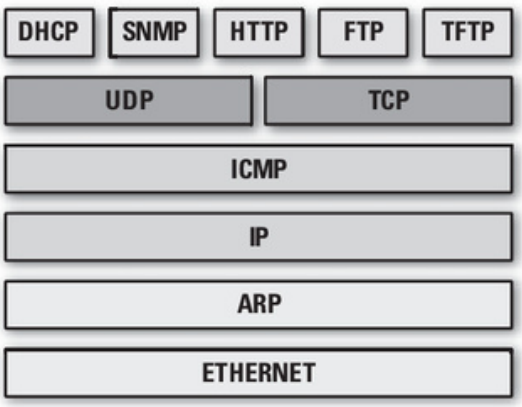


FIGURA 1. **El stack está formado por varia capas, donde cada una brinda servicios a la superior.**

Está programado en C, requiere aproximadamente **32 K** de memoria de programa y es portable a través de las distintas familias de Microchip. Esto quiere decir que funciona en micros de 8, 16 y 32 bits. Además, tiene soporte para encriptación de datos, transmisión de audio y conexión Wi-Fi.

Junto con TCP/IP Stack, se instalan dos aplicaciones sumamente útiles al momento de desarrollar una interfaz Ethernet con PIC: Microchip **TCP/IP Configuration Wizard** y Microchip **Web Preview**. La primera nos permite modificar los parámetros del stack en forma gráfica (sin tener que programar en C), mientras que con la segunda podemos visualizar la página web antes de cargarla en la memoria del PIC. Esta página es nuestra interfaz con el usuario final, por lo que es muy importante darle el aspecto adecuado: simple, liviana y con gran preponderancia de las variables por monitorear/controlar.

MICROCHIP TCP/IP CONFIGURATION WIZARD

Podemos darnos una buena idea sobre las posibilidades del stack si estudiamos en detalle el Microchip TCP/IP Configuration Wizard. Si bien esta aplicación nos permite modificar parámetros de la red –como dirección IP, máscara de enlace y default gateway– y seleccionar los protocolos que vamos a utilizar en el sistema, no da la posibilidad de que el usuario agregue su propio código. Al momento de implementar un sistema, necesariamente debemos programar las funciones en C con el MPLAB o cualquier otro entorno de desarrollo.

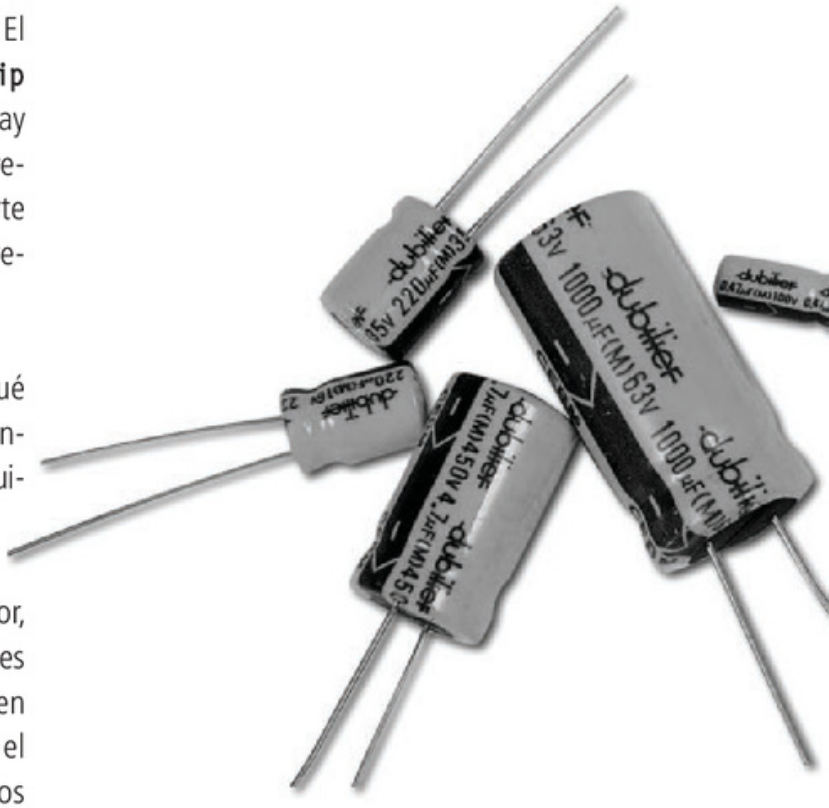
Lo primero que debemos hacer para empezar a trabajar con el Microchip TCP/IP Configuration Wizard

Microchip TCP/IP Stack está formado por un conjunto de capas que dan acceso a distintos protocolos

es seleccionar la carpeta desde la cual vamos a operar, es decir, donde está instalado el stack. El proyecto básico se encuentra en **C:\Microchip Solutions library\TCPIP Demo App**. Si bien hay varios ejemplos realizados con el stack, es conveniente empezar con este. Hay otros con soporte para Wi-Fi y librerías de encriptación SSL, que veremos, más adelante, en el Capítulo 5.

Lo que estamos haciendo es decirle al asistente qué archivos vamos a configurar. Hay que tener presente que el TCP/IP Configuration Wizard agrega o quita funciones del stack comentando código.

Sabemos que el C18, como cualquier compilador, no compila el código comentado ni las funciones asociadas, para ahorrar memoria de programa en el PIC. Por lo tanto, solo debemos seleccionar en el asistente los servicios que realmente necesitamos del stack, como la capacidad para enviar e-mails o servidor seguro. Es importante tener presente que la memoria de programa del PIC18F97J60 se ocupa en un 55% con las opciones básicas de configuración que veremos más adelante; es así que nos queda un 45% para el código.



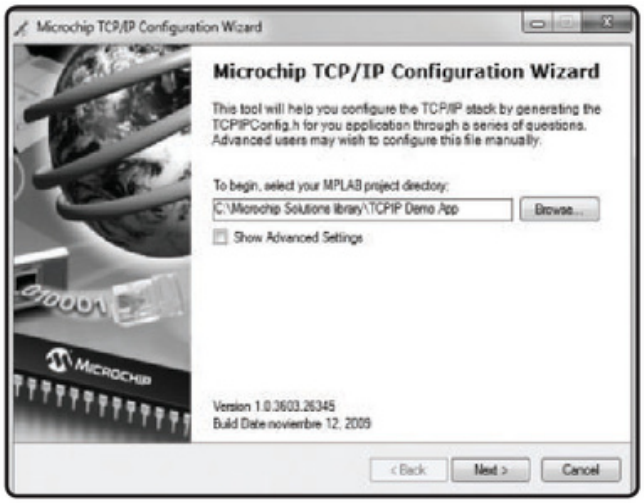
TCP/IP CONFIGURATION WIZARD

Es una aplicación gráfica que modifica los parámetros del stack. Para realizar telemetría y control a través de Ethernet, interactuamos con el stack mediante funciones callback programadas en C, que hacen de nexo entre el firmware del microcontrolador y la página web que ve el usuario final.

Agregar servicios al stack

En esta instancia, vamos a agregar el servicio de **servidor web**, que permite acceder a una página por medio de una dirección IP; y el de **cliente de e-mail**, mediante el cual podremos enviar un mensaje automáticamente ante un determinado evento (**Figura 2**). Esto es útil, por ejemplo, si queremos alertar sobre el aumento de temperatura por encima de un umbral preestablecido. Muchas veces, en aplicaciones industriales es necesario monitorear las revoluciones de un motor; de esta forma, podremos hacerlo con el servidor web y mandar las alertas correspondientes por e-mail.

Como mencionamos anteriormente, cada servicio que deshabilitamos por medio del asistente es comentado en el stack para que el C18 no lo compile. La **Figura 3** muestra que podemos comentar manualmente cada uno. En esta instancia, cabe mencionar que el **stack de Microchip** está organizado en varios archivos; los encabezados



Al momento de implementar un sistema, debemos programar las funciones en C con el MPLAB

que habilitan los servicios se encuentran en **TCPIPConfig.h**. No todas las funciones del stack están disponibles a través del TCP/IP Configuration Wizard, por lo que es importante acceder al archivo TCPIPConfig.h y comentar los servicios que no precisamos. De esta forma, estaremos optimizando el uso de la memoria de programa. Por ejemplo, para deshabilitar el DNS, la instrucción debe quedar comentada por medio del operador:

```
//#define STACK_USE_DNS
```

El archivo **TCPIPConfig.h** está en la siguiente ruta: **C:/Microchip Solutions Library/TCPIP Demo App/TCPIPConfig.h**. En un primer contacto, es interesante recorrer este archivo y notar las definiciones de dirección IP, MAC y demás parámetros asociados a la conexión del Web Server.

FUNCIONES ESPECIALES DEL STACK

Por medio de TCP/IP Configuration Wizard, podemos agregar funciones especiales. Una opción sumamente útil es hacer un **conversor RS232 - Ethernet**. De este modo, los paquetes que ingresen por el conector de red serán retransmitidos por la **UART** del **PIC** y viceversa. Así, podremos darle conectividad

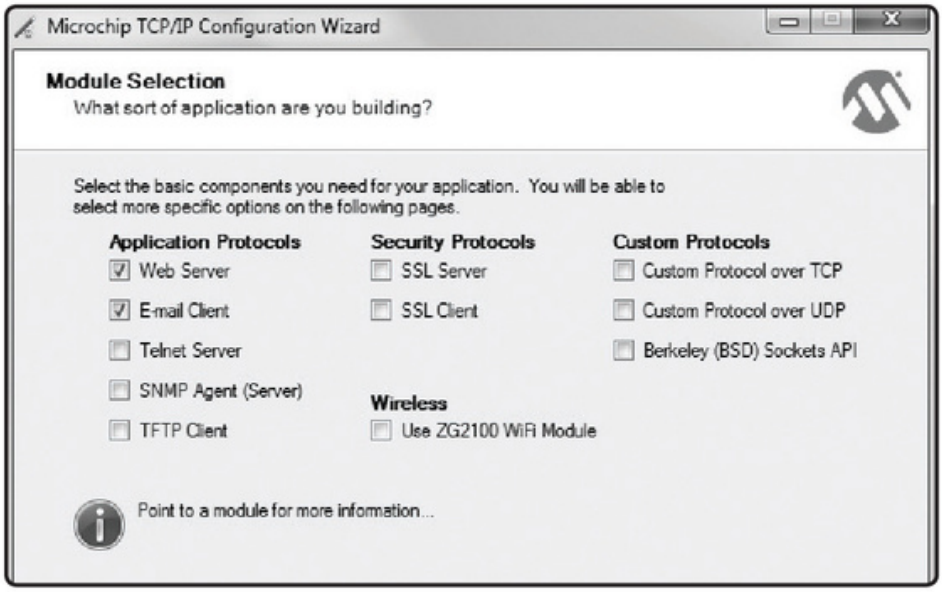


FIGURA 2.
Opciones de servidor web y del cliente de e-mail seleccionadas en el TCPIP Configuration Wizard.

```
// ============================================================================
//   Application Options
// ============================================================================

/* Application Level Module Selection
 *   Uncomment or comment the following lines to enable or
 *   disabled the following high-level application modules.
 */
#define STACK_USE_UART                  // Application demo using UART for IP address display
#define STACK_USE_UART2TCP_BRIDGE       // UART to TCP Bridge application example
//#define STACK_USE_IP_GLEANING
#define STACK_USE_ICMP_SERVER           // Ping query and response capability
#define STACK_USE_ICMP_CLIENT           // Ping transmission capability
//#define STACK_USE_HTTP_SERVER         // Old HTTP server
#define STACK_USE_HTTP2_SERVER          // New HTTP server with POST, Cookies, Authentication,
//#define STACK_USE_SSL_SERVER          // SSL server socket support (Requires SW300052)
//#define STACK_USE_SSL_CLIENT          // SSL client socket support (Requires SW300052)
#define STACK_USE_AUTO_IP               // Dynamic link-layer IP address automatic configurati
#define STACK_USE_DHCP_CLIENT           // Dynamic Host Configuration Protocol client for obta
#define STACK_USE_DHCP_SERVER           // Single host DHCP server
//#define STACK_USE_FTP_SERVER          // File Transfer Protocol (old)
#define STACK_USE_SMTP_CLIENT           // Simple Mail Transfer Protocol for sending email
#define STACK_USE_SNMP_SERVER           // Simple Network Management Protocol v2C Community Ag
//#define STACK_USE_TFTP_CLIENT         // Trivial File Transfer Protocol client
#define STACK_USE_GENERIC_TCP_CLIENT_EXAMPLE    // HTTP Client example in GenericTCPClient.c
#define STACK_USE_GENERIC_TCP_SERVER_EXAMPLE    // ToUpper server example in GenericTCPServer.
#define STACK_USE_TELNET_SERVER         // Telnet server
#define STACK_USE_ANNOUNCE              // Microchip Embedded Ethernet Device Discoverer serve
#define STACK_USE_DNS                   // Domain Name Service Client for resolving hostname s
#define STACK_USE_NBNS                  // NetBIOS Name Service Server for repsonding to NBNS
#define STACK_USE_REBOOT_SERVER         // Module for resetting this PIC remotely.  Primarily
#define STACK_USE_SNTP_CLIENT           // Simple Network Time Protocol for obtaining current
```

FIGURA 3.
Cómo habilitar o deshabilitar los servicios manualmente.

Ethernet a sistemas que, originalmente, solo tenían comunicación RS232. Vale destacar que la mayoría de las aplicaciones con microcontoladores utilizan el protocolo **RS232** para tener comunicación con una PC. Otra función interesante es medir el tiempo de viaje de un paquete entre el cliente y el host, lo que comúnmente denominamos **PING**. El stack nos devuelve el tiempo de viaje en milisegundos. Para activar este servicio, debemos marcar **`TCP Performance Test`** desde el TCPIP Configuration Wizard (**Figura 4**).

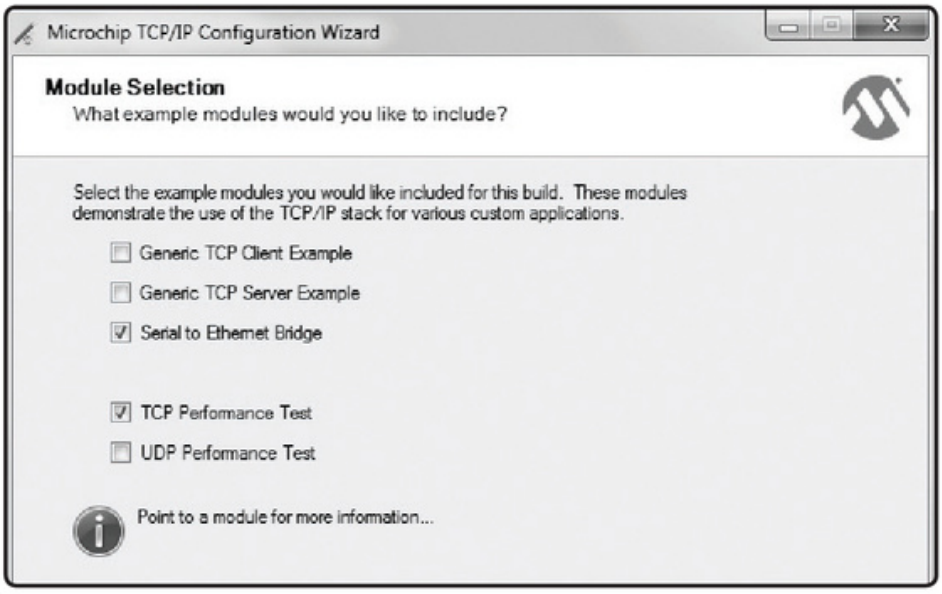


FIGURA 4. **Habilitación de las funciones especiales** `Serial to Ethernet Bridge` **para realizar la conversión serie-Ethernet, y** `TCP Performance Test` **para medir el tiempo de viaje de los paquetes.**

Vamos ahora a definir el **nombre de host**, es decir, uno con el cual identificar a nuestra placa en la red, de modo que no tengamos que recordar la dirección IP. Por otra parte, como la IP puede ser asignada por el servidor en forma dinámica, resulta conveniente definir un nombre de host que no varíe ante diferentes asignaciones.

En vinculación estrecha al nombre de host, está la dirección **MAC**. Por norma, esta debe ser unívoca; esto quiere decir que existe un ente (en este caso, la IEEE) que asigna a los fabricantes direcciones únicas. Por lo tanto, al momento de comercializar nuestro desarrollo, tendremos que adquirir un bloque de direcciones MAC a la IEEE.

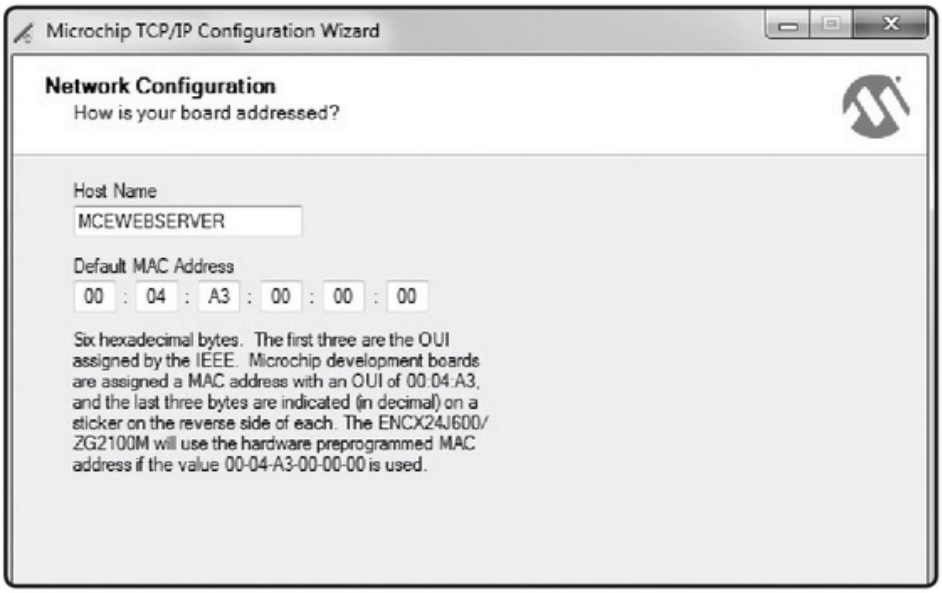


FIGURA 5. **Dirección MAC predefinida del PIC y nombre de host.**

Es posible configurar el servicio cliente de e-mail para que el stack envíe un mensaje ante cierto evento

Existen memorias de Microchip que tienen pregrabada una MAC única, que podemos vincular al microcontrolador a través de la interfaz **SPI** o **I2C**. Claro que para hacer pruebas, podemos recurrir a la MAC predefinida que trae el stack (**Figura 5**).

Configuración del Web Server

A continuación, vamos a establecer el nombre de la página principal, es decir, aquella que se mostrará por defecto; en este caso, es **index.htm**. Cada vez que coloquemos la IP de nuestro Web Server en el explorador, se cargará la página por defecto. En cambio, si queremos acceder a una página distinta, tendremos que ingresar: **http://192.168.1.103/variables.htm**, donde primero observamos la dirección IP y, luego, el nombre de la página a la cual estamos accediendo (**variables.htm**).

Un parámetro relevante es la cantidad de conexiones que vamos a permitir, lo cual hace referencia a la cantidad de usuarios que pueden monitorear el sistema simultáneamente. El stack permite un máximo de 32 conexiones a la vez, pero antes de establecer este valor, debemos considerar aspectos como el ancho de banda y la estabilidad de la red, ya que una excesiva demanda puede deteriorar el rendimiento del sistema.

UBICACIÓN DE LA PÁGINA WEB

Es posible establecer diferentes ubicaciones para almacenar la página web que verá el usuario final del **Web Server**. Esta puede encontrarse en la memoria de programa del microcontrolador o en una memoria externa del tipo **EEPROM**, a la cual se accede desde el PIC a través de SPI.

Si bien en la **Figura 6** vemos otras opciones, como memoria Flash y sistema FAT (permite almacenar la página en una tarjeta SD), las mencionadas anteriormente son las más representativas, porque determinan si la página está alojada en la memoria del microcontrolador o en una externa.

En los ejemplos de esta colección, vamos a utilizar la memoria externa de **Microchip: 25LC1024**. Es del tipo EEPROM, tiene una capacidad de **1 Mbit** y se conecta al PIC por SPI. Con una memoria externa, dispondremos de toda la capacidad del programa para escribir nuestro código y alojar el stack TCP/IP.



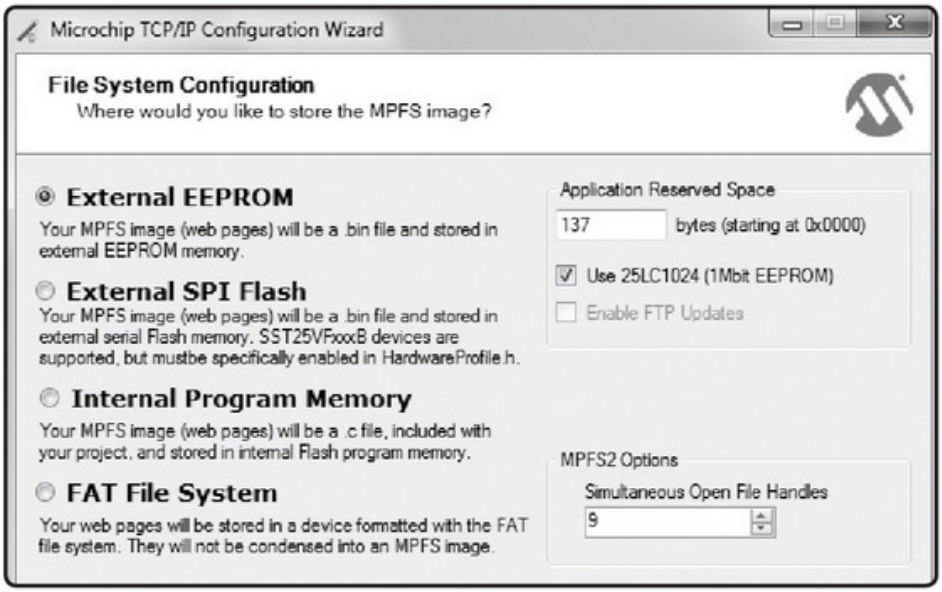


FIGURA 6. **Opciones de almacenamiento para la página web.**

COMPILAR EL STACK

Una vez configurados los parámetros básicos de la aplicación a través del Microchip TCP/IP Configuration Wizard, estamos en condiciones de compilar el stack y cargarlo en el PIC.

Para empezar, abrimos el proyecto **TCPIP Demo App-C18.mcp** desde el MPLAB, y ejecutamos la compilación haciendo clic en **Build all**. De este modo, estamos generando un archivo **HEX** listo para grabar en el PIC.

Dentro de la carpeta **TCPIP Demo App**, encontramos varios proyectos preparados para utilizar con micros de 8 a 32 bits. Es interesante destacar que los parámetros configurados con el asistente sirven para cualquiera, ya que solo se está modificando el archivo de cabecera **TCPIPConfig.h**. Seleccionamos el proyecto **TCPIP Demo App-C18.mcp** porque vamos a trabajar con un micro de la familia PIC18 (8 bits), como venimos haciendo a lo largo de la colección. En este caso, el micro es el PIC18F97J60, que incorpora el módulo Ethernet.

La ruta predefinida para acceder al proyecto es: **C:/Microchip Solutions Library/TCPIP Demo App/ TCPIP Demo App-C18.mcp**. Una vez compilado, precisamos un microcontrolador donde cargar

CUESTIÓN DE ESPACIO

Básicamente, tenemos dos opciones para almacenar la página web en el sistema: memoria externa o memoria interna del PIC. Debemos decidir por una u otra considerando el costo adicional que significa una EEPROM como la **25LC1024**.

el proyecto. Vamos a utilizar la placa comercial **MCE WebServer DEV**, desarrollada por la empresa MCElectronics (**www.mcelectronics.com.ar**). En este punto, la única finalidad es mostrar la capacidad del stack y todas sus funciones, motivo por el cual no veremos en detalle el hardware hasta la próxima clase. Simplemente, vamos a mencionar que la placa trae una memoria externa de 1 Mbit para alojar la web, un sensor de temperatura que utilizaremos en las prácticas y dos relays para controlar un motor (**Figura 7**).

ACCEDER A LA PÁGINA WEB

El stack predefinido habilita el cliente **DHCP**, por lo que podemos conectar la placa a un router con DHCP habilitado (capacidad de asignar direcciones IP) y obtener conectividad desde una computadora en la misma red. Si no contamos con un router, es posible conectar el Web Server directamente a la placa de red de la computadora. En este caso, la IP es fija y se asigna en forma estática. La IP fija puede cambiarse desde el **TCP/IP Configuration Wizard** o desde el archivo **TCPIPConfig.h**.

Los parámetros configurados con el asistente sirven para cualquier proyecto

Vamos a conectar el servidor web a un router; en este caso, la IP asignada es **192.168.0.101**. Si lo conectamos a la PC, la IP cambia, y es por eso que el ejemplo sirve para ambos casos. Sobre la izquierda de la página inicial del **Web Server** aparece un menú con las distintas opciones de la placa

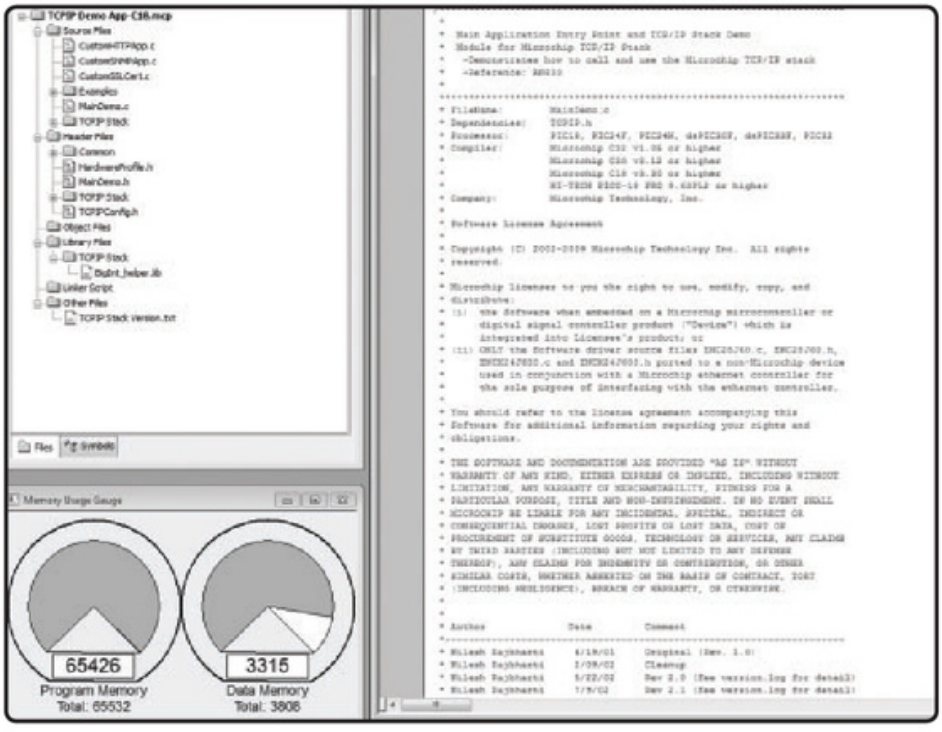


FIGURA 7.
Proyecto en el MPLAB listo para compilar.

(Figura 8), las cuales habilitamos a través del TCP/IP Configuration Wizard, como envío de e-mails, servidor HTTP y servidor seguro (nos permite acceder a una página protegida).

Lo que nos interesa en este momento es el recuadro ubicado arriba a la derecha, que muestra en tiempo real el estado de **8 LEDs** y la temperatura enviada desde el sensor que tiene la placa. Podríamos, por ejemplo, monitorear otro LED o controlar un relay desde la interfaz web. Este tipo de interacción, donde una variable reporta o modifica el estado de un puerto, se realiza mediante funciones **`callback`**, que veremos más adelante. Es interesante destacar que la actualización de los datos es automática y no requiere refrescar la pantalla del navegador.

En la sección **`Configuración`**, encontramos parámetros como la dirección IP, el default gateway, la máscara de red, la MAC y el servidor DHCP (**Figura 9**). Todo esto puede cambiarse en tiempo de ejecución y, es importante, protegerlo mediante una contraseña.

El stack predefinido habilita el cliente DHCP, por lo que podemos conectar la placa a un router



También tenemos la posibilidad de enviar mensajes desde un formulario web, para lo cual se utiliza el protocolo SMTP. Simplemente, hay que completarlo con el nombre del servidor **SMTP**, el nombre de usuario y la clave.

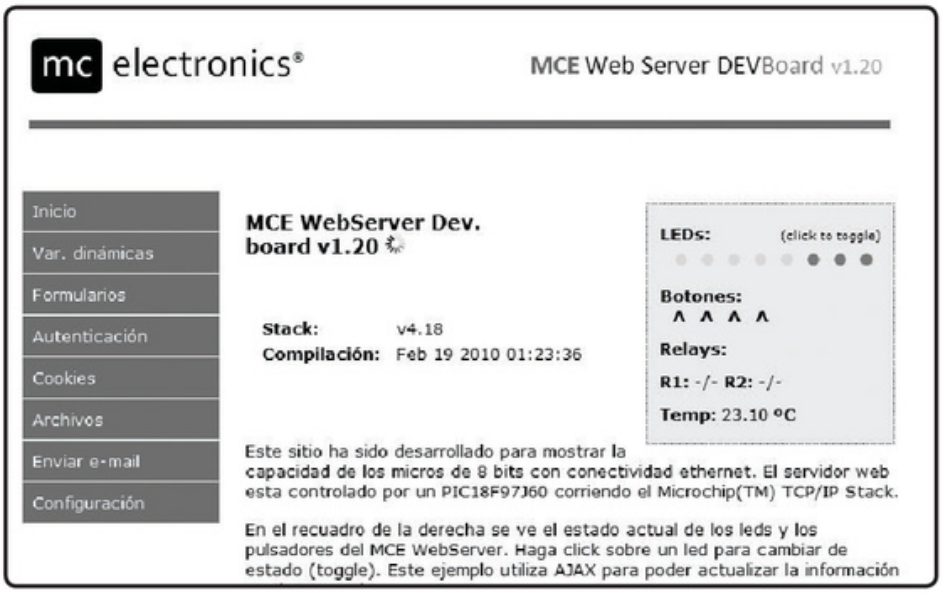


FIGURA 8. **Página inicial** index.htm, **a la que se accede colocando la dirección IP asignada por el router.**

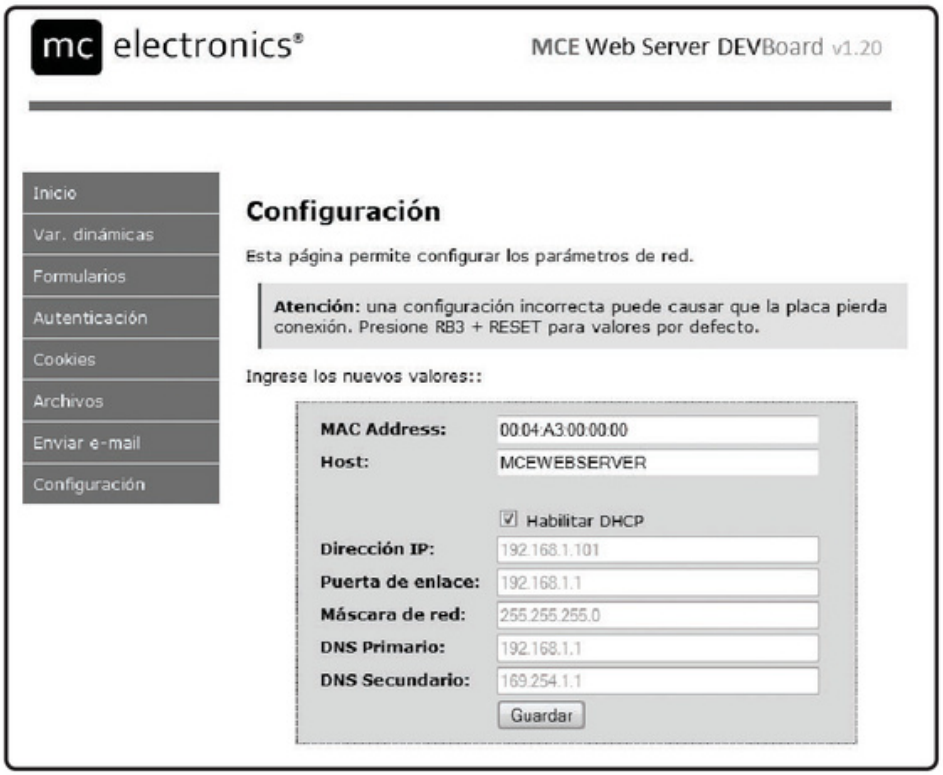


FIGURA 9.
Página para configurar los parámetros de la red.



FIGURA 10.
Formulario para envío de e-mails en forma manual.

Por último, colocamos la dirección del destinatario y el cuerpo del mensaje. Por defecto, se adjunta un archivo CSV (se puede abrir con Excel) donde se muestra el estado de los LEDs, los pulsadores y el valor de temperatura. Más adelante veremos el código de este ejemplo (Figura 10).

Antes de pasar a la programación del stack, analicemos una última pantalla de la interfaz web (**Figura 11**). Se trata del envío de variables a través de los métodos **GET** y **POST**. Quien esté acostumbrado a programar formularios web sabrá la ventaja que conlleva utilizar este recurso.

Por ejemplo, con el método GET podemos enviar una serie de variables junto con la URL; es lo que habitualmente vemos como: **http://192.168.1.101/forms.htm?relay2=on&relay1=off**.

Luego de la dirección IP de la placa, está el nombre del formulario que vamos a actualizar y, a continuación del signo de interrogación, el nombre de cada variable y su valor, separadas por el signo **&**.

Notemos que estamos actualizando el estado de dos variables (**relay1** y **relay2**) de una sola vez. GET tiene una limitación, y es el hecho de que la URL no puede contener más de 1024 bytes. Si necesitamos enviar una mayor cantidad de variables, debemos utilizar el método POST, que no tiene restricciones de longitud. Si bien implementar este método en el stack TCP/IP es un poco más laborioso, suele hacerse para actualizar el texto de un LCD. Otra ventaja de este sistema radica en que la URL no se ve afectada, y el usuario final no ve las variables que estamos actualizando.

LOS ARCHIVOS PRINCIPALES DEL STACK

Como mencionamos anteriormente, es necesario conocer el stack para poder agregar nuestras propias funciones y, así, lograr un sistema de telemetría y control remoto acorde a nuestras necesidades. Para este fin, vamos a desarrollar los archivos principales del stack. En la **Figura 12** vemos la estructura general del TCP/IP Stack en el MPLAB.

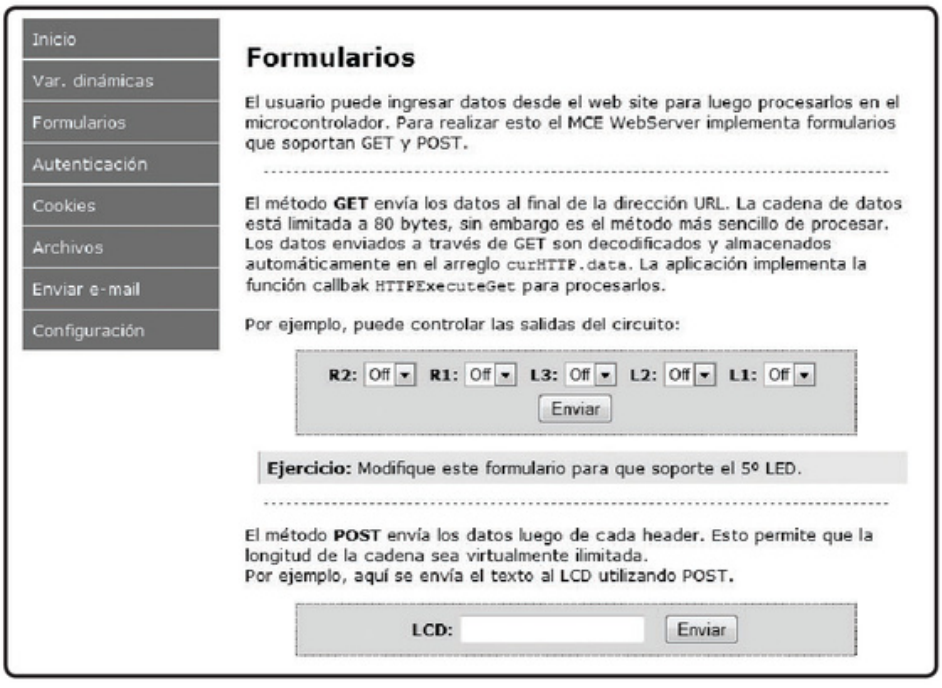


FIGURA 11.
Formulario de prueba para los métodos GET y POST.

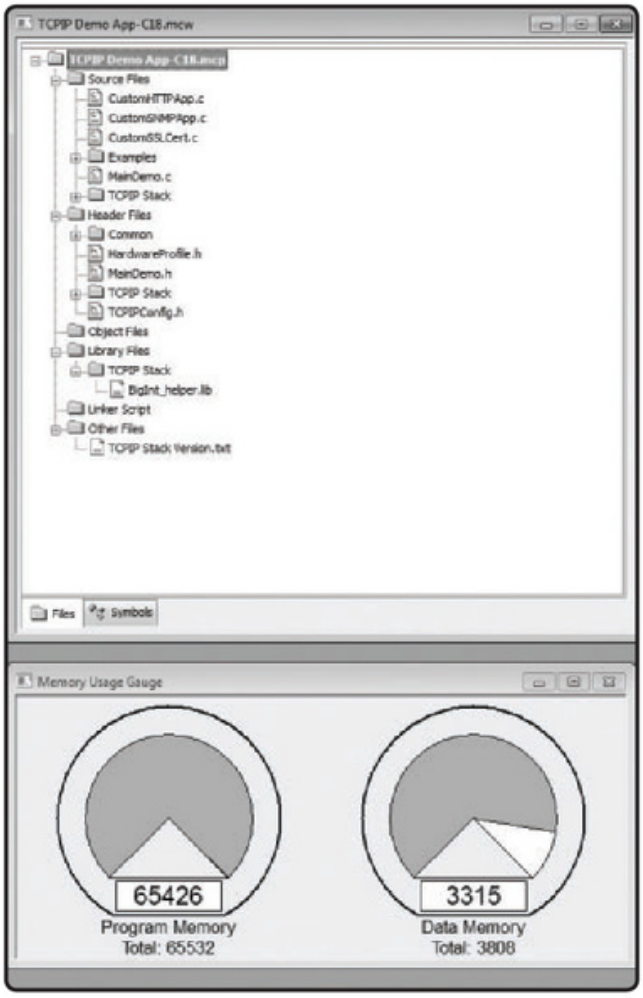
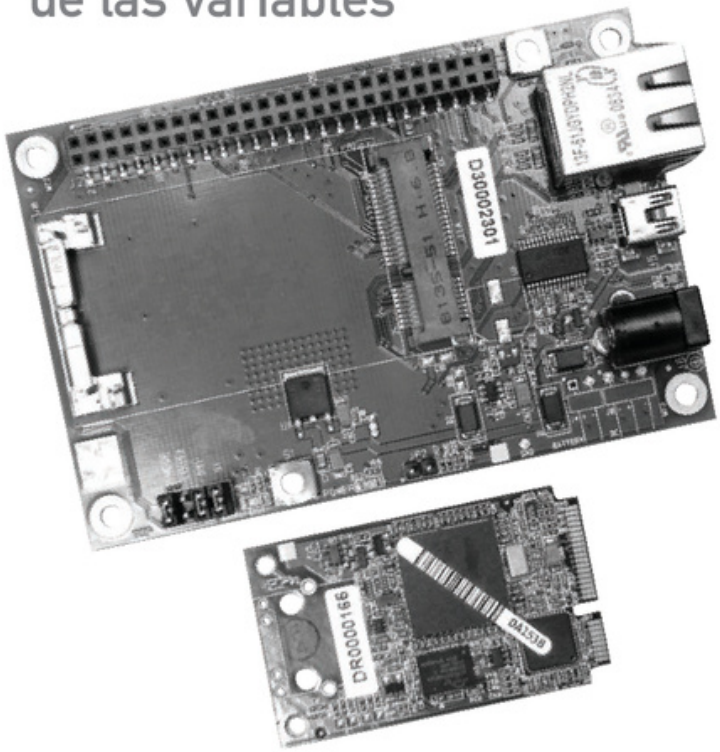


FIGURA 12. **Estructura de archivos del stack.**

Podemos enviar un e-mail desde la interfaz web manualmente para reportar el estado de las variables



Se distingue la estructura dividida en archivos de cabecera (**.h**) y el código en los archivos (**.c**). Los archivos que necesariamente debemos conocer y que nos permiten personalizar nuestra aplicación son:

- **HardwareProfile.h**: en este archivo, asociamos un puerto a un nombre fácil de recordar. Por ejemplo, si colocamos un LED en el puerto RF3 del PIC:

```
#define LED_TRIS (TRISFbits.TRISF3)
#define LED_IO  (PORTFbits.RF3)
```

Estamos asociando el nombre **LED_IO** al puerto **RF3**. Entonces, cada vez que pongamos **LED_IO = 1**, el port **RF3** se pondrá en "uno lógico".

- **TCPIPConfig.h**: como vimos, se utiliza para habilitar o deshabilitar funciones del stack como DNS, FTP, SNTP, SSL y UART–, dependiendo de las características de nuestra aplicación y de la memoria disponible en el micro.
- **MainDemo.c**: es el programa principal, en el que podemos encontrar el mapeo de puertos, la configuración de los registros de la **USART**, el

ADC (conversor analógico digital), la interfaz SPI y la inicialización de la placa. Por ejemplo, colocamos ADCON0=0x0D; (Registro ADCON0 del PIC) para seleccionar el pin AN3 como entrada analógica.

- **CustomHTTPApp.c**: este es el cuarto archivo que necesitamos para una aplicación básica. En él se encuentran las funciones callback, es decir, aquellas que interactúan con la pagina web y reportan el estado de las variables dinámicas.

FUNCIONES CALLBACK

Echemos un vistazo al archivo **CustomHTTPApp.c**, donde se encuentran las funciones sallback. En este ejemplo, vamos a mostrar el estado de un LED en la página web por medio de **ON** y **OFF**. La función **callback** contiene el nombre y el valor de la variable dinámica (LED) que va a mostrarse en la Web (**Figura 13**). Un error común es enviar directamente al buffer Ethernet el estado del LED mediante la variable int **LED_IO**. Pero hay que tener en cuenta que solo pueden enviarse strings por HTTP, por lo que, en numerosas ocasiones, es sumamente útil emplear las funciones **itoa** o **ftoa** (entero a ASCII y float a ASCII) para convertir los tipos de datos. Analicemos el siguiente código:

```
ROM BYTE ESTADO_LED_UP[] = "ON";
ROM BYTE ESTADO_LED_DOWN[] = "OFF";

void HTTPPrint_LED(void)
{
  TCPPutROMString(sktHTTP,
(LED_IO?ESTADO_LED_UP:ESTADO_LED_DOWN));
}
```

En este caso, en vez de usar los conversores de tipo, creamos dos variables auxiliares, **ESTADO_LED_UP y ESTADO_LED_DOWN**, para mostrar el estado del **LED**

FIGURA 13. **Vemos una parte del archivo** `CustomHTTPApp.c`, **donde se definen las funciones** `callback`.

```
  Function:
    void HTTPPrint_varname(void)

  Internal:
    See documentation in the TCP/IP Stack API or HTTP2.h for details.
  ***************************************************************************/

void HTTPPrint_builddate(void)
{
    curHTTP.callbackPos = 0x01;
    if(TCPIsPutReady(sktHTTP) < strlenpgm((ROM char*)__DATE__" "__TIME__))
        return;

    curHTTP.callbackPos = 0x00;
    TCPPutROMString(sktHTTP, (ROM void*)__DATE__" "__TIME__);
}
```

FUNCIONES CALLBACK

Se encuentran en el archivo `CustomHTTPApp.c`, dentro del proyecto del TCP/IP Stack. Son las funciones que nos permiten, por ejemplo, encender un LED, monitorear la temperatura o controlar un motor a través de Ethernet.



mediante ON y OFF. La función **void HTTPPrint_LED(void)** envía a la página web el valor de la variable dinámica **LED**. Para verificar el estado del puerto RF3 (donde está mapeado el LED), utilizamos la sentencia condicional IF, aquí representada en una línea, como se muestra a continuación.

```
LED_IO?ESTADO_LED_UP:ESTADO_LED_DOWN
```

Recordemos que, anteriormente, definimos **LED_IO** como el estado del puerto **RF3**. El **IF** revisa el estado de **LED_IO**. Si el puerto está en 1, entonces la sentencia es verdadera y envía, por **HTTP**, **ESTADO_LED_UP**; en cambio, si está en cero, la sentencia es falsa y envía **ESTADO_LED_DOWN**. De esta forma, tenemos un sistema básico que reporta el estado de las variables ante un llamado del servidor HTTP.

Ahora debemos diseñar una página web que muestre las variables al usuario final.

PROGRAMACIÓN DE LA PÁGINA WEB

Microchip organiza los archivos del sitio web de modo de ahorrar espacio. Se trata de no repetir la información (**Figura 14**). Por ejemplo, al pie de todas las páginas aparece la misma leyenda. Entonces, utilizamos el archivo **footer.inc** que será llamado desde cada página, en vez de escribir el texto otra vez. Esto permite reducir sensiblemente el tamaño total del sitio; recordemos que tenemos 1 Mbit para almacenar la Web.

En el sitio hay que incluir la variable dinámica **LED**, para lo cual tenemos que modificar los archivos **status.xml** e **index.htm**. La ruta predefinida para acceder a los archivos que componen el sitio es **C:/Microchip Solutions Library/TCPIP Demo App/ WebPages2/**.

La función callback contiene el nombre y el valor de la variable dinámica (LED) que se mostrará en la Web

CONVERSIÓN DE DATOS

Solo es posible enviar strings a través de HTTP. Es adecuado utilizar las funciones itoa (entero a ASCII) y ftoa (float a ASCII) para convertir los tipos de datos. Para monitorear una variable, como la temperatura, tendremos que convertirla de float a ASCII antes de ponerla en el buffer de Ethernet.

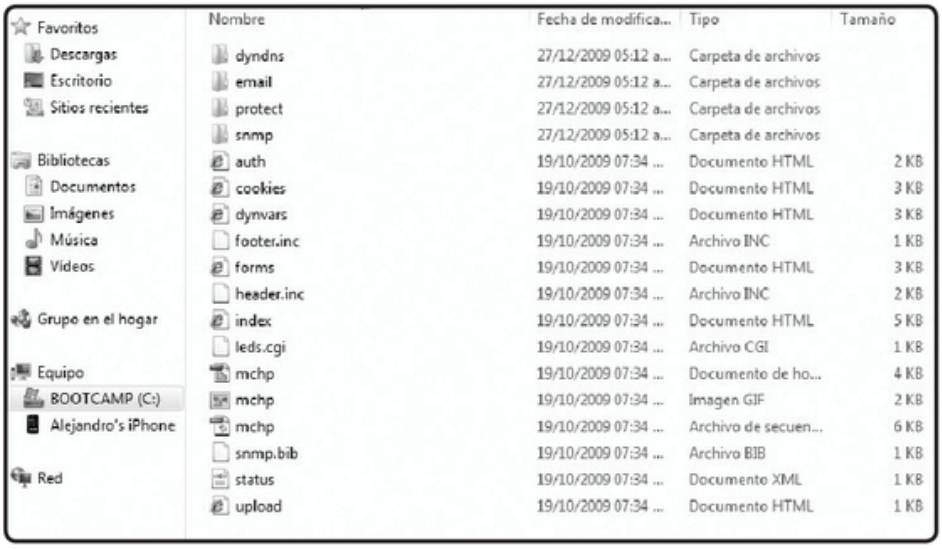


FIGURA 14.
Estructura de archivos que componen el sitio web.

Por un lado, **status.xml** contiene todas las variables dinámicas que se actualizan (LEDs, temperatura y pulsadores). Agregamos la línea correspondiente a la nueva variable dinámica:

```
<LED>~LED~</LED>
```

Entre los signos < > se indica el nombre del divisor que se utilizará en el index para definir la posición de la variable. Por su parte, **Index.html** es el archivo que ve el usuario final. Donde aparezca **<"LED">**, el Web Server lo reemplazará con **ON** u **OFF** según el valor de **LED_IO**:

```
<span id="LED">?</span>;
```

Es importante destacar que este ejemplo utiliza AJAX para actualizar la información sin tener que presionar la tecla **<F5>** en el explorador. A continuación, debemos agregar la siguiente sentencia al final del archivo **index.htm**:

```
document.getElementById('LED').innerHTML
= getXMLValue(xmlData, 'LED');
```

Lo que hicimos hasta ahora fue crear una variable dinámica LED en el código del stack y enviarla a la Web a través de una función **callback**. Luego, configuramos los archivos **index.htm** y **status.xml** para visaulizar el estado de la variable en la página.

SISTEMA DE ARCHIVOS MPFS

Ahora necesitamos cargar la Web en la memoria externa 25LC1024. Antes de hacerlo, debemos compactar los archivos que componen el sitio mediante la aplicación MPFS (*Microchip PIC File System*), que se instala con el stack y se encuentra en la



ruta que venimos trabajando. El acceso directo desde el menú **Inicio** es **Microchip/TCPIP Stack - MPFS2**.

MPFS es el sistema de archivos que utiliza el micro para leer la página web. Una vez que tenemos nuestro sitio terminado con los archivos **XML**, **HTML**, **JPG** y **GIF**, debemos convertirlos al sistema **MPFS**, para, luego, poder cargarlos en el micro o en la memoria externa. Lo que hace internamente este conversor es juntar todos los archivos en uno solo; de hecho, podemos abrir el resultante con el Bloc de notas y veremos, ordenado en forma secuencial, el contenido de cada archivo que compone el sitio (**Figura 15**).

El MPFS permite transferir la Web automáticamente a la memoria externa luego de compactarla

La aplicación **MPFS.exe** (**Figura 16**) convierte el conjunto de archivos que integran la página web a dos formatos posibles: **.bin** para cargar el website en la memoria EEPROM externa y **.c** para hacerlo en la memoria de programa del PIC (**Figura 17**).

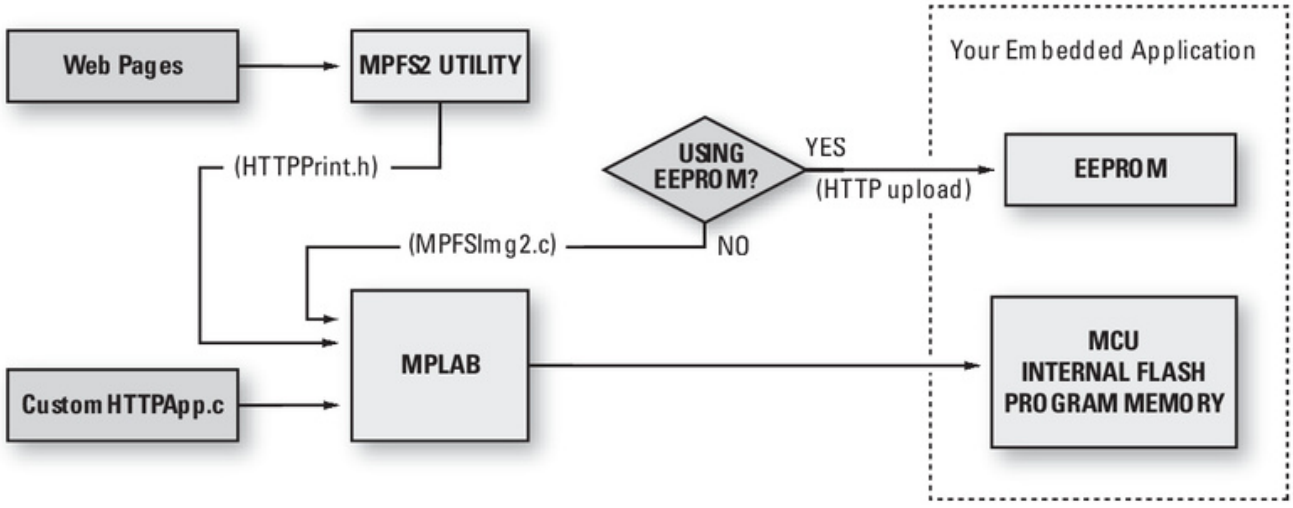


FIGURA 15. **En este diagrama vemos las distintas opciones para almacenar la página.**

SOBRE AJAX

AJAX significa *Asynchronous JavaScript And XML*. Es una técnica para crear aplicaciones interactivas que se ejecutan en el navegador del cliente mientras se mantiene la conexión asíncrona con el servidor. Permite actualizar la información de la Web sin necesidad de refrescar la pantalla.

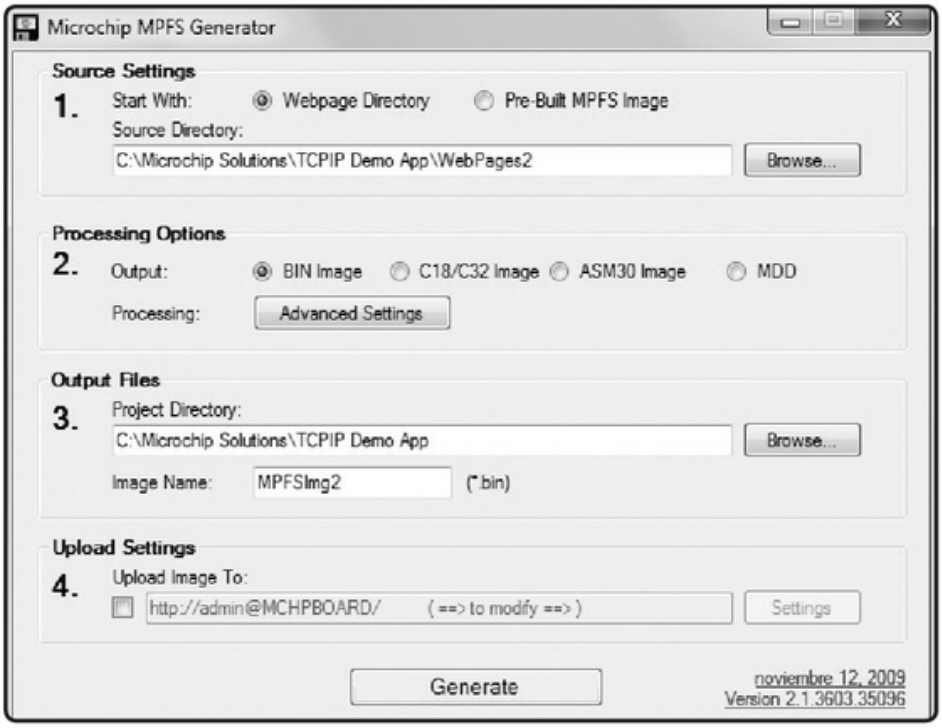


FIGURA 16. **Pantalla principal del `MPFS.exe`, con las opciones requeridas para compactar la página y cargarla en la memoria.**

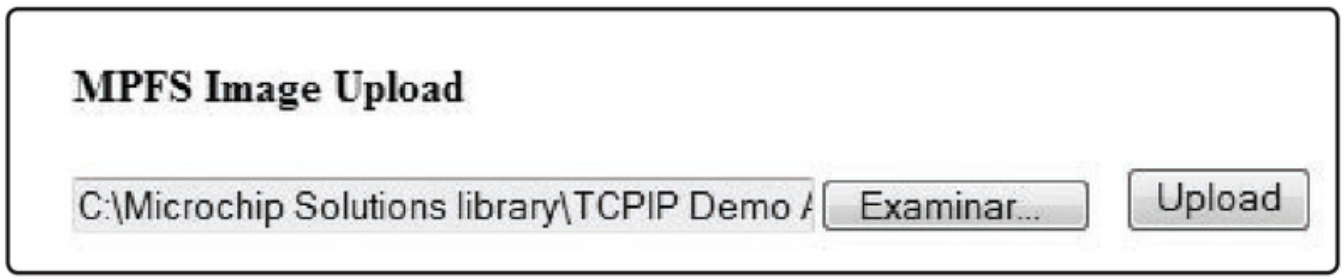


FIGURA 17. **Servicio para cargar el sitio web en forma remota, con el cual debemos incluir el archivo `.bin` resultante de la compactación.**

Al cargar la Web en la memoria externa, toda la memoria de programa del PIC queda disponible para la aplicación. Si bien esta opción es más costosa (porque requiere una memoria externa), priorizamos la disponibilidad de la memoria interna para escribir las funciones callback y demás código necesario.

El MPFS nos da la opción de transferir la Web automáticamente a la memoria externa luego de compactarla. Para hacerlo, la placa debe estar conectada a la red (o la PC) y tener cargada la última versión del Microchip TCP/IP Stack.

Debemos tener en cuenta que cada variable que agreguemos en la Web requiere su correspondiente variable en el stack. Por lo tanto, es necesario recompilar el stack y cargarlo en el PIC antes de compactar la página y transferirla a la memoria.



Cada variable que agreguemos en la Web requiere su correspondiente variable en el stack

La secuencia correcta es la siguiente:

- Modificar el stack según las necesidades de telemetría requeridas.
- Agregar las variables dinámicas correspondientes en status.xml e index.htm.
- Compactar la Web con MPFS.
- Compilar el stack con el MPLAB.
- Cargar el stack en el PIC.
- Transferir la página a la memoria externa.

Es preciso respetar este procedimiento, porque si las variables de la Web no se corresponden con las del stack, la página no se cargará en el explorador. Por lo general, lleva tiempo darse cuenta de este error.

Tenemos dos alternativas para cargar la Web en la memoria externa: a través del MPFS o en forma remota por medio del explorador. Este último es un sistema de transferencia de archivos por medio de HTTP, para lo cual debemos ingresar en **http://192.168.1.101/mpfsupload**. Las aplicaciones más seguras no permiten transferir la Web a la memoria externa por este medio, porque cualquiera podría acceder a la dirección mencionada conociendo solo la IP, y así modificar la página. En este caso, debemos proteger el sistema de carga mediante contraseña o bien almacenar la Web en la memoria de programa del PIC.

Hay que colocar la menor cantidad posible de imágenes en la aplicación final, dado que la memoria **25LC1024** posee **1 Mbit** de espacio, y si nuestra web ocupa más, no se cargará correctamente.

HTTPPRINT.H

Es importante tener en cuenta que, al agregar o quitar variables de nuestra web (por medio de `status.xml`), se modifica el archivo `HTTPPrint.h`, por lo que hay que volver a compilar el stack antes de cargar la aplicación en el micro.

CARGAR LA WEB EN LA MEMORIA INTERNA DEL PIC

En aplicaciones que deben ser extremadamente económicas o en las que la seguridad es un factor prioritario, podemos optar por almacenar la Web en la memoria de programa del microcontrolador. Para hacerlo, ejecutamos el MPFS y seleccionamos **C18/C32 Image** desde el menú principal.

Esto generará un archivo **.c** que debemos agregar al stack para que sea compilado con el resto del código. Por defecto, el nombre del archivo es **MPFSImg2.c** (es decir, imagen de la Web en Microchip PIC File System). El número 2 hace referencia a que el stack TCP/IP incorpora una versión mejorada del MPFS (**Figura 18**). Para agregar la Web –es decir, el archivo **MPFSImg2.c**–, debemos seguir dos pasos.

Primero, comentamos la sentencia **#define MPFS_USE_EEPROM** dentro del archivo **TCPIPConfig.h**. De este modo, estamos indicando que la Web no estará en la memoria EEPROM externa.

Segundo, incluimos el archivo **MPFSImg2.c** en el proyecto, para lo cual hacemos clic con el botón derecho del mouse sobre **Source Files** y seleccionamos **Add Files**, como se muestra en la **Figura 19**. De esta forma, prescindiremos de la memoria **EEPROM** externa, con el consecuente ahorro de dinero y espacio físico en la placa.

Ya tenemos la Web incluida en el stack, ahora debemos compilar el proyecto completo y cargarlo en el PIC. Una vez que compilamos, podemos notar que queda mucho menos espacio disponible en la memoria de programa. Una página promedio ocupa cerca del 25% de la memoria de un **PIC18F97J60**. Este es el PIC que está en la placa **MCE WebServer DEV** y que vamos a estudiar en detalle en la próximo capítulo. Por lo tanto, en caso de que necesitemos

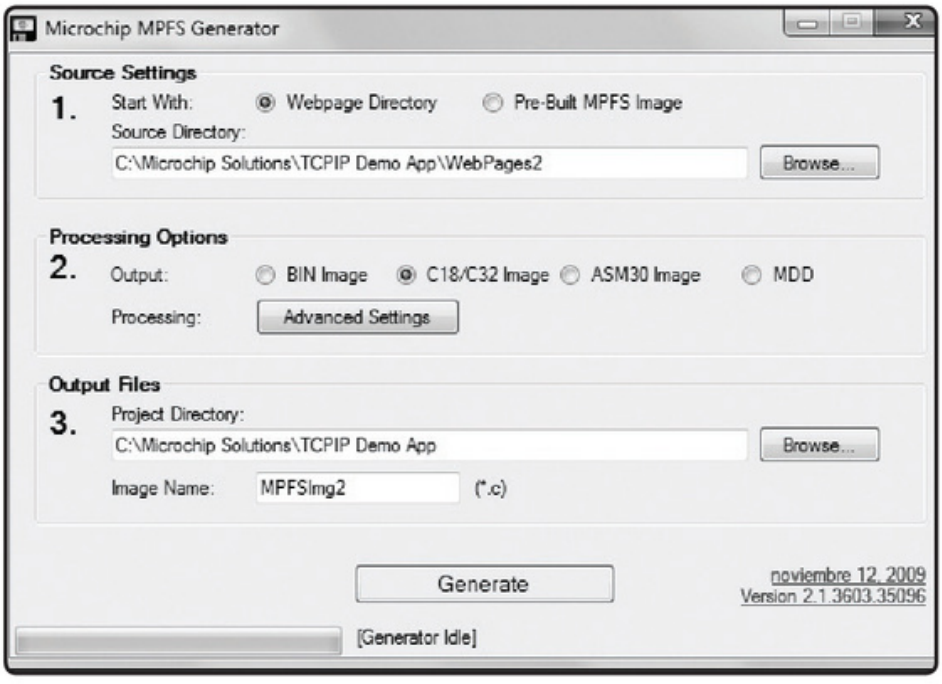


FIGURA 18.
MPFS configurado para generar un archivo .c.

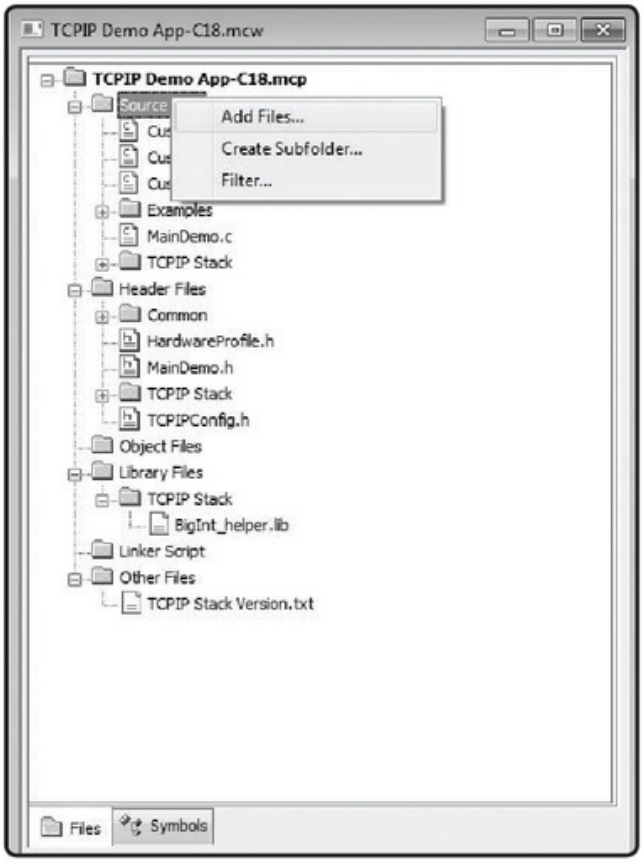
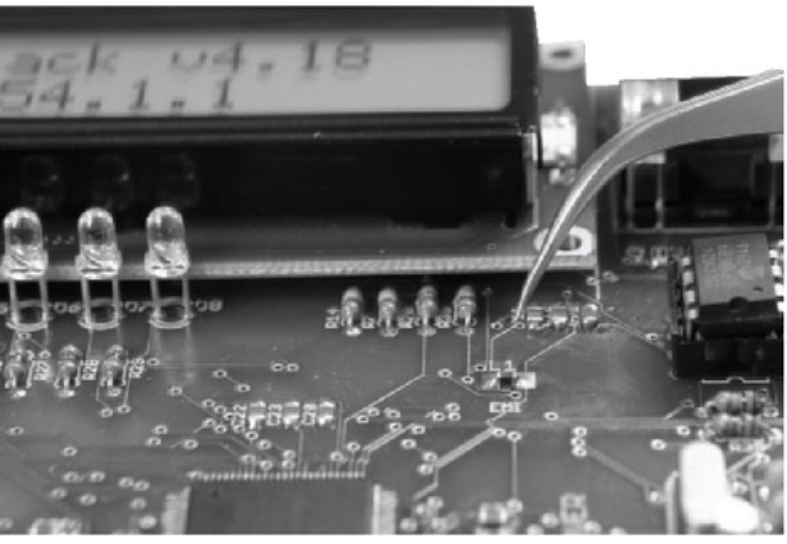


FIGURA 19. **Aquí vemos cómo agregar el archivo** `MPFSImg2.c` **al proyecto.**



almacenar la Web de esta forma, es crítico reducir la cantidad de imágenes y minimizar su resolución al máximo posible.

MICROCHIP WEBPREVIEW

Esta aplicación permite visualizar el aspecto que tendrá la página antes de cargarla en el sistema, y resulta de gran utilidad porque, de otra forma, deberíamos compactarla y transferirla a la memoria (ya sea del PIC o la memoria externa) cada vez que modifiquemos algún parámetro estético. Microchip WebPreview se encuentra dentro de la carpeta del stack en el menú **Inicio**.

En primer lugar, ingresamos la ruta del archivo **index.htm** y la ubicación de la carpeta donde están las imágenes y demás archivos que componen el sitio (**Figura 20**). Respectivamente, son:

```
[C:\Microchip Solutions library\TCPIP
Demo App\WebPages2\index.htm]
[C:\Microchip Solutions library\TCPIP
Demo App\WebPages2]
```

Luego, hacemos clic en **Preview Page** y veremos la página generada por la aplicación en una ventana de Windows. Notemos que las variables, como las que indican el estado de los pulsadores, aparecen con (?).

En aplicaciones económicas, podemos almacenar la Web en la memoria de programa del microcontrolador



FIGURA 20.
Pantalla principal de Microchip WebPreview.

Esto es lógico, pues la aplicación no está cargada en el micro y, por lo tanto, no hay una función callback que devuelva esos valores. Tampoco tenemos posibilidad de interacción con los botones ni vemos el valor del sensor de temperatura.

Si bien el Microchip WebPreview es útil para visualizar el aspecto de nuestra página, necesariamente debemos compilar el proyecto y grabarlo en el microcontrolador para, luego, transferir la Web a la memoria. De este modo, verificamos que todas las funciones callback estén funcionando como corresponde. Además, debemos asegurarnos de que las variables dinámicas responden de manera adecuada.

Recordemos que siempre es necesario recompilar el stack si agregamos o quitamos variables dinámicas de la página web (**Figura 21**).

WebPreview permite previsualizar el aspecto que tendrá la página web desde la PC

Envío de e-mails

Para enviar e-mails desde el stack TCPIP debemos habilitar el cliente **SMTP**, dejando de comentar la sentencia **`STACK_USE_SMTP_CLIENT`** en el archivo **`TCPIPConfig.h`**. Las funciones requeridas agregan aproximadamente 1 K de código a la memoria de

WEB PREVIEW

El Microchip WebPreview permite visualizar el aspecto de la Web sin tener que transferirla a la memoria de programa del PIC o a la memoria externa 25LC1024, lo cual reduce notablemente el tiempo de desarrollo. Es muy útil al momento de diseñar la interfaz de usuario.

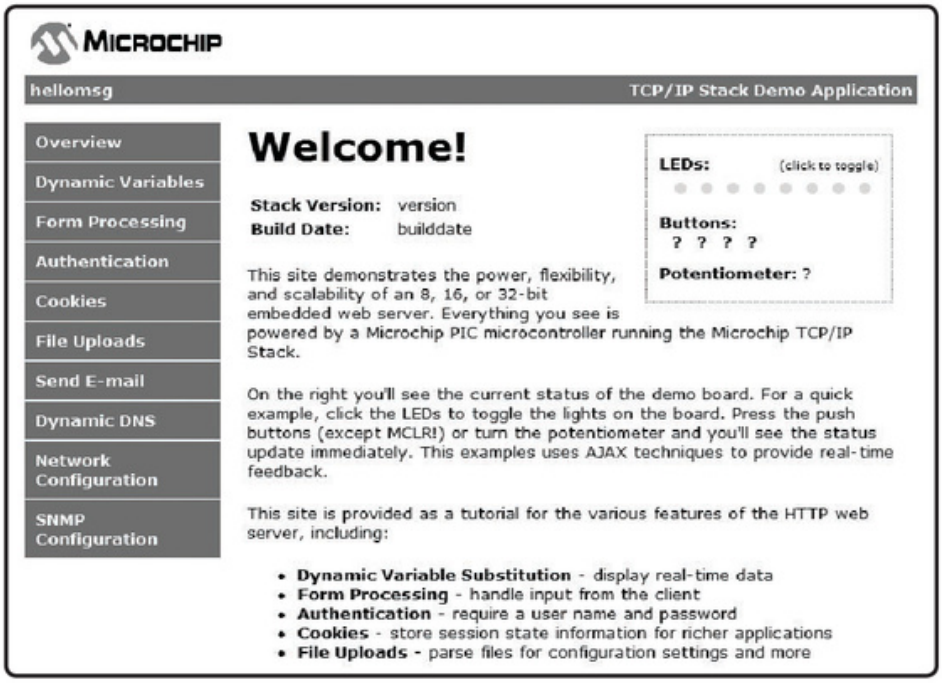


FIGURA 21.
Página generada con el WebPreview.

programa. El proceso puede invocarse en cualquier momento, por ejemplo, si sube la temperatura, si se presiona un pulsador, etc. El mail de este ejemplo envía el estado de los pulsadores a la dirección que se especifique.

Veamos un esquema que representa el funcionamiento del cliente SMTP. Antes de intentar enviar un mail con la función **SMTPSendMail()**, tenemos que verificar si no hay otro en proceso, para lo cual utilizamos, en primer término, **SMTPBeginUsage()**, como indica el diagrama.

Si nuestro servidor de correo saliente requiere autorización, usamos **SMTPClient.Server**, **SMTPClient.Username** y **SMTPClient.Password** para configurar los parámetros. Recordemos que esto lo hicimos cuando enviamos un e-mail desde el formulario generado por el stack. Por último, incluimos los campos De, Para, Asunto y Mensaje antes de llamar a **SMTPSendMail()**.

Mientras se intenta enviar el mensaje, estamos en el bucle de la función **SMTPIsBusy()**. En este punto, si el e-mail se envió correctamente, **SMTPEndUsage()**

APLICACIONES DEL SMTP

Este código se encuentra implementado en el main del TCP/IP Snack. Simplemente, debemos activar el servicio SMTP desde el archivo **TCPIPConfig.h**; e ingresar los datos del destinatario, los de nuestra cuenta de e-mail y el cuerpo del mensaje.

devolverá 0; de lo contrario, tendremos la opción de comenzar el proceso otra vez o cancelar el envío. Un e-mail puede no enviarse debido a fallas en la conexión o porque los datos de autentificación son incorrectos. Si ingresamos erróneamente la dirección del destinatario, el stack lo mandará de todas formas (Figura 22).

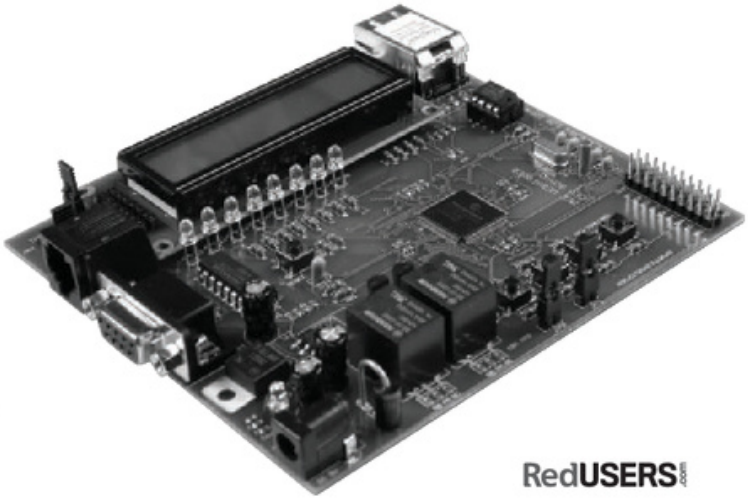
EJEMPLO COMPLETO

Veamos el proceso de envío, con un ejemplo:
En este caso, utilizamos **SMTPClient.Username**

```
if(SMTPBeginUsage())
{
static BYTE RAMStringTo[] = "soporte@
mcelectronics.com.ar";
static BYTE RAMStringBody[] = "Mensaje
generado por el MCE WebServer" VERSION
" \r\n\r\nBotones: ";
RAMStringBody[sizeof(RAMStringBody)-2]
= '0' + BUTTON0_IO;
RAMStringBody[sizeof(RAMStringBody)-3]
= '0' + BUTTON1_IO;
RAMStringBody[sizeof(RAMStringBody)-4]
= '0' + BUTTON2_IO;
RAMStringBody[sizeof(RAMStringBody)-5]
= '0' + BUTTON3_IO;
SMTPClient.Server.szROM = (ROM
BYTE*)"mail.mcelectronics.com.ar";
SMTPClient.ROMPointers.Server = 1;
SMTPClient.Username.szROM = (ROM
BYTE*)"info@mcelectronics.com.ar";
SMTPClient.ROMPointers.Username = 1;
SMTPClient.Password.szROM = (ROM
BYTE*)"123456";
SMTPClient.ROMPointers.Password = 1;
SMTPClient.To.szRAM = RAMStringTo;
SMTPClient.From.szROM = (ROM
BYTE*)"\"SMTP Service\" <info@mcelec-
tronics.com.ar>";
SMTPClient.ROMPointers.From = 1;
SMTPClient.Subject.szROM = (ROM
BYTE*)"Mensaje de prueba";
SMTPClient.ROMPointers.Subject = 1;
SMTPClient.Body.szRAM = RAMStringBody;
SMTPSendMail();
MailState++;
}
```

y **SMTPClient.Password** para loguearnos en el servidor seguro. En negrita aparecen los parámetros como el destinatario, los datos del servidor y el asunto del mensaje. Las variables que indican el estado de los pulsadores se llaman **BUTTON?_IO**, donde (?) representa el número de pulsador.

Como dijimos anteriormente, podemos invocar este código dentro del programa principal. Por ejemplo, es posible hacer un **IF** que compare el valor de



temperatura con un umbral y, en caso de superarlo, envíe un mensaje de alerta a la dirección especificada por `RAMStringTo[]`.

Puente Serie-Ethernet

Mediante este puente podemos actualizar sistemas que originalmente no tenían conexión a **Ethernet**. Por ejemplo, podemos monitorear celdas de comunicaciones no atendidas desde de Internet.

Las variables que indican el estado de los pulsadores se llaman BUTTON?_IO

Como la celda reporta su estado a través del puerto serie (utilizando RS232) es posible conectarle un webserver configurado como puente serial-Ethernet y, así, monitorear y controlar la celda a través de internet. Vemos en la **Figura 23** cómo activar la función **Serial to Ethernet Bridge** desde el TCP/IP Configuration Wizard.

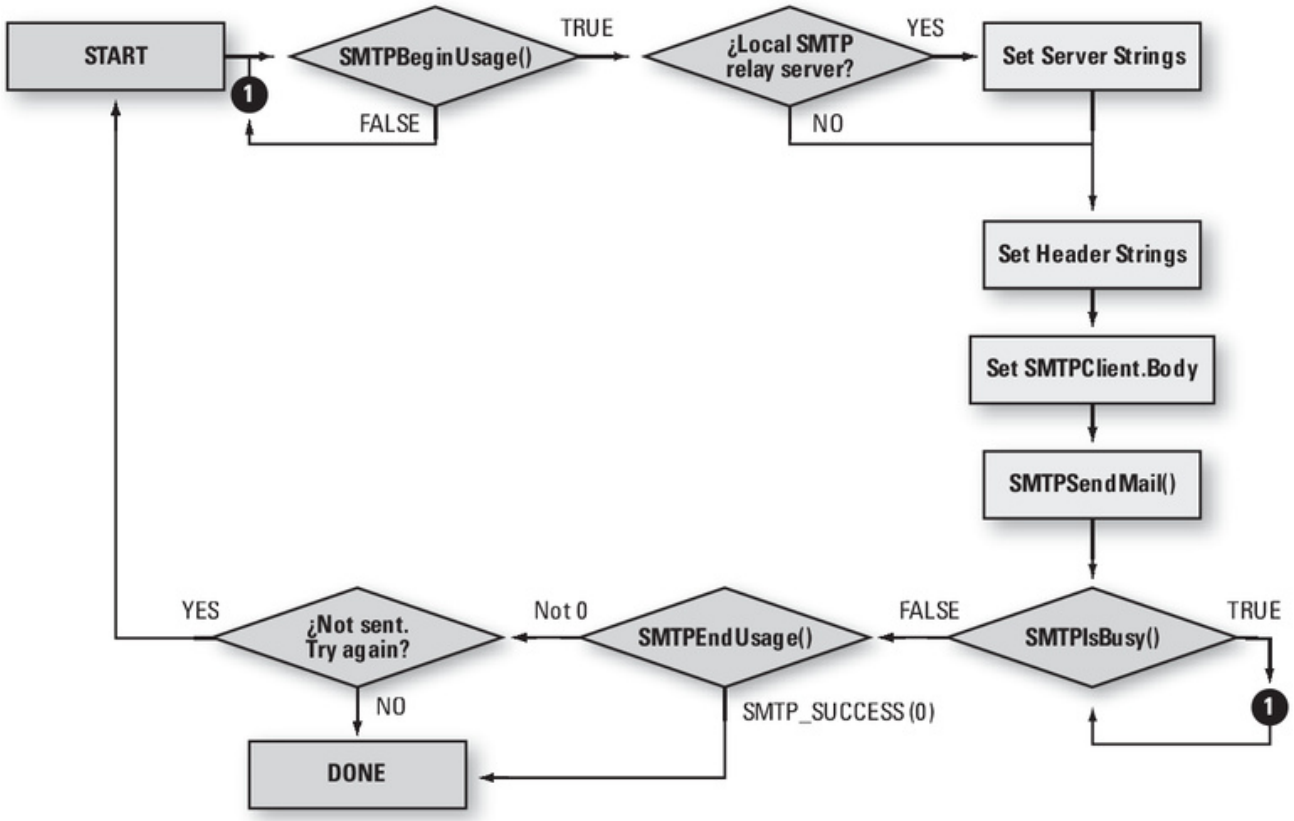


FIGURA 22. **Funciones del stack TCPIP para envío de e-mails.**

Este ejemplo acepta conexiones en el **puerto 9761**. Una vez que establecemos la comunicación a través de un cliente **telnet**, todos los datos recibidos serán retransmitidos por la **UART** del **PIC** y los datos recibidos desde la UART serán enviados al cliente por Ethernet.

De esta forma, siguiendo el ejemplo anterior, conectamos el webserver al puerto serie de la celda de comunicaciones y podemos monitorearlo desde un cliente telnet a través de Internet.

Recordemos que **cliente telnet** es un programa que nos permite realizar conexiones a determinados equipos, como el WebServer. Necesitamos ingresar la dirección IP y el número de puerto.

El **cliente puTTY** se descarga desde **www.putty.org**. Es un archivo **.EXE**, que simplemente, se ejecuta y en el cual debemos ingresar la **IP** del **WebServer** y el puerto al cual queremos acceder, en este caso el 9761. Una vez establecida la comunicación con el WebServer, nos pide usuario y contraseña, por defecto son **admin** y **microchip**, respectivamente. Ahora estamos en condiciones de recibir los datos

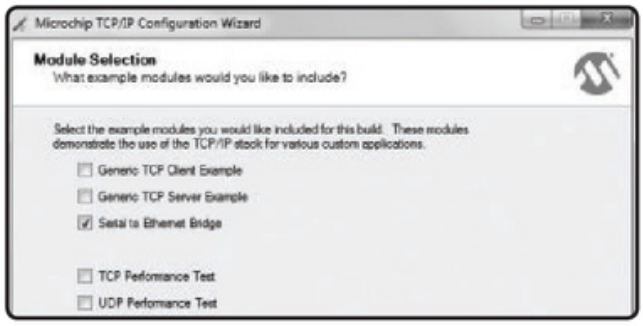


FIGURA 23. **Seleccionamos la opción para habilitar el** `puente Serie - Ethernet` **y así poder controlar la celda desde Internet.**

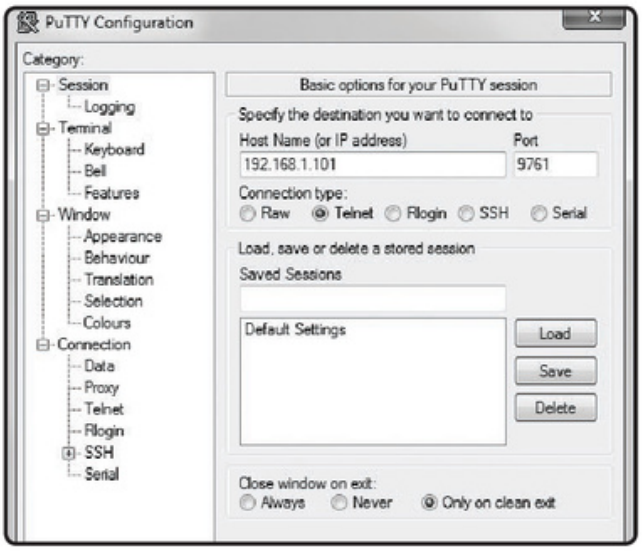


FIGURA 24. **Cliente telnet PuTTY configurado para conectarse al puerto 9761 del WebServer.**

desde el puerto serie, de hecho, podemos conectar un sistema que envie el estado de un sensor por **RS232** y lo veremos en la pantalla del **puTTY** a través de Ethernet. También es posible enviar comandos mediante el puTTY y controlar desde Ethernet el dispositivo serie (**Figura 24**).

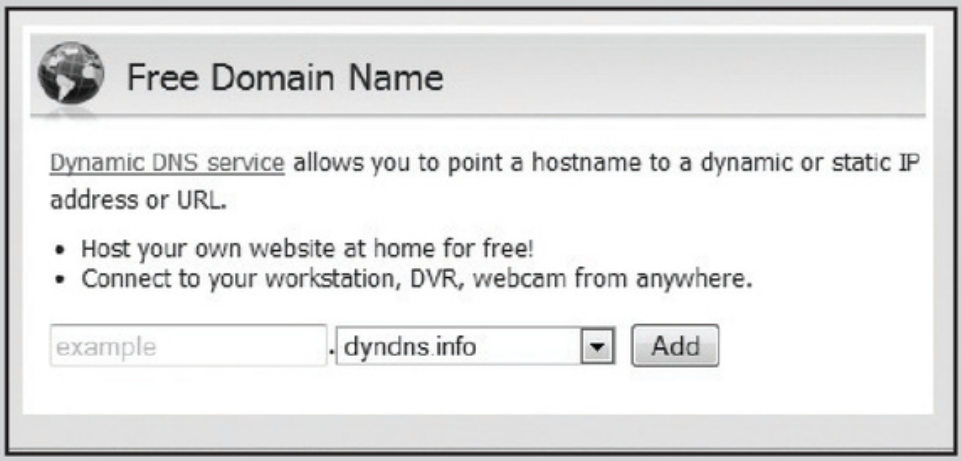
ACCESO DESDE INTERNET

Una de las preguntas más recurrentes es cómo acceder al Web Server con PIC a través de Internet. Si bien podemos hacerlo desde la red local mediante la dirección IP privada (por ejemplo, 192.168.1.101), esto no es posible desde Internet, a menos que contratemos una IP fija a nuestro ISP.

Para la mayoría de los usuarios, la IP es dinámica –cambia dentro de un rango al conectarnos–. Para solucionar este tema, debemos asociarle un DDNS (Dynamic DNS) o DNS dinámico, el cual aplica un nombre de dominio a nuestra IP pública, como **www.mcewebserver.com.ar**. Ver el **Paso a paso 1**.

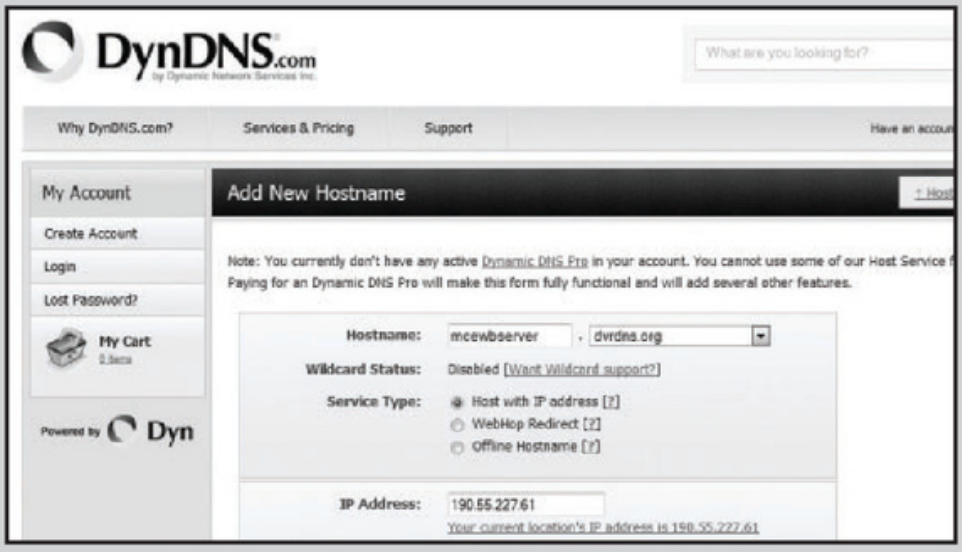
PASO A PASO /1
Acceso desde Internet

1



Una opción para vincular nuestra IP pública a un nombre de dominio es abrir una cuenta en **www.dyndns.org** y configurar el router para que actualice la base de datos de dyndns cada vez que la IP pública cambie. Para registrarnos en **dyndns**, hacemos clic en **Join NOW** desde la pantalla inicial y seleccionamos la opción gratuita.

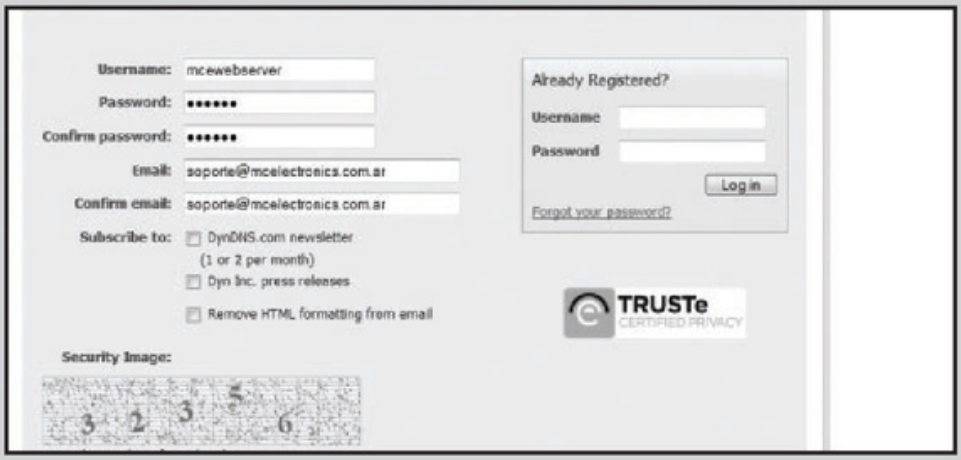
2



Estos son los datos que debemos completar: nombre de host y dirección IP actual (ingresamos la que nuestro ISP nos está asignando es este momento). El nombre de host puede ser cualquiera que no haya sido previamente registrado en dyndns.

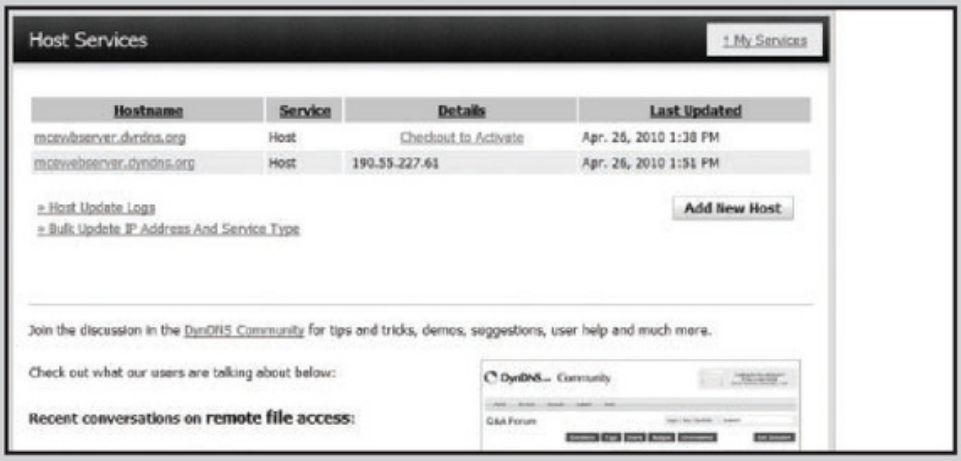
PASO A PASO /1 (cont.)

3



Para finalizar, hacemos clic en **Next**. El asistente nos pedirá un nombre de usuario y una clave para acceder a nuestra cuenta. Luego, utilizaremos estos datos para configurar el router, al que va a estar conectado nuestro servidor web. Ya tenemos el host creado en dyndns; el nombre es **mcewebserver.dyndns.org**.

4



Vemos el host creado recientemente y la dirección IP actual. Si hacemos clic sobre el nombre de host, podemos configurar el TTL (tiempo de vida), parámetro que hace referencia al tiempo durante el cual el host permanece vinculado a la IP en el servidor antes de actualizar la información. Vamos a seleccionar un TTL de 4 horas.
En este caso, dyndns revisará nuestra IP cada 4 horas y actualizará su base de datos.

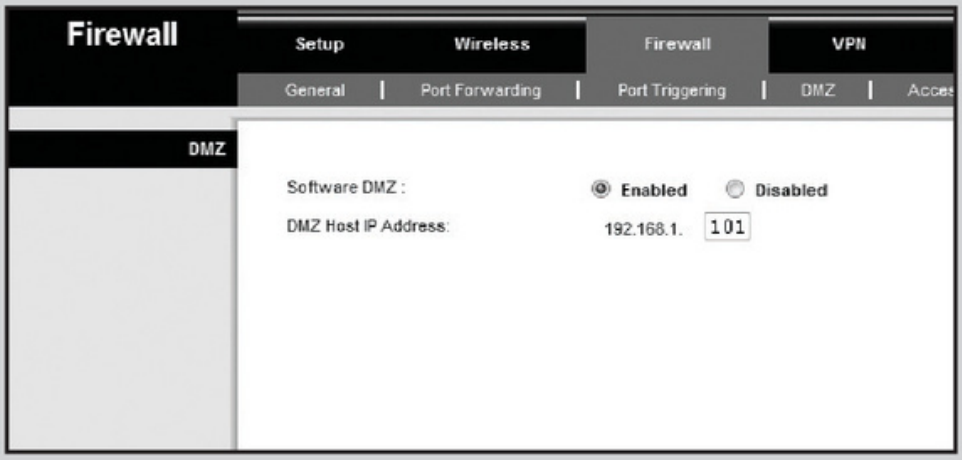
PASO A PASO /1 (cont.)

5



Configuramos el router para que esté vinculado a la cuenta de dyndns y actualice la IP en la base de datos automáticamente. Entonces, vamos a **Setup/DDNS** e ingresamos los datos del host creado: usuario y clave. Ya tenemos acceso al router desde Internet mediante un DNS.

6



Faltaría tener acceso al Web Server, para lo cual vamos a habilitar la función **DMZ** (*Demilitarized Zone*), que expone todos los puertos de una determinada dirección IP a Internet. En este caso, lo hacemos para 192.168.1.101, que es la IP asignada por el router al Web Server. Al acceder a **http://mcewebserver.dyndns.org**, seremos redirigidos a 192.168.1.101 en nuestra red local.

Multiple **choice**

▶1 ¿Cuánta memoria de programa necesita, aproximadamente, Microchip TCP/IP Stack?

a- 2 k.
b- 8 k.
c- 16 k.
d- 32 k.

▶2 ¿Cuál de las siguientes aplicaciones nos permite modificar los parámetros del stack en forma gráfica?

a- MLAB.
b- Microchip Libraries.
c- Microchip TCP/IP Configuration Wizard.
d- Microchip Web Preview.

▶3 ¿Cuál de las siguientes aplicaciones nos permite agregar funciones especiales?

a- MLAB.
b- Microchip Libraries.
c- Microchip TCP/IP Configuration Wizard.
d- Microchip Web Preview.

▶4 ¿Cuál de las siguientes aplicaciones nos permite visualizar la página web antes de cargarla en la memoria del PIC?

a- MLAB.
b- Microchip Libraries.
c- Microchip TCP/IP Configuration Wizard.
d- Microchip Web Preview.

▶5 ¿Cómo se llama la página principal por defecto del Web Server?

a- paginaprincipal.htm
b- principal.htm
c- index.htm
d- Ninguna de las anteriores.

▶6 ¿Qué debemos habilitar para enviar e-mails desde el stack TCP/IP?

a- El servidor FTP.
b- El cliente POP.
c- El cliente SMTP.
d- El servidor HTTP.

Respuestas: 1-a, 2-c, 3-c, 4-d, 5-c, 6-c.

Capítulo 4

Telemetría y control remoto Ethernet con PIC



Aprenderemos a realizar algunas aplicaciones para los sistemas de monitoreo y control a distancia.

RedUSERS.com

Arquitectura del PIC18F97J60

Para alojar el stack TCP/IP en un microcontrolador, precisamos que este sea muy potente y tenga suficiente memoria de programa. Es así como se desarrolló el **PIC18F97J60**, para que un sistema dedicado al monitoreo y control pueda tener una comunicación con la red Ethernet, y evitar los incómodos controladores de red externos, que consumían muchas líneas de datos para el flujo de información.

Las características del PIC18F97J60 son las siguientes:

- Rango de alimentación: 2,35 V a 3,6 V (3,1 V a 3,6 V utilizando el módulo Ethernet).
- Regulador interno de 2,5 V para alimentar al núcleo del procesador.
- 128 KB de memoria de programa flash.
- 3 KB de memoria de datos RAM.
- Encapsulado SMD TQFP de 100 pines.
- 70 pines I/O digitales.
- Controlador Ethernet compatible con el estándar IEEE 802.3.
- Compatible para redes 10/100/1000BaseT.
- MAC integrada.

Estas especificaciones hacen del PIC18F97J60 un microcontrolador poderoso, con capacidad de conexión a una red, al mismo tiempo que conserva la arquitectura de los PIC18F que estudiamos en esta obra.

ALIMENTACIÓN

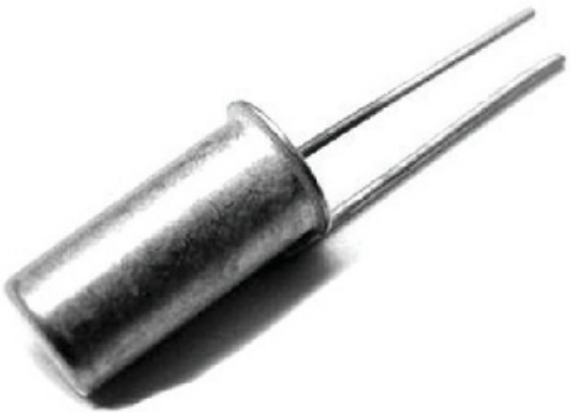
El PIC18F97J60 debe ser alimentado con una fuente capaz de entregar entre 2,35 V y 3,6 V; si deseamos utilizar el módulo Ethernet, tenemos que restringir el rango de alimentación entre 3,1 V y 3,6 V. Con estos valores, el fabricante garantiza el funcionamiento correcto del dispositivo. Además, el núcleo de procesamiento de este microcontrolador trabaja a 2,5 V.

Es importante aclarar que el PIC cuenta con un regulador interno que solo necesita un capacitor externo en el pin Vddcore/Vcap de 10 µF con un bajo ESR (resistencia serie equivalente), como los que proporcionan los capacitores de tantalio (**Figura 1**).

En caso de alimentar el microcontrolador con 3,3 V, debemos colocar el pin ENVREG a un nivel alto para habilitar el regulador interno.

PUERTOS I/O

Este microcontrolador dispone de nueve puertos de entrada y salida digital, enumerados de la **A** a la **H** y luego el **J** (**no existe puerto I**); es decir PORTA, PORTB, etc. Todos son de 8 pines, excepto el PORTA, que tiene 6. Los puertos poseen diferentes drivers para manejar distintos niveles de potencia. Entonces, los B y C pueden manejar cargas mayores, como la de un LED; en tanto que los D, E y J trabajan con cargas medias, como la interfaz a memoria externa;



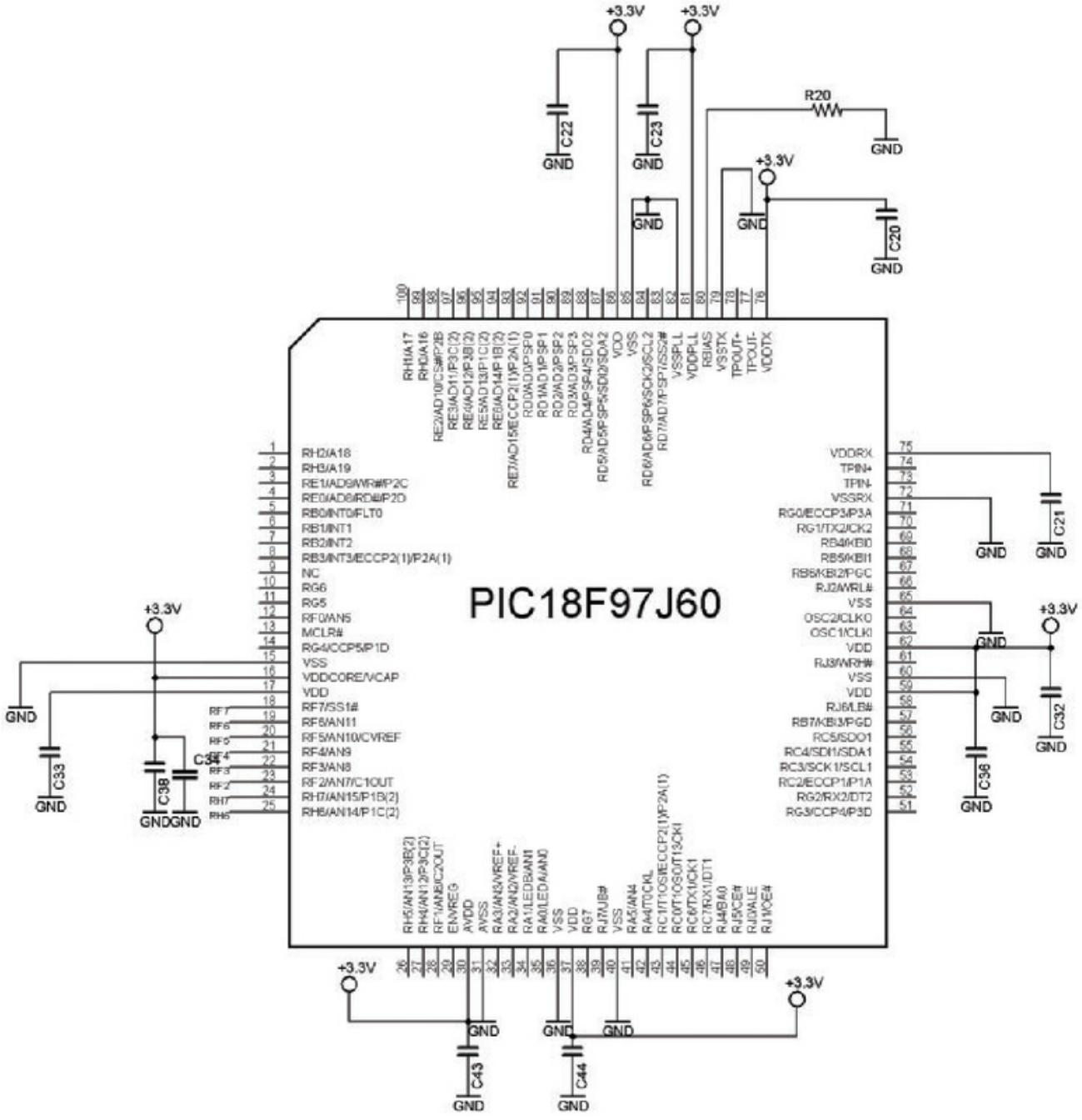


FIGURA 1. Esquema con las alimentaciones generales del PIC. Se colocaron capacitores de filtro en los pines de alimentación y el capacitor externo para el regulador interno.

y los A, F, G y H lo hacen con cargas mínimas y pueden utilizarse como indicadores de estados. Todos los puertos tienen los registros PORT, LAT y TRIS para su control y configuración. Los A y F pueden usarse, además, como canales de entrada al conversor A/D.

MÓDULO ETHERNET

La familia de PICs a la que pertenece este microcontrolador cuenta con controladores Ethernet incorporados que soportan el estándar **IEEE 802.3** para una conexión 10-BaseT

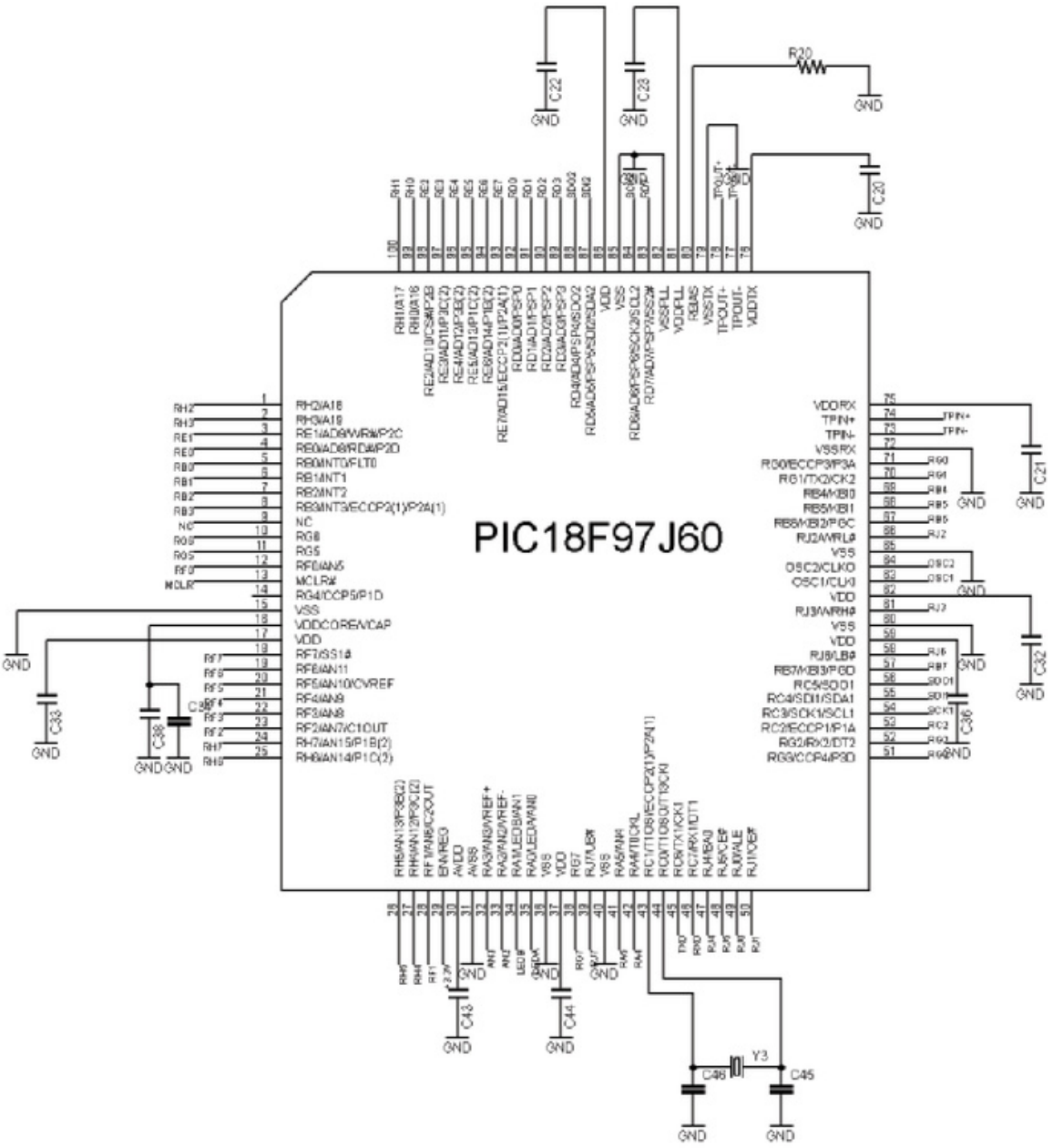


FIGURA 2a. **Conexión del PIC a un conector RJ-45 que posee bobinas en su interior.**

con par trenzado. Incluyen implementaciones completas para el control de acceso al medio (MAC) y el transceptor de la capa física (PHY).

Además, este módulo dispone de un buffer de 8 K de memoria RAM para almacenar los paquetes de datos recibidos y los que serán transmitidos.
La conexión física se realiza a través de 4 pines dedicados a la recepción y la información de datos, ellos son:

- TPIN+
- TPIN-
- TPOUT+
- TPOUT-

En la **Figura 2** podemos ver el esquemático correspondiente. Lo único que se necesita de manera externa son unos transformadores, pero en la actualidad encontramos conectores RJ-45 que los tienen en su interior.

Este PIC cuenta con controladores Ethernet incorporados que soportan el estándar IEEE 802.3

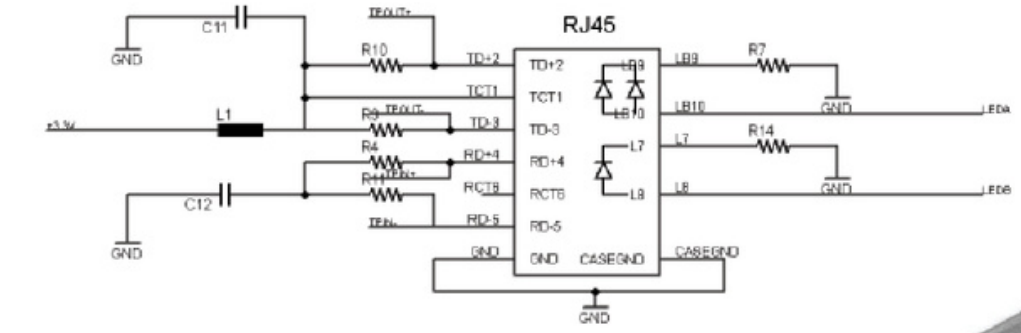


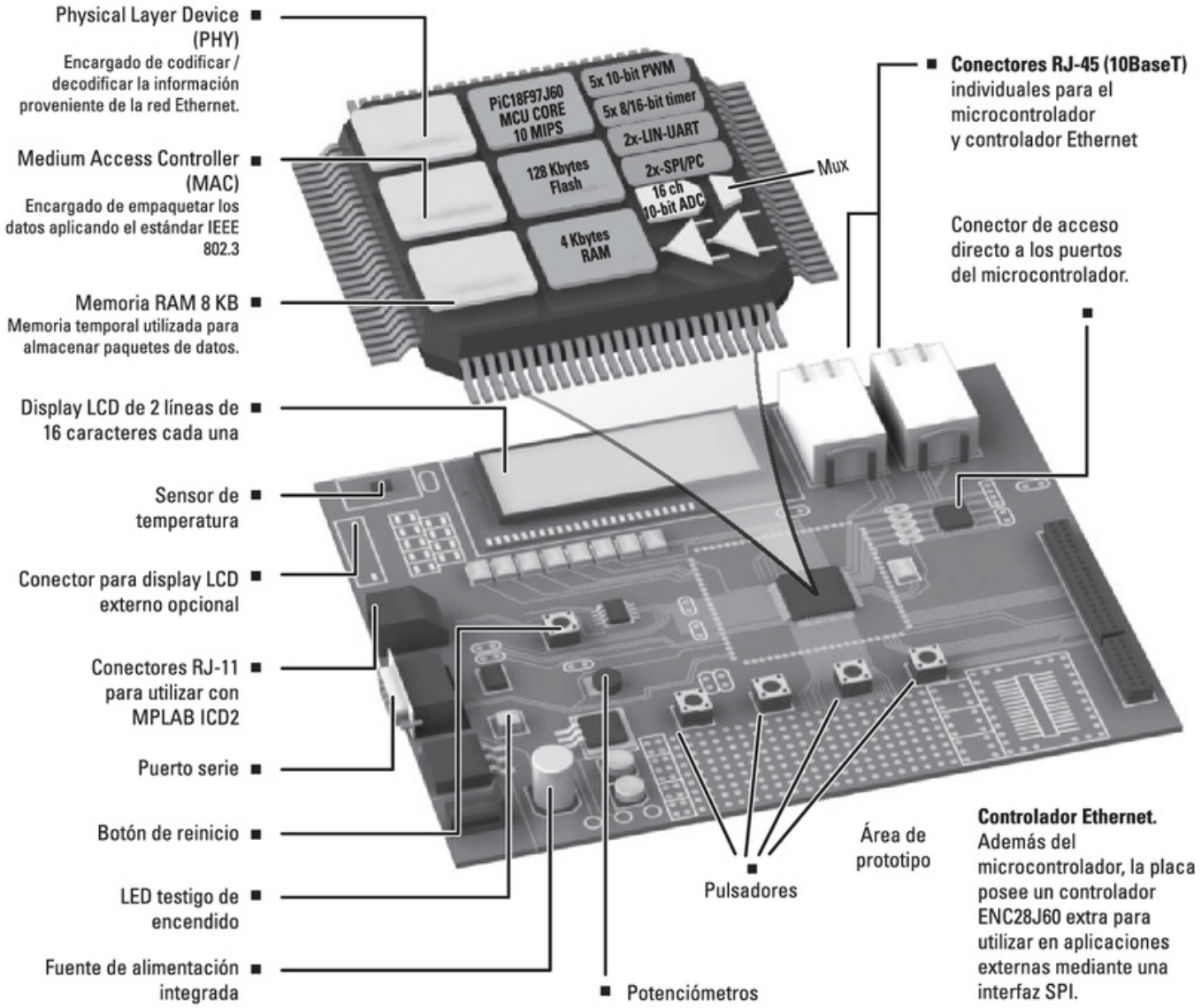
FIGURA 2b. **Continuación del diagrama.**

INFOGRAFÍA 1: ETHERNET Y TCP/IP CON PIC18F97J60

Tenemos a nuestra disposición una amplia gama de herramientas y dispositivos para desarrollar completas soluciones de conectividad vía Ethernet y TCP/IP. En esta ocasión veremos una tarjeta de desarrollo basada en PIC18F97J60.

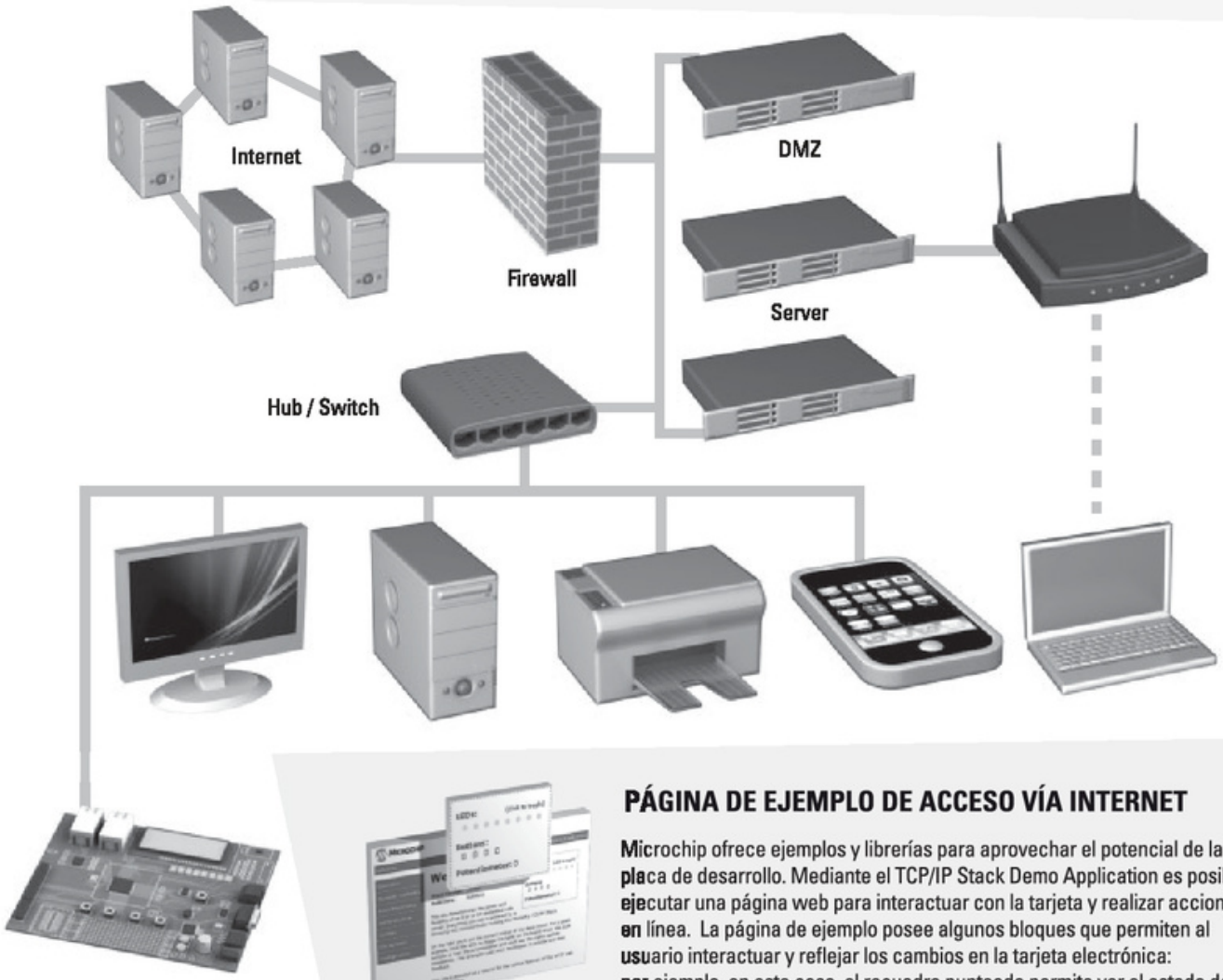
MICROCONTROLADOR PIC18F97J60

Es el corazón de la placa. Permite controlar, transmitir y recibir datos vía Ethernet. Funciona en base a un reloj de 25 MHz.



APLICACIONES

La tarjeta de desarrollo que ofrece Microchip permite realizar gran cantidad de soluciones de conectividad vía Ethernet o Internet. Es posible montar un servidor web para realizar numerosas tareas, como: monitoreo de señales y variables, activación de equipos a distancia, y registro de eventos y sucesos, por ejemplo.



PÁGINA DE EJEMPLO DE ACCESO VÍA INTERNET

Microchip ofrece ejemplos y librerías para aprovechar el potencial de la placa de desarrollo. Mediante el TCP/IP Stack Demo Application es posible ejecutar una página web para interactuar con la tarjeta y realizar acciones en línea. La página de ejemplo posee algunos bloques que permiten al usuario interactuar y reflejar los cambios en la tarjeta electrónica: por ejemplo, en este caso, el recuadro punteado permite ver el estado de los LEDs y cambiarlo. Lo mismo sucede para los botones. La opción del potenciómetro muestra un valor numérico de acuerdo con la posición del eje (el cual se puede variar).

Telemetría

Como parte de las tareas de automatización, es importante realizar el monitoreo de variables para que el sistema pueda compensar las variaciones no deseadas de estas magnitudes; muchas veces, resulta de utilidad hacerlo a distancia. En las industrias químicas o en domótica (casas inteligentes), por ejemplo, el monitoreo de variables a distancia es muy beneficioso, porque permite tener un punto de control central de diferentes procesos y mantener así un flujo eficiente del procesamiento de datos.

Una posibilidad de la telemetría es enviar los datos necesarios para el monitoreo de un proceso utilizando una red Ethernet (**Figura 3**). Y aquí la pregunta que surge es: ¿por qué Ethernet? Simplemente, porque en la actualidad, es la red más utilizada, presente en muchas oficinas y edificios industriales, porque está basada en estándares **IEEE 802.3**, que garantizan la fiabilidad de las conexiones de red y las transmisiones de datos. Además, el uso de Internet permite alcanzar grandes distancia entre la estación de monitoreo y el operador (**Figura 4**).

TELEMETRÍA DE VARIABLES ANALÓGICAS

Los sistemas embebidos o dedicados permiten el monitoreo a distancia de variables analógicas –como temperatura, presión o caudal– gracias a que poseen conversores A/D que digitalizan la señal. El sistema luego encapsula el paquete de datos TCP/IP y lo envía por la red Ethernet, que puede ser una LAN local o Internet, a través de un proveedor del servicio (**ISP**). El microcontrolador PIC18F97J60 es ideal para estas aplicaciones, ya que incorpora un conversor A/D y tiene el stack TCP/IP cargado en su interior.

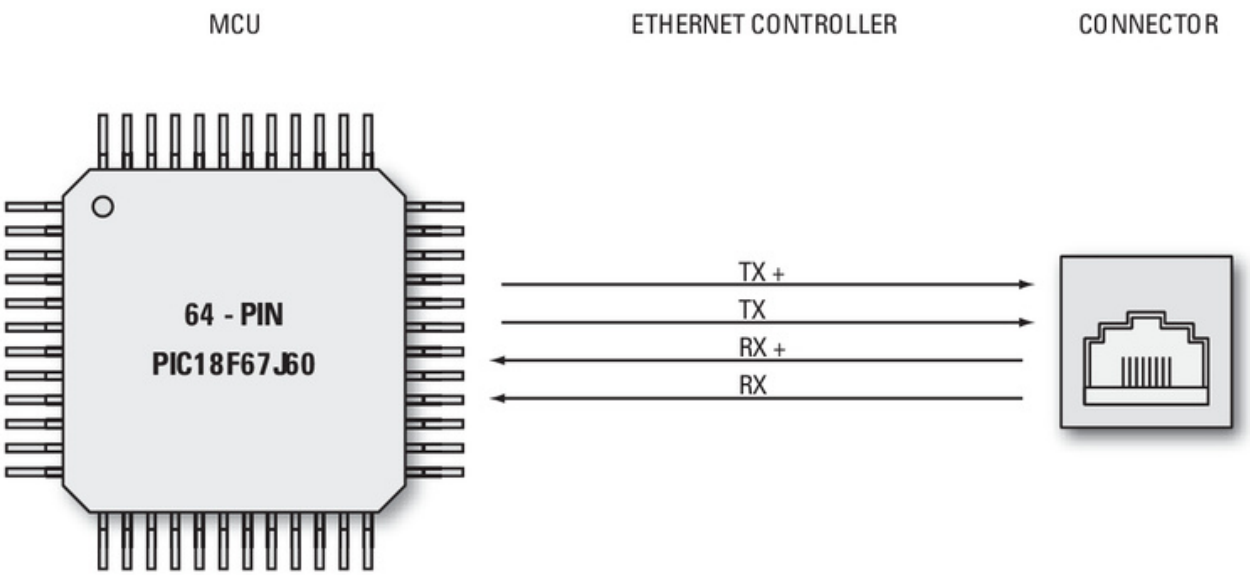


FIGURA 3. **Usando los microcontroladores PIC con** módulos Ethernet **incluidos, no tenemos necesidad de recurrir a módulos que realicen la interfaz entre ambos.**

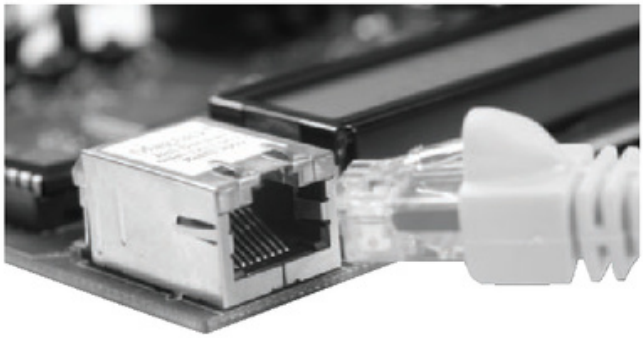


FIGURA 4. **Gracias a la red de redes, Internet, podemos efectuar el monitoreo de variables a grandes distancias.**

VARIABLES DINÁMICAS

Las variables dinámicas permiten que el servidor web tome datos del sistema, como puede ser el valor de un sensor o de la memoria de programa; luego, combinados con una plantilla de una página web, es posible mostrar esos valores en la pantalla de una PC remota.

La página web finalizada se transmite a través de una red o de Internet, y los datos del sistema se presentan en la pantalla del usuario. Por ejemplo, la **Figura 5** indica cómo el sistema toma el dato de una temperatura, el cual, combinado con la plantilla de una página web, se muestra en pantalla. Con estas variables, podemos insertar cadenas de textos en las páginas web y generar vistosos gráficos. Esto nos ayuda a diseñar el formato y los elementos necesarios de una página.

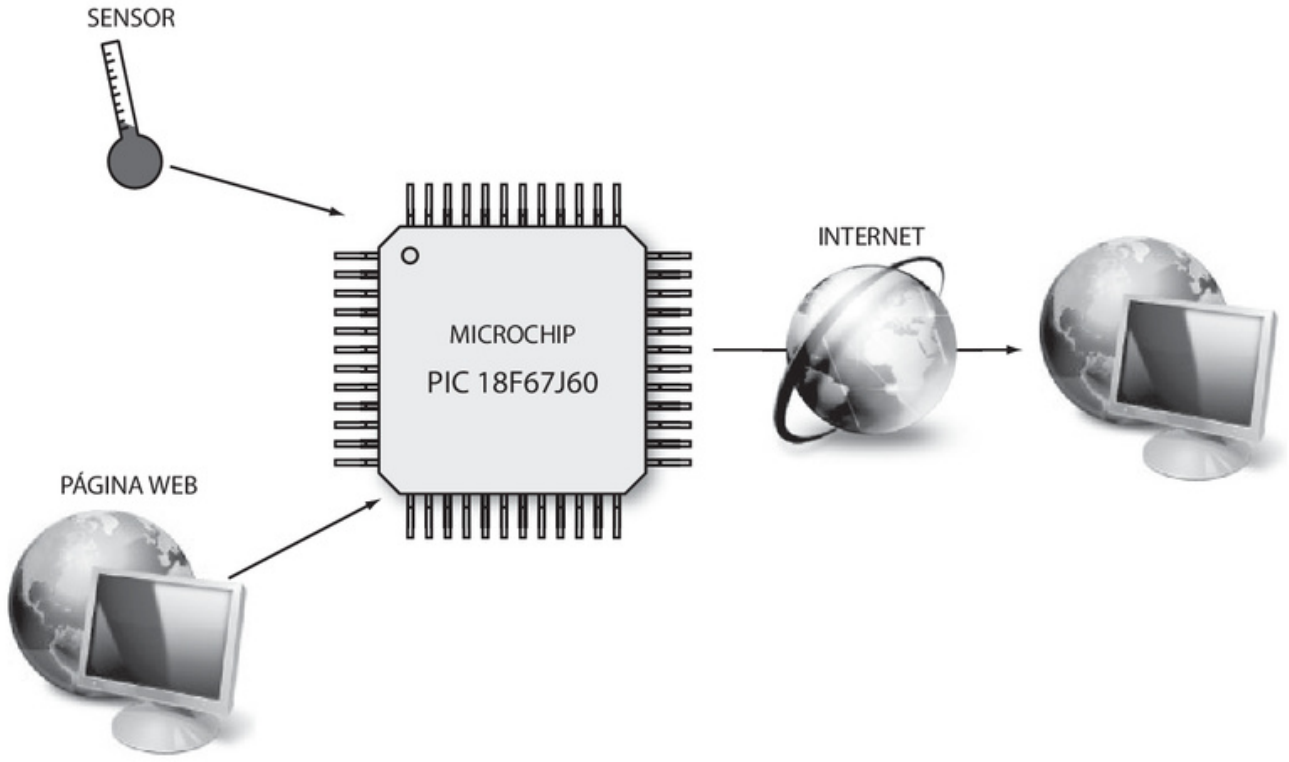


FIGURA 5. **Aquí vemos cómo actúa una variable dinámica. El sistema se encarga de combinar el dato con ella y de enviar la página completa por la red.**

Cómo insertar una variable dinámica

Para insertar una variable dinámica en nuestro proyecto, debemos realizar dos pasos. Primero, colocar el nombre de dicha variable dentro del código de la página en HTML. Luego, implementar la función que tendrá el control sobre esa variable dinámica.

Para realizar el primer paso, debemos contar con una plantilla o una página web a la que le colocaremos una variable dinámica. Estas variables se identifican en el código HTML porque están escritas entre dos caracteres "tilde" (~). Entonces, si queremos agregar una llamada "**VarDin**", incluimos en el código la expresión **~VarDin~**.

Cuando el servidor web se encuentra con esa variable dinámica, ejecuta un llamado a una función **callback**, que debe estar implementada en el código en C, más precisamente, en el archivo **CustomHTTPApp.c** explicado en la clase anterior. La función debe tener el siguiente formato:

```
Void HTTPPrint_VarDin (void)
{
        // Sentencias
}
```

Esta función será llamada cada vez que el servidor web se encuentre con la variable dinámica **VarDin**.

Cómo utilizar una variable dinámica

Supongamos que debemos supervisar distintas máquinas expendedoras, y nuestro trabajo es asegurarnos de que estas no se queden sin el producto que venden. Esta sería una ardua labor si tuviéramos que revisar las máquinas todos los días. Por eso, sería mucho más sencillo monitorear el estado de cada una desde la PC de nuestro escritorio.

Esta tarea puede realizarse por medio de variables dinámicas que nos muestren información, como el nombre de la máquina, su ubicación y la cantidad de productos que tiene almacenados a la espera



CUSTOMHTTPAPP.C

Cuando descargamos la librería de Microchip con el stack TCP/IP, podemos encontrar el archivo CustomHTTPApp.c en el directorio TCPIP Demo App. Éste se utiliza para definir las funciones callback que manejan las variables dinámicas.

de ser vendidos. Entonces, si queremos ver, por ejemplo, el nombre de la máquina o su identificación cuando navegamos por la página del servidor web, creamos una variable dinámica **~hostname~** en algún lugar de la plantilla de la página web alojada en el servidor (imaginemos que un desarrollador de páginas web armó esta plantilla para nosotros).

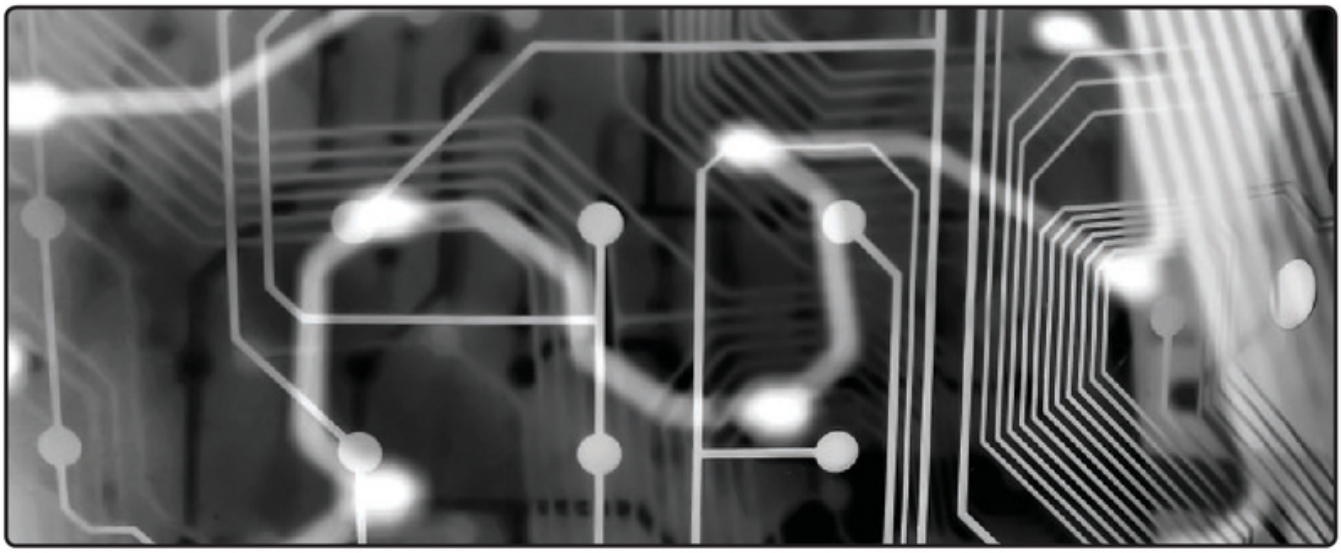
Para el ejemplo, dentro del archivo **index.htm** colocamos el siguiente código:

```
<div id = "locations">
Maquina ~hostname~
</div>
```

Aquí podemos observar que después de la palabra **máquina**, sigue el nombre de la que corresponde.

Luego, escribimos la función que será llamada por el servidor cada vez que encuentre esta variable, recordemos que esta se escribe dentro del archivo **CustomHTTPApp.c**.

Las variables dinámicas permiten que el servidor web tome datos del sistema



SOBRE CALLBACK

Las funciones `callback` deben contener el nombre y el valor de la variable que se define. Además, es necesario tener en cuenta que los datos enviados desde el servidor web tienen que estar codificados en código ASCII. Si pasamos esto por alto, visualizaremos errores en nuestra página.

```
Void HTTPPrint_hostname (void)
{
       TCPPutString( sktHTTP, Maquina_
Name);
}
```

Esta función se encarga de colocar el nombre de la estación dentro de la página web, reemplazando el valor de la variable dinámica y mostrándola en pantalla. En el nombre de la función podemos ver cómo, luego de **HTTPPrint_**, se debe acoplar el nombre de la variable dinámica insertada en la página.

Dentro de la función, enviamos una cadena de caracteres sobre un segmento TCP con el nombre de la estación, que en esta oportunidad es almacenada en una variable que podemos definir al comienzo del código (Figura 6).

Parámetros de las variables dinámicas

También es posible transmitir parámetros junto con las variables dinámicas, y mostrarlos en la página. Esto resulta muy útil cuando tenemos que enviar un vector o un array de datos a la página del servidor. Veamos cómo agregar los parámetros continuando con el ejemplo anterior.

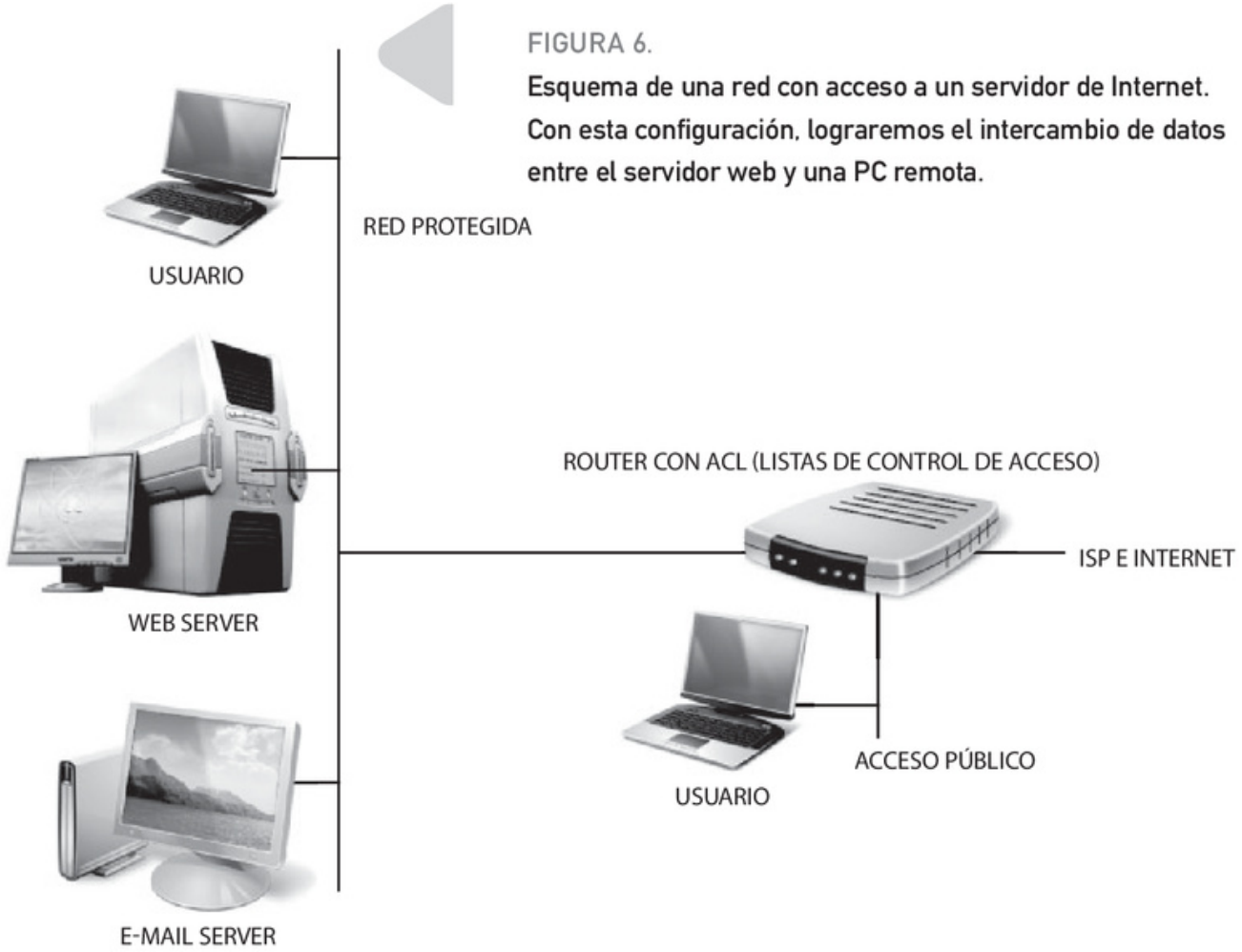


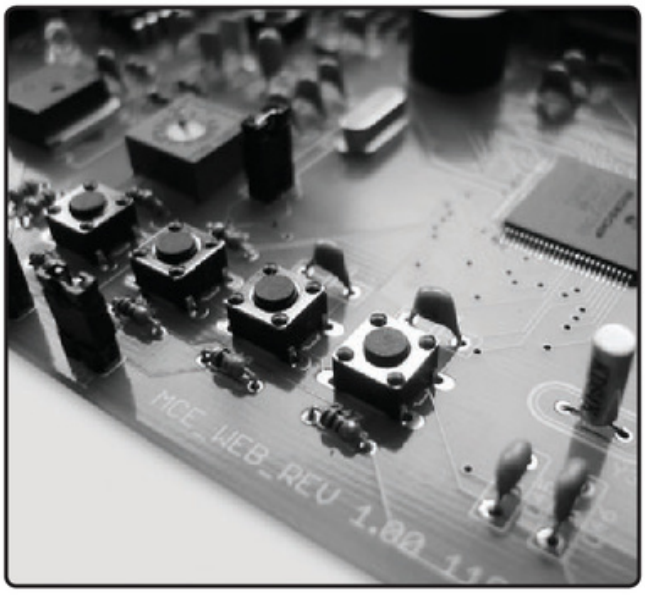
FIGURA 6.
Esquema de una red con acceso a un servidor de Internet. Con esta configuración, lograremos el intercambio de datos entre el servidor web y una PC remota.

Supongamos que cada estación contiene siete productos para vender, por lo que la información que nos interesara mostrar es el nombre del producto y la cantidad de cada uno. Entonces, en la página vamos a encontrarnos con siete bloques del código que se muestra a continuación por cada producto.

```
<div class="productname">~name(0)~:</
div>
<div class="stock">~stock(0)~</div>
</div>
```

Vamos a incluir estos bloques por cada producto que contiene la máquina y a aumentar el índice del array a cada paso. Luego, realizaremos las funciones **callback** en el archivo **CustomHTTPApp.c**.

La ventaja de este procedimiento es que, en vez de escribir siete funciones para la variable **name** y otras siete para **stock**, solo escribiremos dos, y los valores los pasaremos como parámetros.




```
void HTTPPrint_name(WORD item)
{
TCPPutString(sktHTTP, items[item].name);
}

void HTTPPrint_stock(WORD item)
{
BYTE digits[4];
uitoa(items[item].stock, digits);
TCPPutString(sktHTTP, digits);
}
```

En cada función colocamos entre paréntesis el parámetro de la variable que tomará un valor de 16 bits, llamado **item**. Luego, una estructura devolverá el nombre del producto en la primera función. Para la segunda función, como queremos enviar un número, debemos convertirlo en una cadena de caracteres ASCII y después enviarla.

MONITOREO DE LA TEMPERATURA POR ETHERNET

En el Paso a paso 1, realizaremos el monitoreo de una variable analógica, como la temperatura de un recinto, desde un lugar remoto, y veremos la variación de esta magnitud en tiempo real. Mostraremos el esquemático y los pasos para asignar una variable dinámica a nuestra página y compilar el proyecto.

Las variables dinámicas son útiles cuando tenemos que enviar un array a la página de un servidor

En la **Figura 7** podemos observar el esquema de la práctica, donde tenemos el microcontrolador conectado a la red a través de un conector RJ-45.

Servidores web dedicados

En la actualidad, muchas industrias tienden a incorporar dispositivos embebidos o dedicados a tareas específicas. A su vez, se trata de que estos puedan comunicarse con otros dispositivos para crear una estructura de supervisión mucho más efectiva. Los servidores web dedicados son un claro ejemplo.

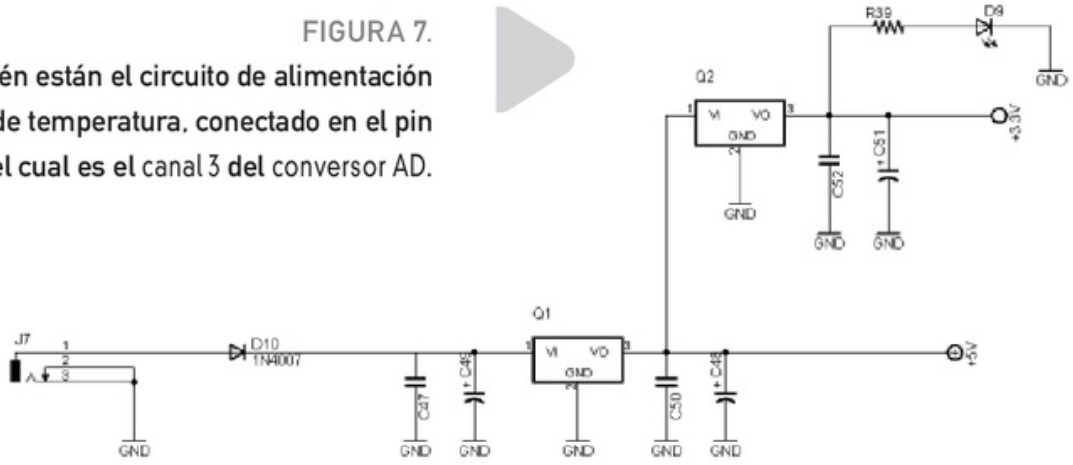
FUNCIONES DE UN SERVIDOR WEB DEDICADO PARA CONTROL

Los sistemas embebidos suelen realizar tres tareas:

- Toman una entrada, que puede ser de un sensor.
- Realizan un proceso en tiempo real.
- Obtienen una salida, que puede ser una respuesta o el control de un actuador.

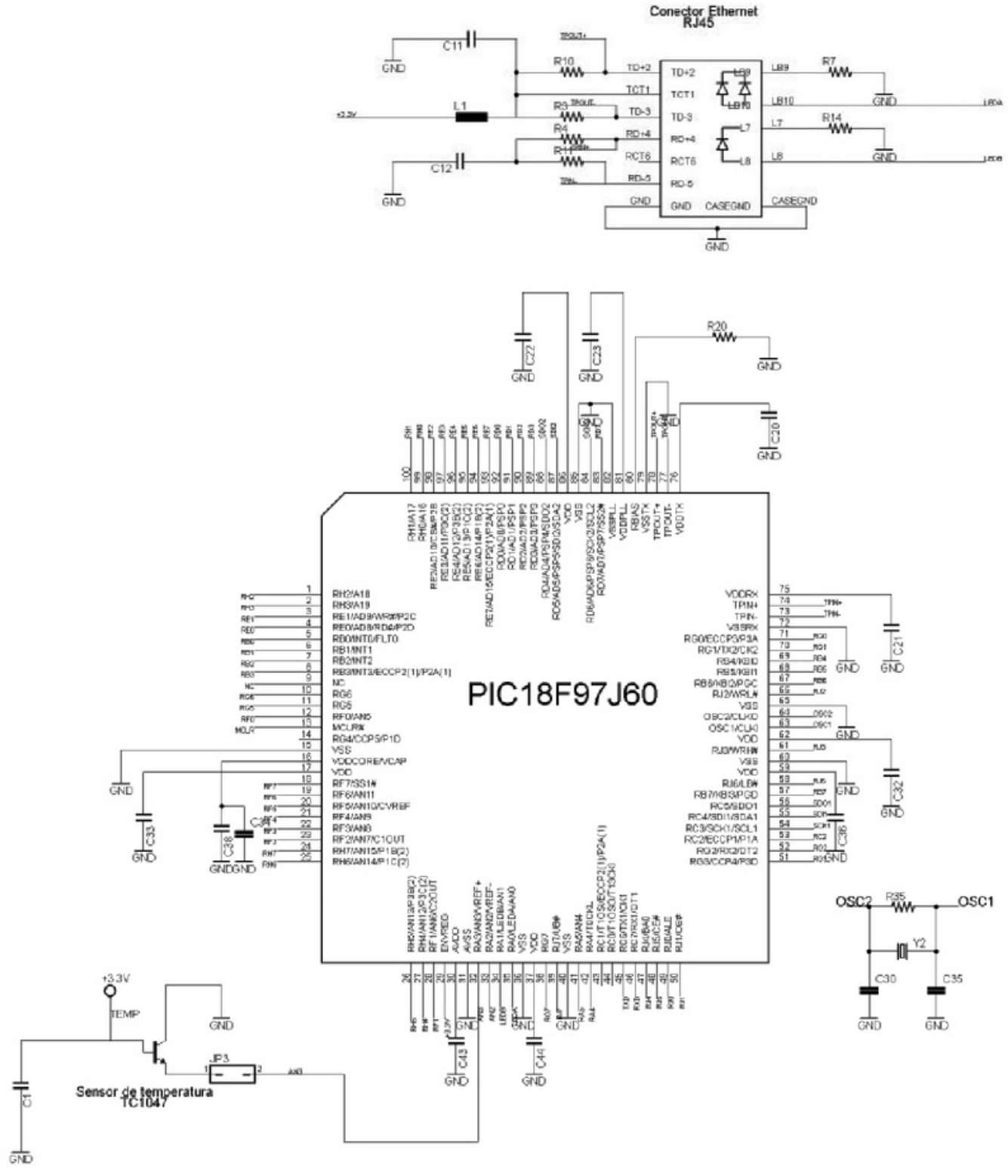
La ventaja de un servidor web es que permite realizar el control de una planta a distancia.

FIGURA 7.
También están el circuito de alimentación y el sensor de temperatura, conectado en el pin RA3/AN3, **el cual es el** canal 3 **del** conversor AD.



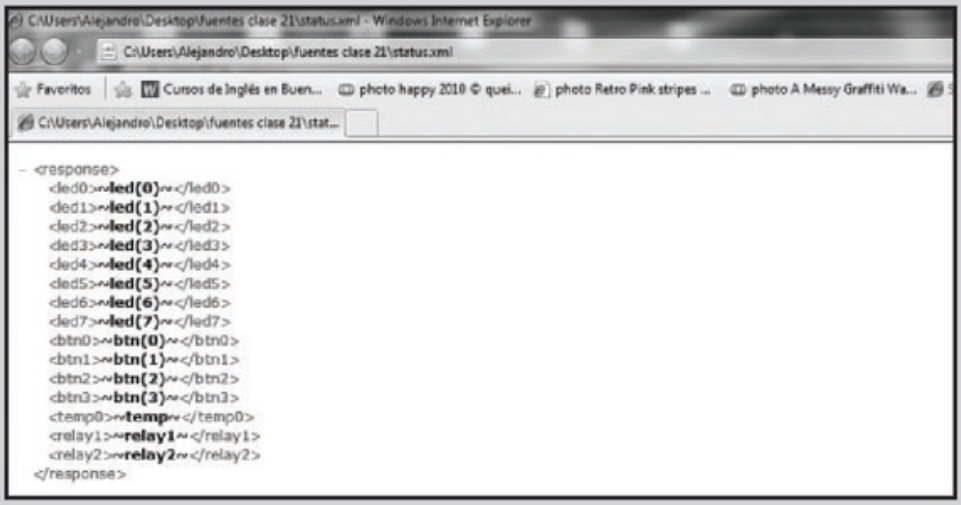
CONVERSIÓN

Para realizar la conversión de números enteros a código ASCII, podemos utilizar las funciones `itoa` o `uitoa` (esta última, para valores sin signo). Ambas se encuentran definidas en la librería estándar de C, dentro de la biblioteca `stdlib.h`.


Conector Ethernet
RJ45
PIC18F97J60
Sensor de temperatura
TC1047

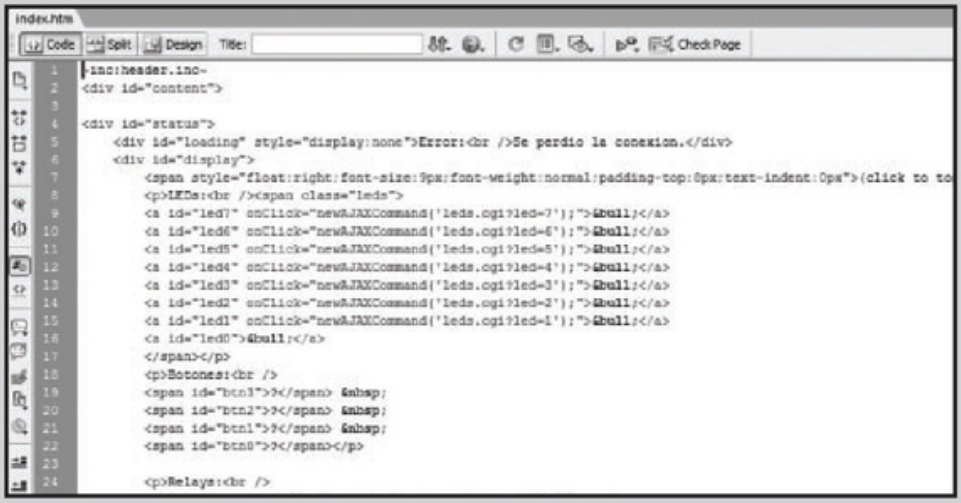
PASO A PASO /1
Monitoreo de la temperatura por Ethernet

1



Comience insertando la variable dinámica, para lo cual proceda como se explicó en la anteriormente. Primero edite el archivo **status.xml**, en el que va a declarar la variable dinámica temp con la siguiente sentencia: **<temp0>~temp~</temp0>**.

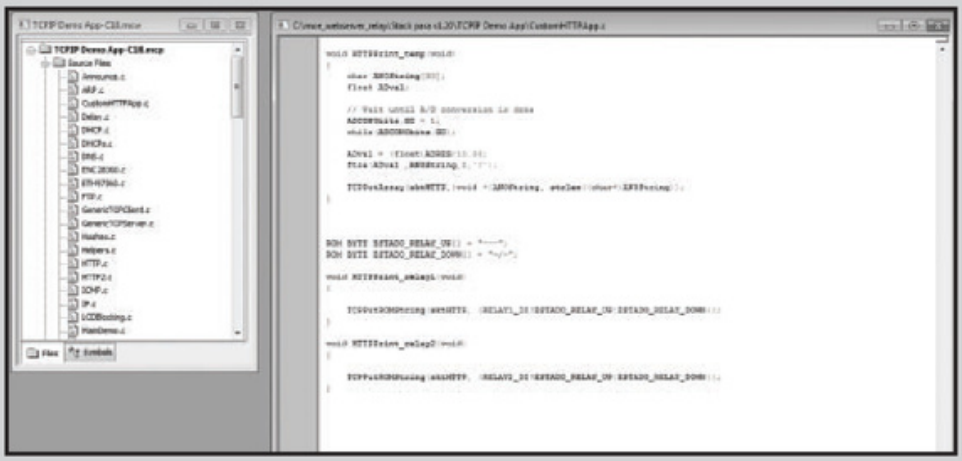
2



Busque la plantilla de la página web **index.html** que provee la librería de Microchip y coloque la siguiente sentencia: **<p>Temp: ?**. Esta reemplazará a la variable dinámica y colocará el valor de la temperatura que arroje la función **callback**.

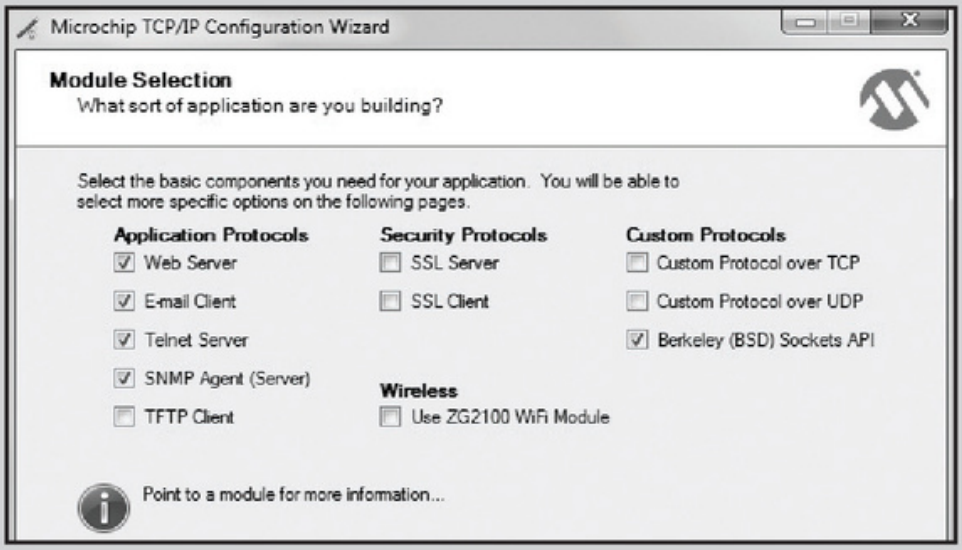
PASO A PASO /1 (cont.)

3



En el archivo **`CustomHTTPApp.c`** escriba la función **`callback`** que controlará la variable dinámica definida. En esta función realice la conversión a digital de la temperatura y llévela a ASCII para mandarla a la página mediante la función **`ftoa`**. Luego, envíe los caracteres con **`TCPPutArray()`**.

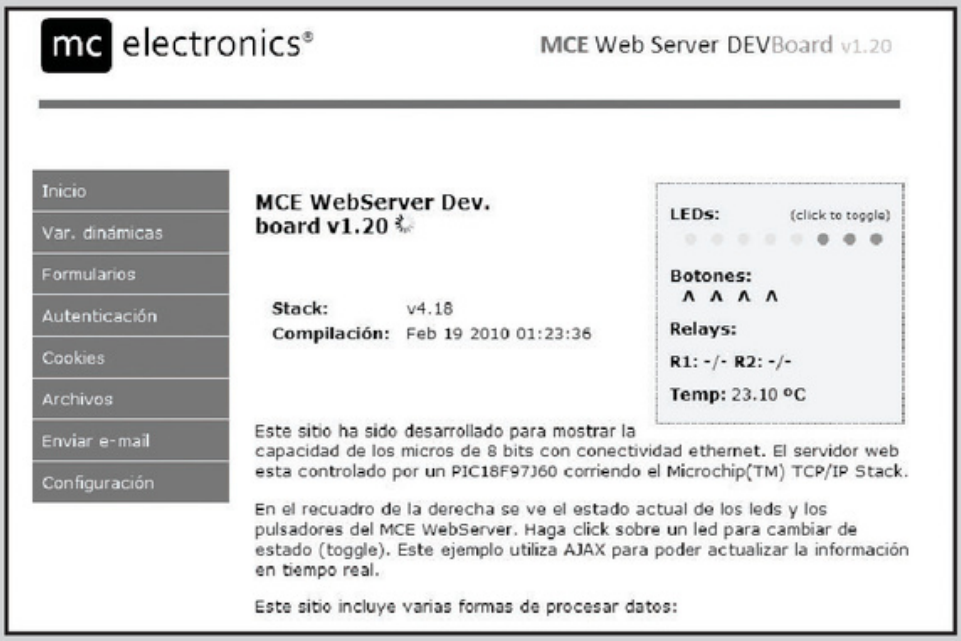
4



Ejecute la utilidad que vimos en el capítulo anterior, **TCP/IP Configurations Wizard**. Configure y compile el proyecto y grábelo al PIC. A continuación, cargue la página con el MPFS en la memoria externa del PIC.

PASO A PASO /1 (cont.)

5



Una vez hecho esto, puede ver la página accediendo desde el browser a la dirección IP del servidor. Observe que la temperatura se actualiza con cada llamada a la función que la controla.

ALTERNATIVAS COMERCIALES

Si necesitamos acortar los tiempos de desarrollo o queremos evitar la complejidad del proceso de armado de un servidor web dedicado, existe la alternativa de comprar módulos de entrenamiento, que proveen las herramientas necesarias para ejercitarse en el manejo del stack TCP/IP de Microchip.

Estos módulos incluyen el hardware necesario para conectarse a una red Ethernet y vienen provistos de un firmware para comunicarse con el módulo a través de una página web. En el mercado encontramos el módulo de entrenamiento **MCE Web Server DEV**, que cuenta con un microcontrolador PIC18F97J60. El PIC está cargado con un programa demostrativo para aprender a utilizar el stack TCP/IP de Microchip y realizar todas las aplicaciones sobre Ethernet con PIC que estudiamos en esta obra.

Además, cuenta con algunos periféricos, como LEDs, display LCD y un sensor de temperatura para interactuar con el PIC (**Figura 8**).

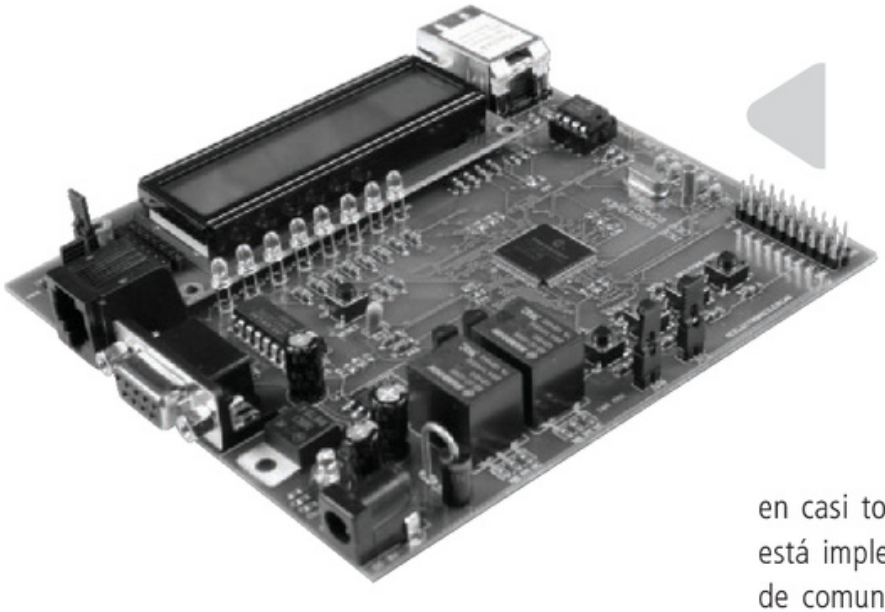


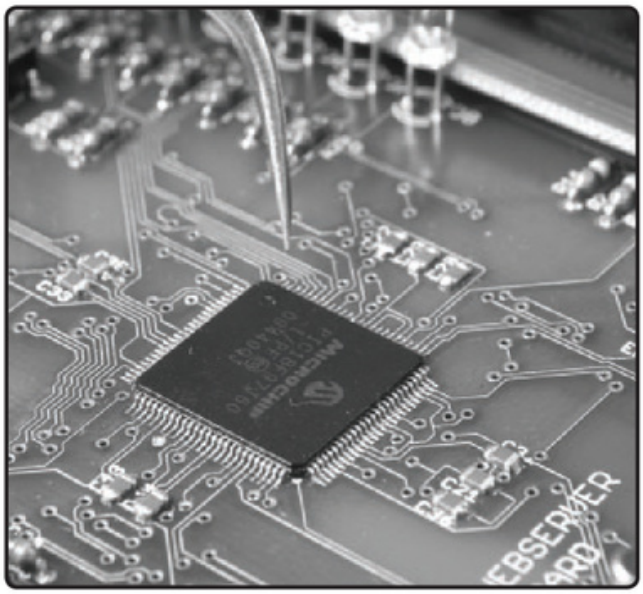
FIGURA 8.
La placa MCE Web Server Dev **es ideal para los desarrollos que necesitan conectividad Ethernet. Más información en:** www.mcelectronics.com.ar.

CONTROL REMOTO

Hasta el momento, vimos cómo monitorear parámetros a distancia; ahora aprenderemos a controlar esas variables de manera remota.

En muchas industrias, es necesario usar algún equipo que permita realizar, por diferentes cuestiones, el control de la planta en forma remota, ya sea porque el lugar no es accesible para el personal de planta o por razones de seguridad.

Existen muchos equipos que permiten realizar esta tarea y que utilizan diferentes protocolos de comunicación. En la actualidad, en diversas industrias y en casi todas las nuevas líneas de producción se está implementando Ethernet industrial como red de comunicación básica, en especial, en aquellos casos en que todo el sistema se controla desde una computadora central.



APLICACIONES COMERCIALES

Algunos módulos solo disponen del hardware necesario para efectuar una conexión a la red; este es el caso del MCE Web Server CORE, que tiene el PIC18F97J60, el circuito de alimentación interno, un conector para programación in-circuit y un conector RJ-45 para cables de red.

La conectividad y la sencillez de la instalación han convencido a los departamentos de ingeniería de producción sobre la conveniencia de reconvertir las líneas de producción existentes para que puedan emplear Ethernet industrial.

Ejemplo de control remoto

Sobre este tema encontramos una infinidad de aplicaciones, como los sistemas de alerta de incendio municipal. Estos trabajan sobre Internet porque los puestos de control de incendio están distribuidos por toda la ciudad. Como todos los edificios cuentan con una red Ethernet, podemos utilizar el protocolo IP para establecer la comunicación entre las estaciones y transportar los datos en los protocolos TCP o UDP para que una computadora central lleve adelante las acciones de control. También encontramos muchos ejemplos en domótica (casas inteligentes), donde podemos enviar un e-mail y controlar, el sistema de refrigeración y las alarmas de un hogar (Figura 9).

En la actualidad, casi todas las nuevas líneas de producción están implementando Ethernet industrial

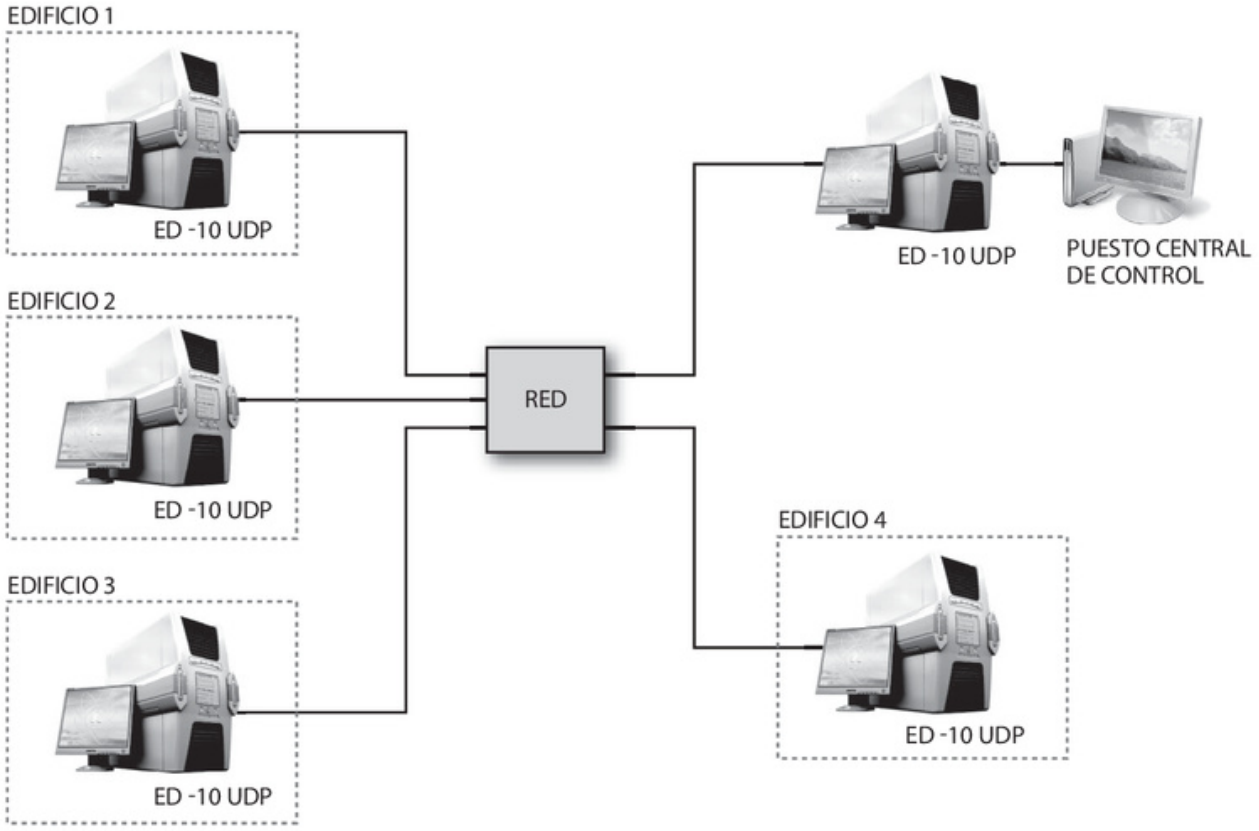


FIGURA 9. **Ejemplo de control para un sistema de alertas de incendio municipal, donde las redes de los edificios son Ethernet.**



MÉTODO GET PARA CONTROLAR VARIABLES

Este es un método sencillo para enviar un comando hacia un servidor web embebido, por medio del uso de formularios. Estos formularios pueden observarse, por ejemplo, al iniciar una búsqueda en Google, cuando tenemos que completar un campo de información y, luego, presionar **[Búsqueda]** para enviarla. De esta manera, el formulario web nos permite crear una interfaz sencilla entre el usuario y el servidor remoto. Dado un formulario, el envío de datos puede efectuarse a través de dos métodos: **GET** y **POST**.

El **método GET** envía los datos anexados con la dirección URL de la página. Estos son más simples de procesar en el microcontrolador, porque quedan guardados directamente en el buffer del módulo. La desventaja de este sistema es que admite un máximo de caracteres para enviar.

Por su parte, **POST** manda los datos en el cuerpo de la petición. El acceso es más complicado, pero el sistema tiene la ventaja de que no está limitado en cuanto a la cantidad de valores que se puede enviar.

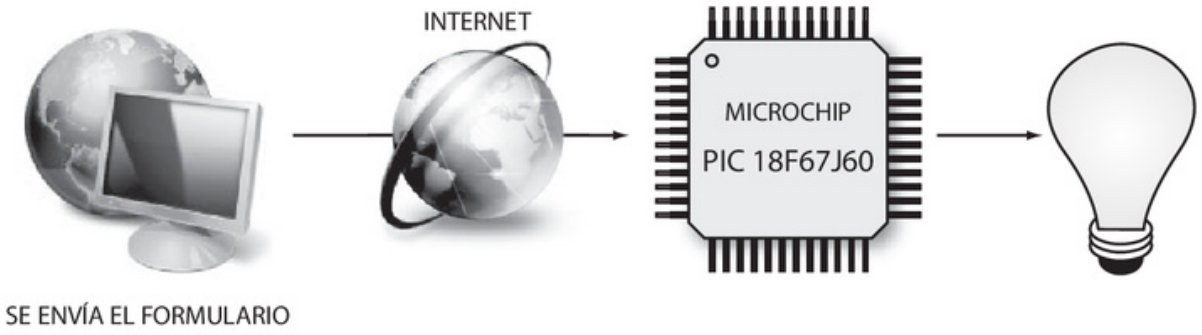


FIGURA 10. **En un formulario web podemos enviar** comandos, **para encender una** lámpara **o un** relay.

SOBRE MÉTODO GET

Los valores enviados por el método `GET` **son más fáciles de procesar, comparados con los de** `POST`**. Pero, tenemos un límite máximo de datos para enviar -normalmente, de unos 100 bytes-. Si la cantidad de datos es menor a este valor, utilizaremos** `GET`**, si es mayor, recurriremos a** `POST`**.**

Cuando las variables se envían al servidor, se codifican como una serie de pares nombre/valor

FORMULARIOS WEB

Los formularios web pueden ser programados en distintos lenguajes HTML, pero son reconocidos cuando observamos la sentencia **<form>**. Dentro de estas etiquetas encontramos los campos **<input>**. Cada etiqueta representa un elemento de entrada de la interfaz con el usuario (como un campo de texto, una casilla o un botón) y se le asigna un nombre de parámetro, que controla el campo devuelto por el explorador (Figura 10).

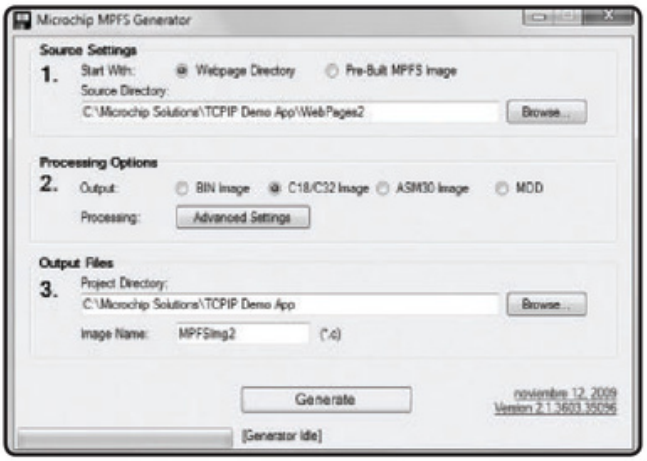
Cuando las variables se envían al servidor, se codifican como una serie de pares nombre/valor. Cada campo está separado por un símbolo (&), y los nombres se encuentran separados de los valores por un signo igual (=).

Veamos un pequeño ejemplo de un formulario web en código HTML. El encabezado se escribe tal como se muestra en el código siguiente:

```
<form method="get" action="luces.htm">

<input type="radio" name="luces"
value="on" ~Val_lights(1)~ /> On
<input type="radio" name="luces"
value="off" ~Val_lights(0)~ /> Off

</form>
```

Dicho código es una pequeña porción extraída de una página, que nos servirá para ver cómo actúa el formulario. En la primera línea abrimos el cuerpo del formulario y le indicamos que se utilizará el método **GET**. Luego, tenemos los campos de **input**: el parámetro **name** controla el nombre del campo, que en este ejemplo es luces. A continuación, aparecen los valores que toman las variables dinámicas definidas en el código.





Por último, se cierra el cuerpo del formulario. Con esto seremos capaces de reconocer un formulario en el código HTML.

MANEJO DE LOS DATOS CON EL MÉTODO GET

El método GET coloca los datos al final de la dirección con el formato que se ve en el código siguiente:

```
/form.htm?luces=on&brillo=50
```

En dicha sentencia se indica que el valor del campo "**luces**" es "**on**" y el "**brillo**" se establece en un valor enter "**50**", por ejemplo. Luego los datos enviados por el método GET se decodifican automáticamente y se almacenan en la matriz **curHTTP.data**.

Esta estructura está definida en el **stack TCP/IP de Microchip**. Luego, podemos realizar cualquier acción cuando se llama a la función **HTTPExecuteGet()**, lo cual sucederá cada vez que se presenten datos desde el formulario.

Encender un LED y un relay **por Ethernet**

Vamos a efectuar de manera práctica el encendido de un LED y de un relay a través de un formulario web, desde el cual enviaremos los datos al servidor web conectado a la PC por medio del conector de red o, si fuera necesario, a través de un router.

En primer lugar, realizaremos en forma sencilla el encendido de un LED por el método GET y, luego, extenderemos el ejercicio para encender de manera simultánea varios LEDs y relays.

Con las nociones anteriores, estaremos preparados para llevar a cabo el control básico a distancia de cualquier sistema que lo requiera.



VARIABLES GET

Con el método `GET` es posible enviar variables a través del campo URL del navegador. Esta información as almacenada en la matriz `curHTTPExecuteGet()`. Podemos, luego, consultar el estado de todas las variables y realizar una determinada acción desde el stack.

ENCENDIDO DE UN LED POR ETHERNET

Comenzaremos con un ejemplo sencillo para encender un LED a través del método que estuvimos estudiando. Lo primero que debemos tener es una plantilla web con un formulario. En la librería de Microchip que trae el CD de la obra, encontraremos una plantilla en la carpeta TCPIP WebVend App; es un ejemplo simple con un formulario para el encendido de un LED.

En el código HTML reconoceremos el formulario, porque comenzará con el campo **<form>** y, luego, seguirán las etiquetas <input>. Veremos algo similar a lo que muestra el código a continuación:

```
<form method="get" action="lights.htm">

<!-- Estos dos campos son creados para
seleccionar el On/Off -->
<input type="radio" name="lights"
value="on" ~lights_chk(1)~ /> On
<input type="radio" name="lights"
value="off" ~lights_chk(0)~ /> Off

<!-- En el formulario hay un botón de
seteo -->
<input type="submit" class="btn"
value="Set"/>

</form>
```

Una vez que reconocemos el campo del **<form>**, vemos cómo se define el método GET para enviar los datos, con la sentencia **method = "get"** (Figura 11).

Luego, vemos el campo de las etiquetas **input**, donde podemos encontrar el parámetro "**name**". Este dato es importante porque será utilizado en la función **callback** que llame el envío de los datos.

En este caso, el parámetro toma el nombre "**lights**". Luego, podemos ver la definición de dos variables dinámicas: **lights_chk(0)** y **lights_chk(1)**, ambas escritas entre los caracteres (**~**). Estas variables dinámicas están para ajustar un valor predeterminado cuando se carga el formulario.

Además, tienen su propia función **callback**, y la única función que cumplen es mantener un valor predeterminado al cargar el formulario. La última etiqueta de **<input>** genera un botón para enviar los datos del formulario. Como no tiene en su sentencia el parámetro "**name**", no manda ninguna información adicional.

FUNCIÓN CALLBACK PARA ENCENDER EL LED

Como mencionamos anteriormente, los datos enviados por el método GET son procesados por la función **HTTPExecuteGet()**. Los que recibimos del formulario automáticamente se decodificarán y se almacenarán en la estructura **curHTTP.data**.

Los datos que recibimos automáticamente se decodificarán y almacenarán en la estructura curHTTP.data



FIGURA 11.
Esta es la plantilla que podemos encontrar en la librería de Microchip, donde está definido el formulario que acabamos de ver.

Luego, podemos obtener los datos recibidos con la función **HTTPGetArg()** y localizar los valores asociados a los nombres de los parámetros. Entonces, la forma básica de una función que es llamada luego del envío de datos GET es la siguiente: primero, inicia la búsqueda de los valores de entrada y, luego, realiza las acciones necesarias, según los valores hallados. El código siguiente muestra esta función.

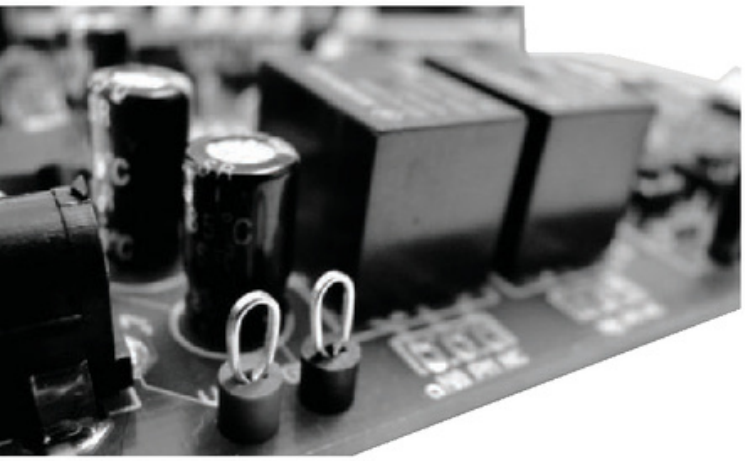
```
HTTP_IO_RESULT HTTPExecuteGet(void)
{
BYTE *ptr, i, name[20];

// Busco el nombre del archivo
MPFSGetFilename(curHTTP.file, name, 20);

// Nos aseguramos que es el archivo deseado
if(strcmppgm2ram((char*)name,(ROM
char*)"lights.htm") != 0)
return HTTP_IO_DONE;

// Buscamos el valor del nuevo estado
ptr = HTTPGetROMArg(curHTTP.data, (ROM
BYTE *)"lights");

if(ptr) // nos aseguramos que el valor
no sea nulo
{// seteamos el nuevo estado
if(strcmppgm2ram((char*)ptr, (ROM
char*)"on")== 0)
LED1_IO = TRUE;
else
LED1_IO = FALSE;
}
// Finalizamos
return HTTP_IO_DONE;
}
```

WEBPREVIEW

Utilizando Microchip WebPreview, podemos visualizar el formato que tendrá la página web sin necesidad de estar conectado al servidor. Esto nos será de suma utilidad al momento de diseñar la interfaz de usuario del servidor.

Hagamos un análisis del código anterior. Comenzamos definiendo la función **HTTPExecuteGet()**, que devuelve un parámetro del tipo **HTTP_IO_RESULT**, una variable definida por el stack TCP/IP de Microchip.

Luego de definir las variables que vamos a utilizar en la función, el primer paso es comprobar el nombre del archivo; podemos utilizar la función **MPFSGetFilename()** para buscar su nombre. Si tenemos varios formularios, nos será de utilidad saber cuál es el que generó el envío de datos. Para ese fin realizamos el bloque condicional if. Simplemente, hacemos la comparación de las cadenas de caracteres con la función **strcmppgm2ram()** y, si son diferentes, salimos de la función. En nuestro caso, de todos modos tenemos un solo formulario para atender.



El próximo paso es buscar el campo de datos. En nuestro ejemplo, solo tenemos un dato, pero es bueno saber que si hay más de uno para recibir, los campos no siempre llegan en orden. Por eso, efectuamos la búsqueda correspondiente con la función **HTTPGetROMArg()**, que recibe como argumento la estructura que contiene los datos (**curHTTP.data**) y el nombre del formulario del cual estos provienen.

Lo que nos devuelve es un puntero a los datos almacenados, que después comparamos con la cadena de caracteres "**ON**". Si son iguales, el LED se enciende; si son diferentes, se apaga. Por último, retornamos con el valor **HTTP_IO_DONE** para indicar que terminamos y que no necesita ser llamado otra vez.

MANEJO DE VARIOS LEDS Y RELAYS

Vamos a agregar más comandos al ejemplo, esta vez, para poder manejar tres LEDs y dos relays. Ahora, en la página tendremos un formulario más grande, donde agregaremos todos los comandos necesarios. Podremos encontrar una plantilla con la cual trabajar en el directorio **TCPIP Demo App\WebPages2**, con el nombre **form.htm**.

La estructura del formulario donde agregamos dos relays y tres LEDs se observa en el siguiente código.

FUNCIÓN STRCMPPGM2RAM

La función `strcmppgm2ram()` compara cadenas de string. Por ejemplo, si dos cadenas de caracteres son iguales, la función devuelve **0**, lo cual es muy útil para validar datos ingresados por el usuario. Esta función está definida en la librería `string.h`.

Luego de definir las variables que utilizaremos en la función tenemos que comprobar el nombre del archivo




```
<form method="get" action="forms.htm">
<div class="examplebox">
<b>R2:</b> <select
name="relay2"><option value="on"~ledS
elected(9,TRUE)~>On</option><option va
lue="off"~ledSelected(9,FALSE)~>Off</
option></select> 
<b>R1:</b> <select
name="relay1"><option value="on"~ledS
elected(8,TRUE)~>On</option><option va
lue="off"~ledSelected(8,FALSE)~>Off</
option></select> 

<b>L3:</b> <select name="led3"><option
value="1" ~ledSelected(3,TRUE)~>On</
option><option value="0"
~ledSelected(3,FALSE)~>Off</option></
select> 
```

```
<b>L2:</b> <select name="led2"><option
value="1" ~ledSelected(2,TRUE)~>On</
option><option value="0"
~ledSelected(2,FALSE)~>Off</option></
select> 
<b>L1:</b> <select name="led1"><option
value="1" ~ledSelected(1,TRUE)~>On</
option><option value="0"
~ledSelected(1,FALSE)~>Off</option></
select> 
<br /><input type="submit"
value="Enviar" style="margin-top:5px;">
</div>
</form>
```

UN DATO CURIOSO

Científicos chinos están trabajando en una tecnología que permitiría transferir datos usando simples LEDs. En sus pruebas, han conseguido conexiones con velocidades de hasta 2 Mbits por segundo. El resultado se obtuvo mediante el parpadeo controlado de las luces.

Luego, tenemos el código de la función que será llamada para controlar el estado de los LEDs y, también, de los relays:

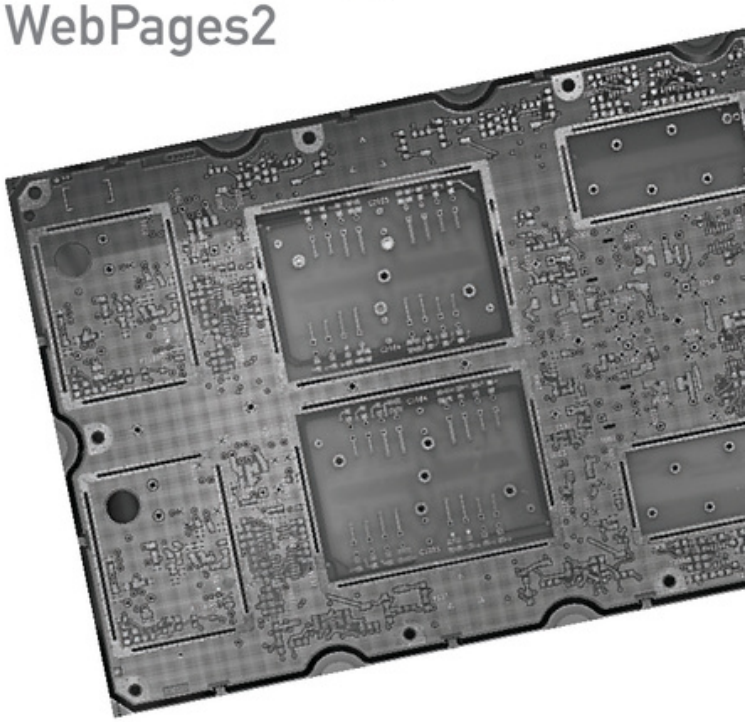
```
HTTP_IO_RESULT HTTPExecuteGet(void)
{
        BYTE *ptr;
        BYTE filename[20];

        // Cargamos el nombre del archivo
        // Asegurarse que filename[] tiene
el tamaño adecuado
        MPFSGetFilename(curHTTP.file, fi-
lename, 20);

        // Si los datos provienen de form.htm
        if(!memcmppgm2ram(filename,
"forms.htm", 9))
        {
                // Revisamos el estado de
cada LED
                ptr =
HTTPGetROMArg(curHTTP.data, (ROM BYTE
*)"led4");
                if(ptr)
                        LED4_IO =
(*ptr == '1');

                ptr =
HTTPGetROMArg(curHTTP.data, (ROM BYTE
*)"led3");
                if(ptr)
                        LED3_IO =
(*ptr == '1');

                ptr =
HTTPGetROMArg(curHTTP.data, (ROM BYTE
*)"led2");
                if(ptr)
                        LED2_IO =
(*ptr == '1');

                ptr =
HTTPGetROMArg(curHTTP.data, (ROM BYTE
*)"led1");
                if(ptr)
                        LED1_IO = (*ptr
== '1');


//--------------------Control del relay
----------------------------

                ptr =
HTTPGetROMArg(curHTTP.data, (ROM BYTE
*)"relay1");
    if(ptr)  // nos aseguramos que el valor
no sea nulo
    {
if(strcmppgm2ram((char*)ptr,(ROM
char*)"on") == 0)

RELAY1_IO = 1;
                        else
                          RELAY1_IO =
0;
                }

                ptr =
HTTPGetROMArg(curHTTP.data, (ROM BYTE
```

```
*)"relay2");
    if(ptr)  // nos aseguramos que el valor
no sea nulo
    {

if(strcmppgm2ram((char*)ptr,(ROM
char*)"on") == 0)

RELAY2_IO = 1;
                          else
                            RELAY2_IO =
0;
                }

}
// Finalizamos
return HTTP_IO_DONE;
}
```

ENCENDER UNA LÁMPARA DE 220 V A TRAVÉS DE ETHERNET

En el **Paso a paso 2**, vamos a utilizar todo lo que aprendimos sobre el método **GET** para enviar comandos y encender una lámpara de 220 V desde nuestra PC. Para esta práctica emplearemos el módulo de enseñanza MCE Web Server Dev y aplicaremos las plantillas de página web que provee

Podemos encontrar una plantilla con la cual trabajar en el directorio TCPIP Demo App\ WebPages2



este módulo. En la **Figura 12**, podemos ver el esquemático de la práctica podemos observar el PIC con sus conexiones hacia el relay que manejará el encendido de la lámpara.

FORMULARIOS

Si tenemos varios formularios que pueden enviar datos con el método GET, tenemos la posibilidad de utilizar la función `MPFSGetFilename` para obtener el nombre del archivo que está interactuando con el servidor. Así, podemos discernir, cuál de los formularios es el que envía los datos.

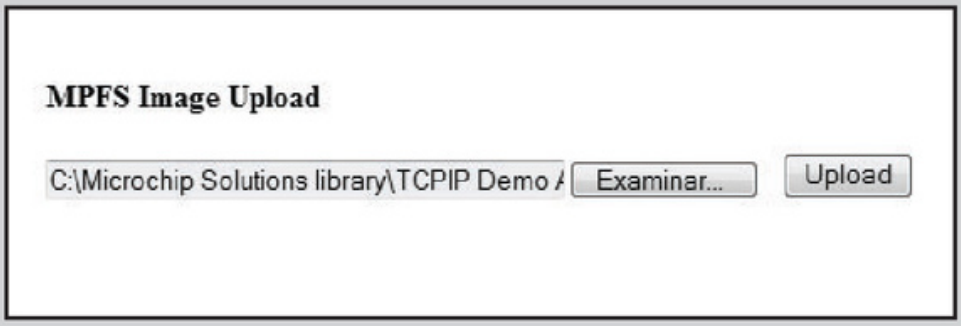
PASO A PASO /2
Encender una lámpara de 220 V a través de Ethernet

1 Tiene los relays del módulo Web Server y con ellos, puede manejar cargas superiores a las que soporta el PIC. Aquí, conectará la lámpara que quiere controlar. El pin central de la salida del relay indica el pin del medio; luego, puede optar entre el terminal normalmente abierto (**NA**) o normalmente cerrado (**NC**).

2 Debe generar el formulario que contiene la página web, que será la interfaz de control entre el usuario y el servidor. Resultará mucho más fácil hacerlo si tiene una plantilla prearmada entregada por Microchip o el módulo de enseñanza. De esta forma, los cambios que deba realizar serán mínimos.

3 Pase a generar el código que manejará los datos recibidos del formulario web. Recuerde que la función **`HTTPExecuteGet()`** es la encargada de realizar las acciones, según los datos que le envíe. Está en el archivo **`CustonHTTPApp.c`**.

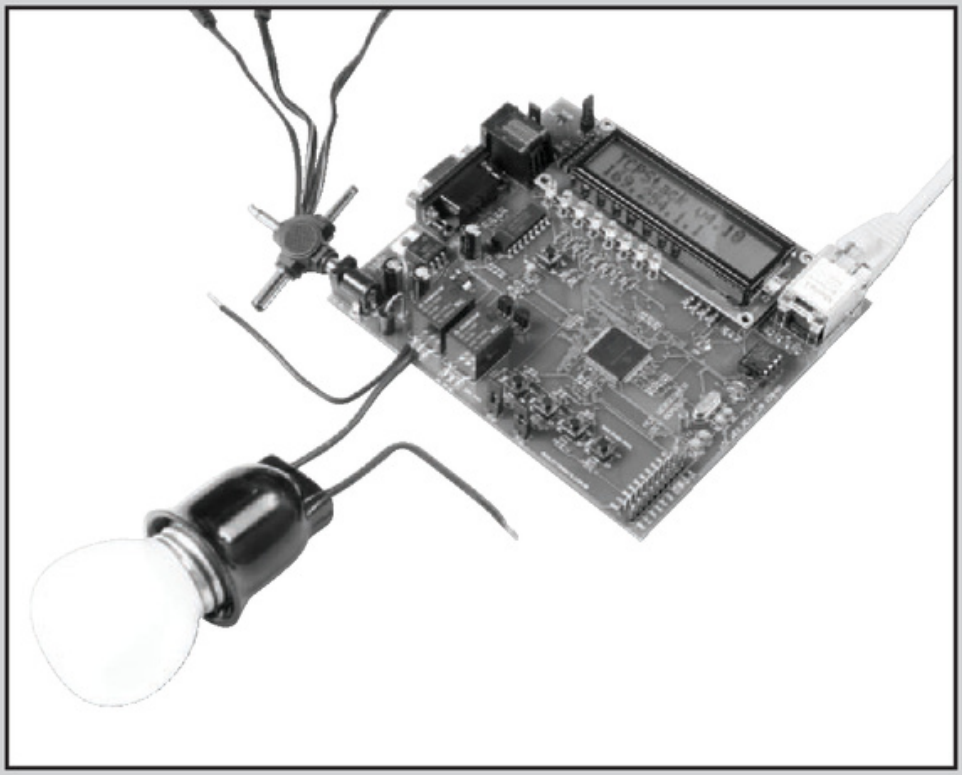
4



Una vez que termine de realizar la página y que escriba las funciones **`callback`** encargadas de manejar las acciones que realizará el servidor web, debe compilar el proyecto en el MPLAB. Si todo está correcto, tiene que compilarse satisfactoriamente y debe cargar el ejecutable en el PIC mediante un programador.

PASO A PASO /2 (cont.)

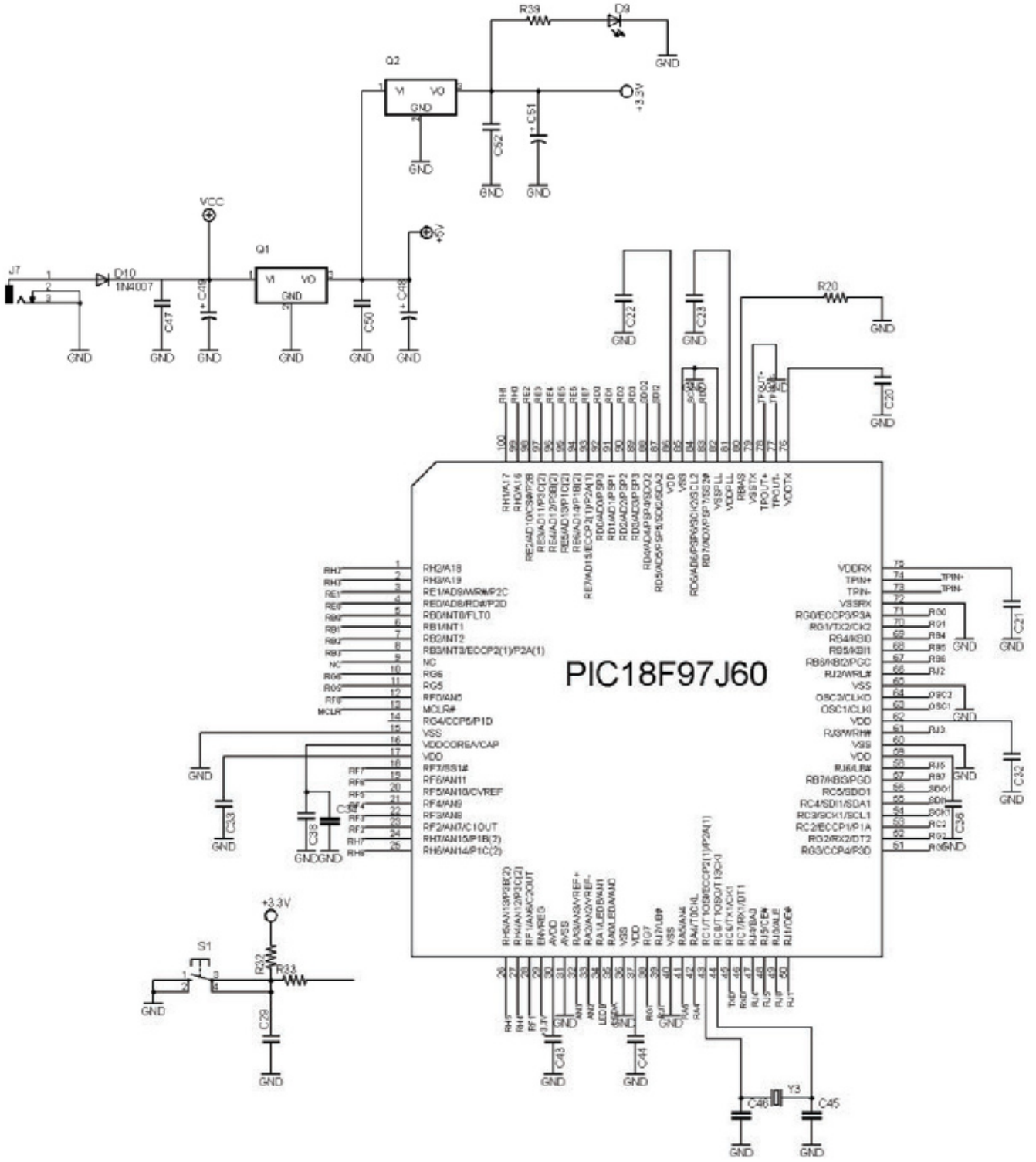
5



Convierta la página en un archivo que se pueda cargar en el PIC. Súbalo al servidor desde la URL **http://192.168.1.101/mpfsupload**. Seleccione la página por cargar y haga clic en **upload**. Desde **Formularios**, maneje el estado del relay que comanda a la lámpara de 220 V.

APLICACIÓN DE CONTROL

Las aplicaciones de control son de gran utilidad en domótica (sistemas capaces de automatizar una vivienda, aportando servicios de gestión energética, seguridad, bienestar y comunicación), ya que nos permiten el encendido de artefactos de un lugar remoto.


PIC18F97J60
R39
D9
Q2
Q1
R20
J7
D10
1N4007
VCC
+5V
+3.3V
S1
R32
R33
Y3
GND

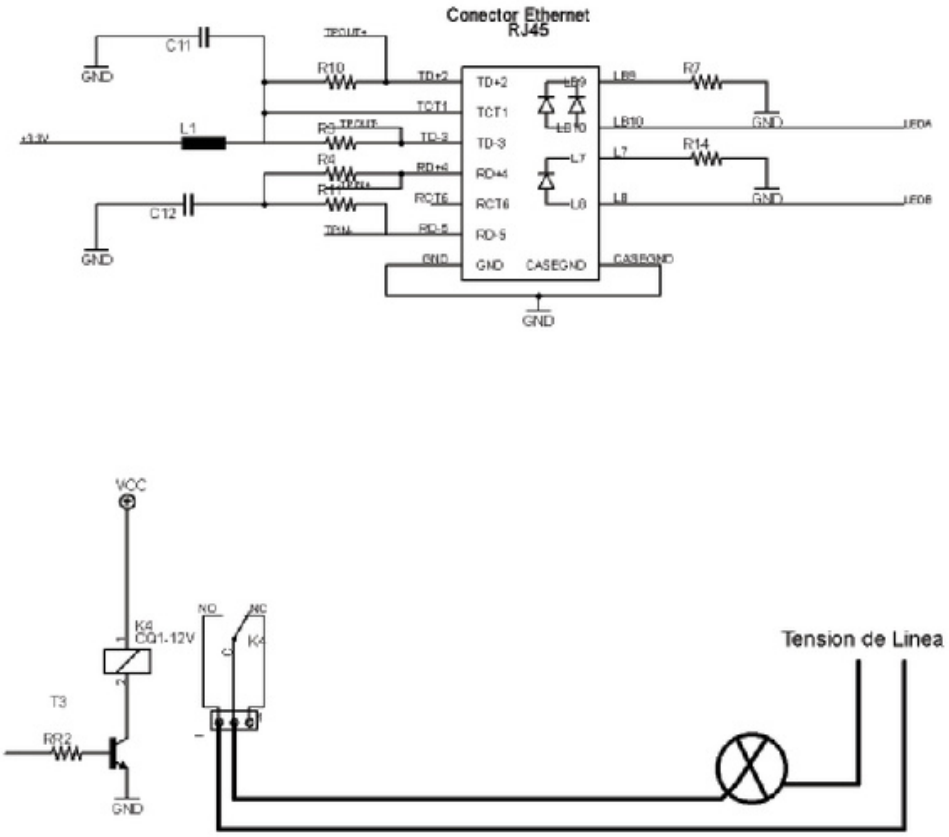


FIGURA 12. En este esquema a doble página podemos ver el PIC con sus conexiones hasta el relay que manejará el encendido de la lámpara. También vemos el conector de red, el RJ-45 y el circuito de alimentación.

¿TE RESULTA ÚTIL?
Lo que estás leyendo es el fruto del trabajo de cientos de personas que ponen todo de sí para lograr un mejor producto. Utilizar versiones "pirata" desalienta la inversión y da lugar a publicaciones de menor calidad.
NO ATENTES CONTRA LA LECTURA. NO ATENTES CONTRA TI. COMPRA SÓLO PRODUCTOS ORIGINALES.
Nuestras publicaciones se comercializan en kioscos o puestos de voceadores; librerías; locales cerrados; supermercados e internet (usershop.redusers.com). Si tienes alguna duda, comentario o quieres saber más, puedes contactarnos por medio de usershop@redusers.com

Multiple **choice**

1 ¿Por qué sentencia son reconocidos los formularios?

a- <input>
b- <form>
c- name
d- GET

2 ¿Cuál de las siguientes etiquetas representa un elemento de entrada de la interfaz con el usuario?

a- <input>
b- <form>
c- name
d- GET

3 ¿Cuál de los siguientes parámetros controla el nombre del campo?

a- <input>
b- <form>
c- name
d- GET

4 ¿Qué método puede enviar variables a través del campo URL del navegador?

a- <input>
b- <form>
c- name
d- GET

5 ¿Cuál de las siguientes funciones procesa los datos enviados por el método GET?

a- lights_chk(0)
b- HTTPGetArg()
c- HTTPExecuteGet()
d- curHTTP.data

6 ¿Con cuál de las siguientes funciones podemos obtener los datos recibidos?

a- lights_chk(0)
b- HTTPGetArg()
c- HTTPExecuteGet()
d- curHTTP.data

Respuestas: 1-b, 2-a, 3-c, 4-d, 5-c, 6-b.

Capítulo 5

WiFi con PIC y ZeroG



Conectaremos nuestros desarrollos
a una red inalámbrica WI-FI,
por medio del módulo ZeroG.

RedUSERS.com

Módulo ZeroG

La posibilidad de agregar conectividad inalámbrica en nuestros desarrollos ofrece muchas ventajas, por ejemplo, la capacidad de interconectar varias estaciones sin tener que modificar la infraestructura edilicia, como ocurre con las redes cableadas. Además, las redes **WiFi** se encuentran estandarizadas por la norma **IEEE 802.11**, lo que nos asegura la compatibilidad entre distintos dispositivos.

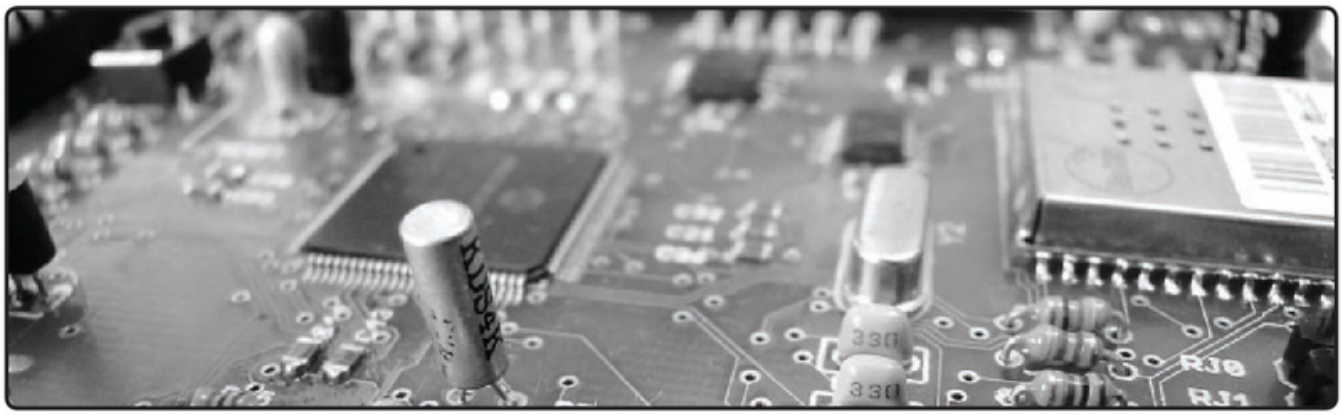
Entre las desventajas que presentan, podemos nombrar la baja velocidad de transferencia, si las comparamos con las redes cableadas, y la falta de seguridad en el envío de datos, ya que estos pueden ser capturados durante la transmisión. Pero esto último puede solucionarse trabajando con redes seguras, que utilizan encriptación de datos (**Figura 1**).

El estándar 802.11 establece las normas de funcionamiento para las capas 1 y 2 del modelo OSI

ESTÁNDAR 802.11

Este estándar establece las especificaciones y las normas de funcionamiento para las capas 1 y 2 del modelo OSI. Como características especiales, esta red trabaja en las zonas de frecuencia de **2,4 GHz** para los estándares **802.11b** y **802.11g**, y en la frecuencia de **5 GHz** en la norma **802.11a**.

Las velocidades más comunes de transmisión están en **11 Mbps** y en **54 Mbps**. Básicamente, encontramos



SOBRE WI-FI

Wi-Fi es una marca de la **Wi-Fi Alliance**, una organización comercial dedicada a certificar los dispositivos que cumplen con los estándares de la IEEE 802.11 para redes inalámbricas de área local. Fomenta el uso de las redes inalámbricas y asegura la compatibilidad entre equipos.

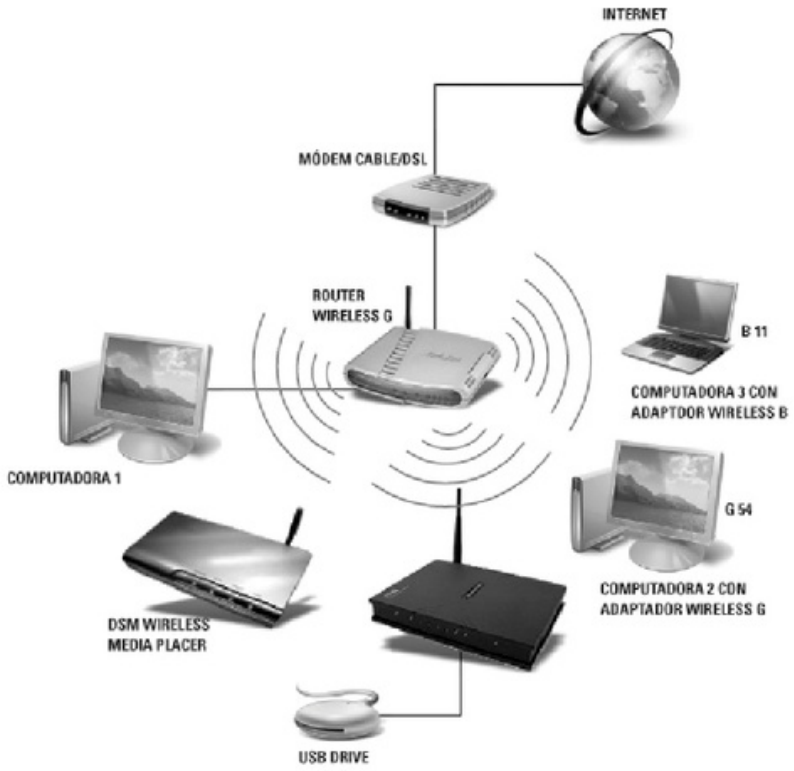


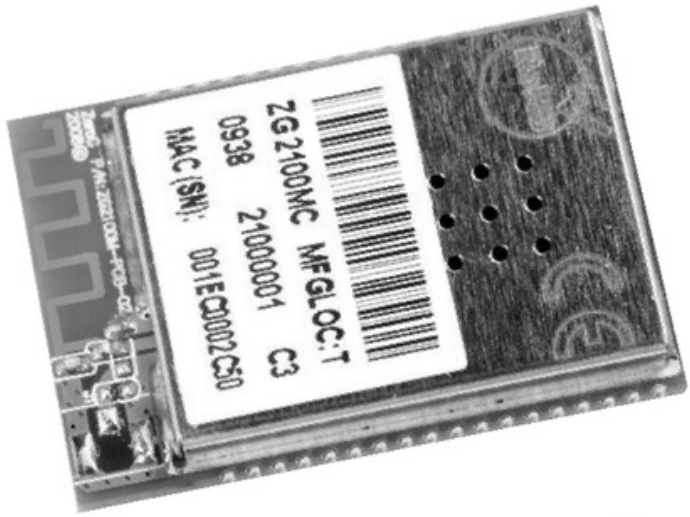
FIGURA 1. **Estructura típica de una** red Wi-Fi hogareña. **El router combinado con un punto de acceso trabaja para que todas las estaciones cliente tengan conectividad.**

dos estructuras para la interconexión de los diferentes dispositivos. Una de ellas se llama red WiFi de **infraestructura**, donde existen dos elementos: uno o más puntos de acceso y las estaciones cliente, que se conectan al servidor a través de los primeros. La otra estructura es la red **ad-hoc**, donde las estaciones cliente se conectan entre sí para intercambiar información.

MÓDULO ZG2100M

Este módulo es fabricado por la empresa **ZeroG** y, básicamente, agrega conectividad Wi-Fi en dispositivos embebidos. Trabaja bajo el estándar **802.11b**, que especifica una frecuencia de trabajo de **2,4 GHz** y alcanza velocidades de hasta **2 Mbps**.

Posee internamente todo lo necesario para la conexión a una red Wi-Fi, porque tiene una dirección **MAC** pregrabada y los componentes de **RF** necesarios para emplear el medio físico, incluyendo la antena.



Esto es ideal para nuestras aplicaciones, porque nos olvidamos de tener que lidiar con el arduo trabajo que implica diseñar una antena en el PCB.

Simplemente, debemos tomar algunas precauciones en el layout (esquema de distribución de los componentes en el PCB) para poder integrar este módulo, que veremos más adelante.

Características del módulo ZG2100M

Estas son las características más interesantes sobre este módulo de conectividad (**Figura 2**):

- Trabaja con el estándar 802.11b de Wi-Fi.
- Tiene la dirección MAC pregrabada.
- Posee los componentes de RF y amplificadores de banda base para establecer la comunicación.
- Soporta protocolos de seguridad WEP, WPA y WPA2.
- Tiene bajo consumo de alimentación.
- Se alimenta a través de una fuente de 3,3 V.
- Presenta velocidades de 1 y 2 Mbps.
- Ofrece licencia gratuita de la pila (stack) TCP / IP para Internet.
- Incluye antena en el PCB.
- Sus dimensiones son mínimas: 21 mm x 31 mm.
- Cuenta con 36 pines de conexión.

Todas estas especificaciones hacen del módulo un elementos ideal para integrar conectividad en diferentes ambientes de la electrónica; por ejemplo, en sistemas de monitoreo y control; en industrias; en sistemas de computación; para realizar dispositivos inalámbricos como teclados, mouse y videocámaras o en dispositivos electrónicos de consumo.



El módulo ZG2100M agrega conectividad Wi-Fi en dispositivos embebidos

Alimentación del módulo

El módulo se alimenta desde una simple fuente que entrega 3,3 V, ya que internamente posee un regulador de tensión LDO (*low drop-out*), que genera

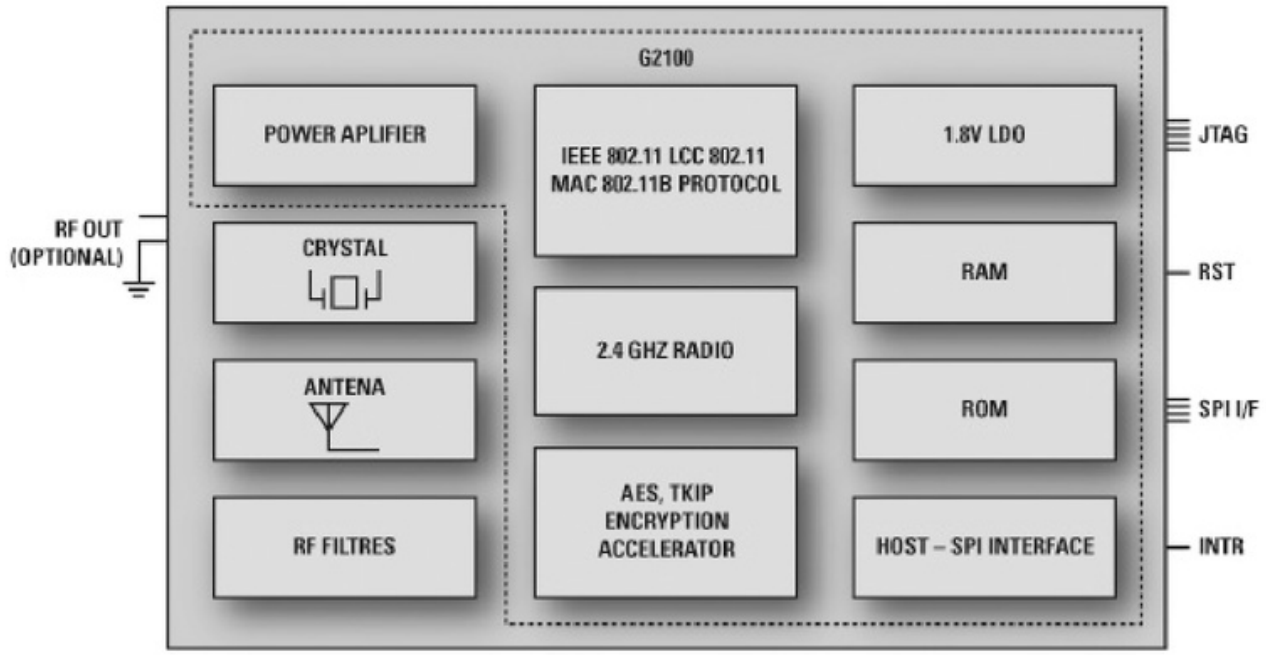


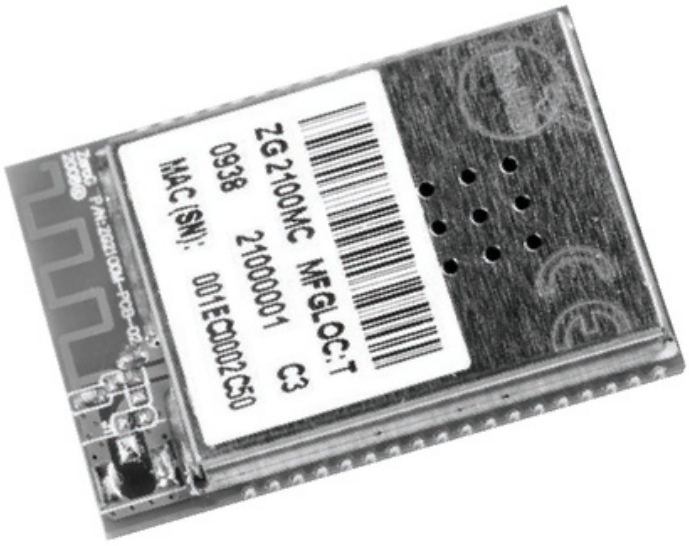
FIGURA 2. **Diagrama funcional del módulo ZG2100M**

las tensiones necesarias en el núcleo de procesamiento. En este módulo contamos con una entrada directa opcional para una fuente de **1,8 V** que alimentará el núcleo del dispositivo.

Sin embargo, podemos no utilizarla, gracias al regulador interno mencionado anteriormente. En la **Figura 3** podemos ver el diagrama de conexión del módulo **ZG2100M**. Otro aspecto para destacar es el bajo consumo que posee el dispositivo, que en modo **Sleep** consume **250 µA**, y en el de hibernación llega a **0,1µA**. Esto permite que sea alimentado desde una batería. Cuando transmite en su máxima potencia, el núcleo llega a consumir **230 mA** de la fuente de alimentación.

Comunicación con el módulo ZG2100M

Este módulo se caracteriza por la simplicidad en la programación, que se efectúa a través de **APIs** (*Application Programming Interface*) y no requiere de un sistema operativo. En el manejo de la conexión, el módulo **ZeroG** controla la capa **MAC** y las capas de base, y se conecta al **microcontrolador que realizará el control del módulo (MCU Host)** por medio de un puerto SPI (**Figura 4**).



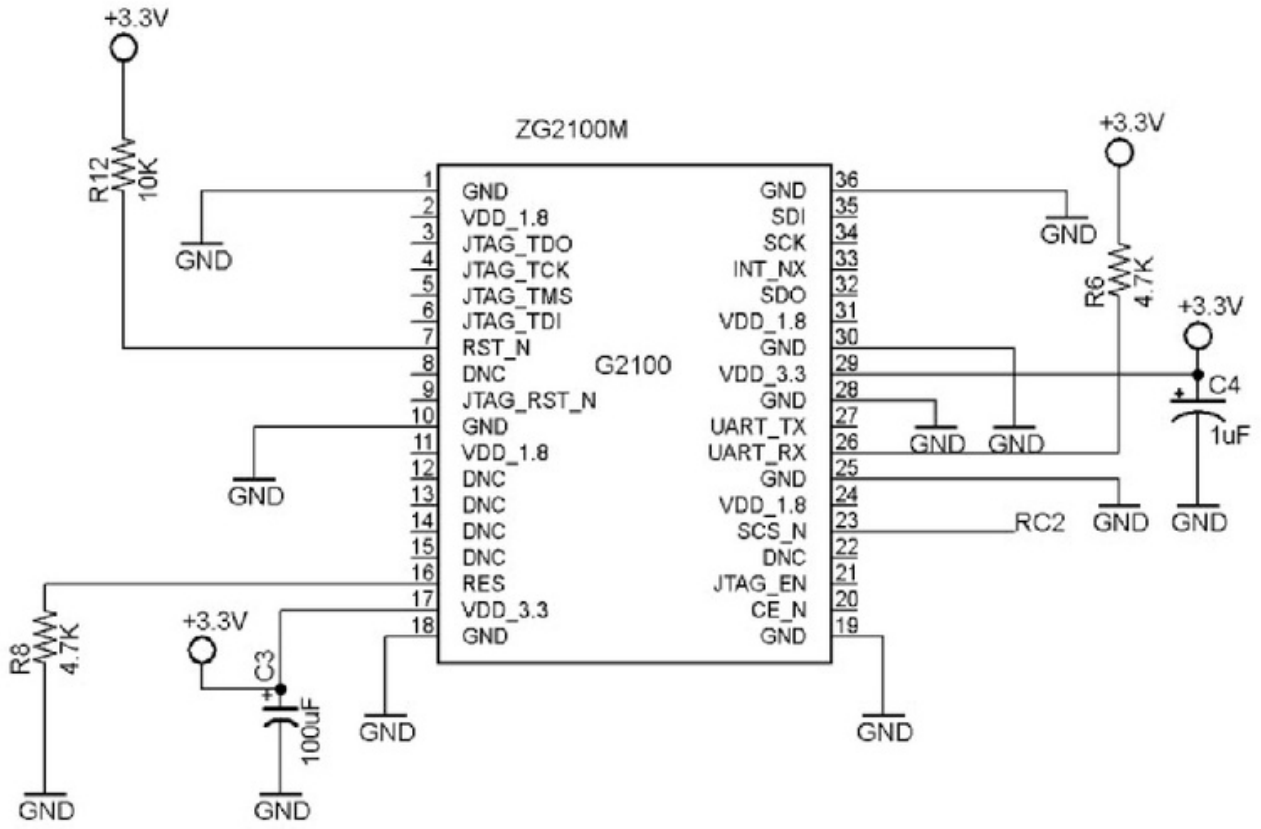


FIGURA 3. **Conexiones de alimentación del módulo ZG2100.**

Los microcontroladores de 8,16 y 32 bits residentes en la placa de desarrollo se encargan del control de la pila de la red de TCP/IP y de correr las aplicaciones del sistema. Actualmente, los módulos ZeroG incorporan soluciones WiFi para microcontroladores PIC, porque se han integrado los **drivers Easy-FI** (estos permiten usar el módulo ZeroG de manera muy simple) en el stack de TCP/IP de microchip, de forma tal que no hay necesidad de descargar un driver separado de Wi-Fi.

Esta es una de las razones que vuelven tan atractiva la integración de este módulo en los desarrollos con PIC.

CHIP SELECT

Debemos recordar que, haciendo uso del pin **Chip Select**, podemos conectar varios dispositivos en un mismo bus de comunicación. Esto nos permite tener el módulo de conectividad trabajando en el mismo bus que utiliza una memoria EEPROM SPI externa.

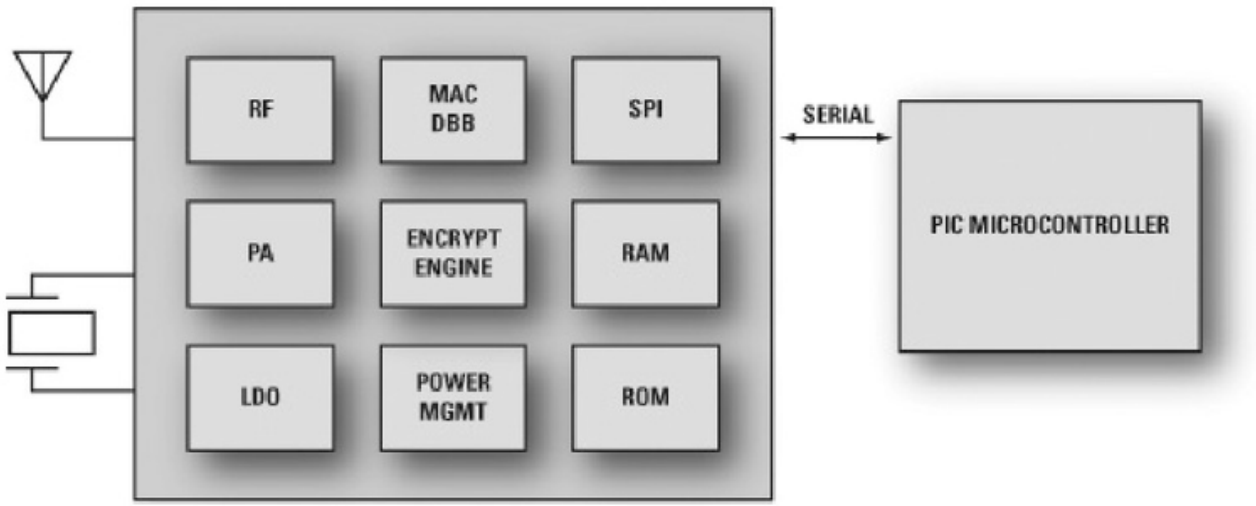


FIGURA 4. **El módulo de conectividad Wi-Fi permite la conexión con los microcontroladores a través de la comunicación SPI.**

CONEXIÓN DEL MÓDULO ZEROG POR SPI

En el módulo ZG2100M hay dos interfaces SPI. Una está dedicada a las comunicaciones **SPI en modo esclavo**, en donde el dispositivo maestro controlará las acciones del módulo.

La otra es un puerto **SPI maestro**, donde el módulo interactúa con una memoria externa tipo **ROM** para obtener un listado de las tareas por realizar. Nosotros utilizaremos la **interfaz SPI esclavo** para la comunicación entre el microcontrolador y el módulo. En la **Figura 5** vemos un esquema donde se muestra la conexión entre los dispositivos. La conexión será similar a los dispositivos SPI que estudiamos en la obra.

El módulo controla la capa MAC y las capas de base, conectándose al micro por medio de SPI

PIC18 F97J60

El microcontrolador PIC18 F97J60, que forma parte de la familia de microcontroladores con arquitectura Harvard perteneciente a Microchip, cuenta con dos módulos SPI, por si queremos dedicar un puerto de comunicación solo para el módulo de conectividad.

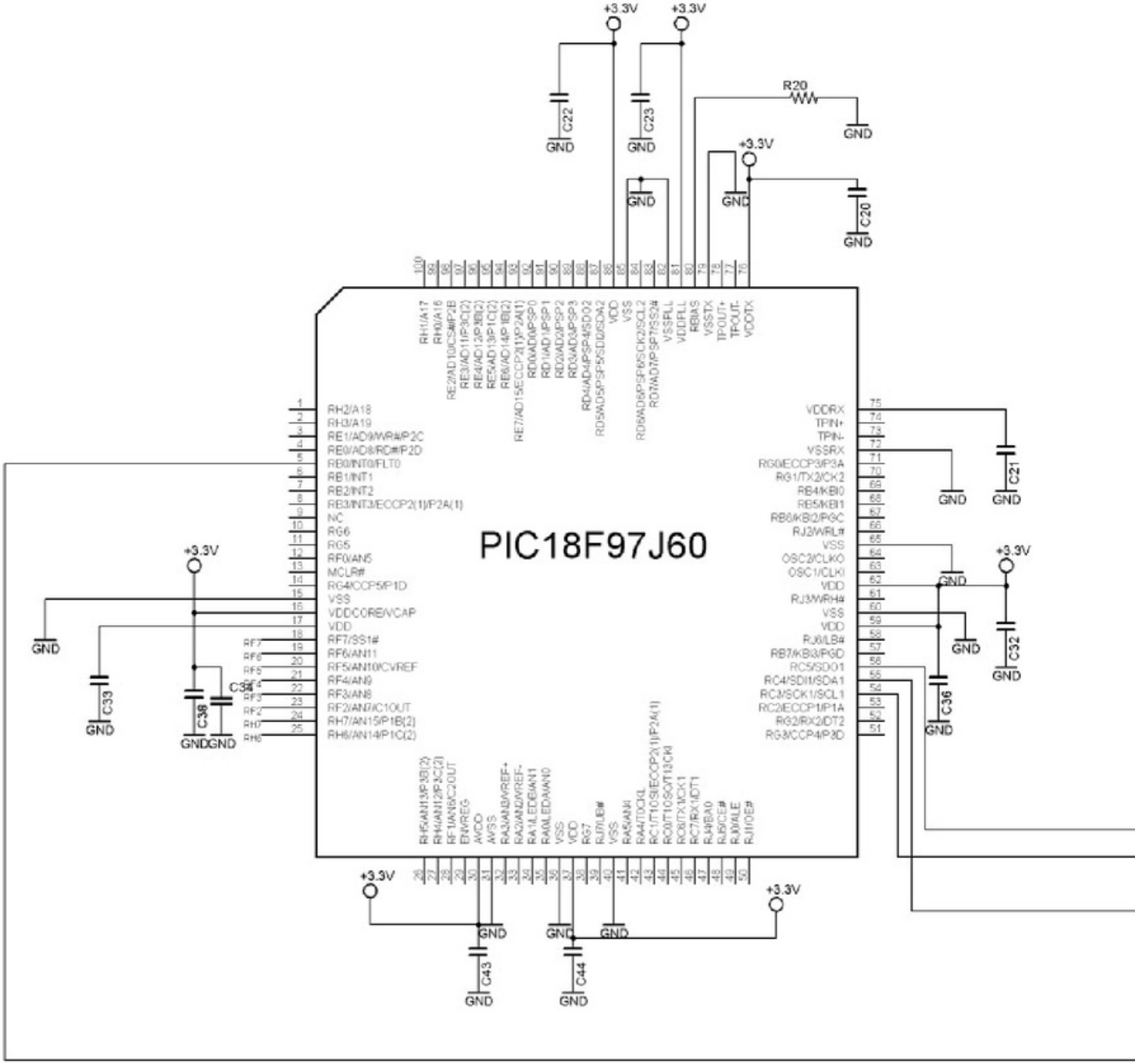
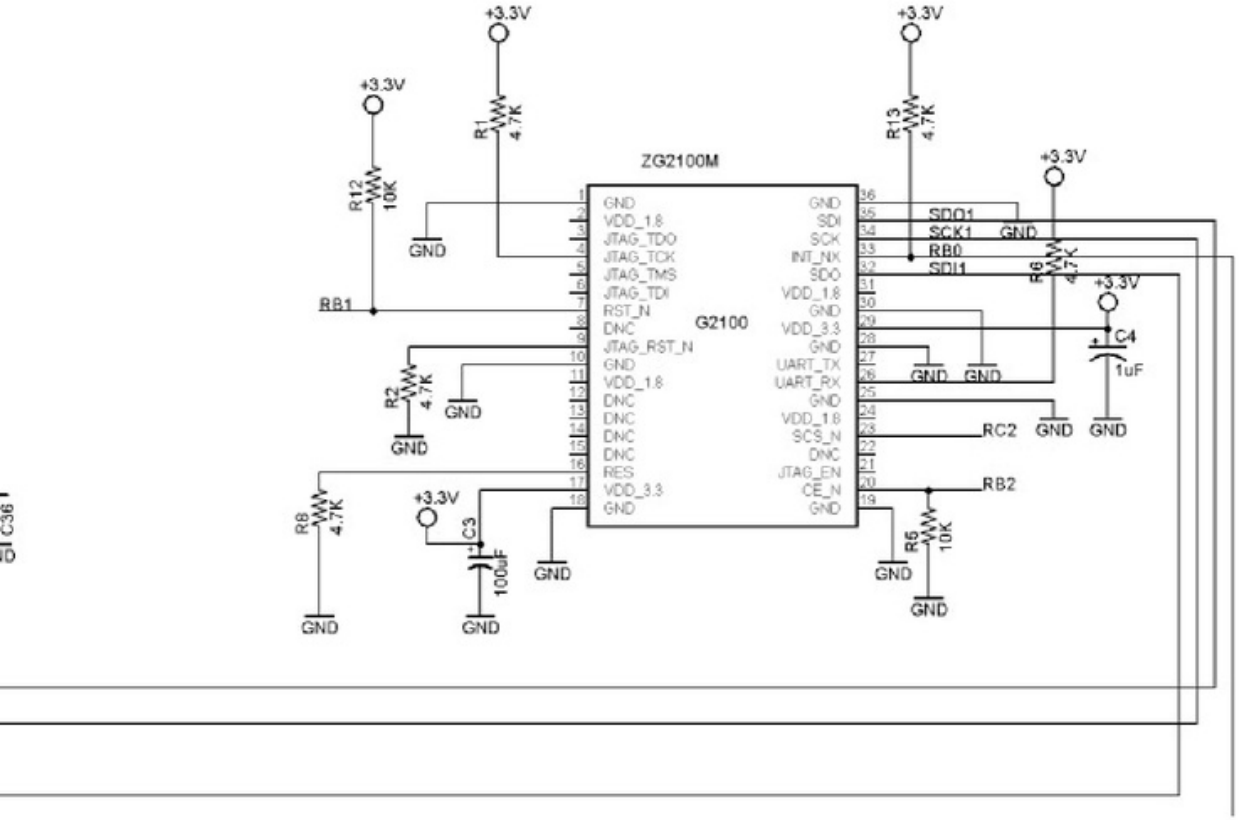


FIGURA 5. **Esquema con las conexiones del módulo ZeroG al microcontrolador PIC18F97J60 a través del la comunicación SPI.**



Interfaz SPI esclavo con interrupciones al host

La interfaz SPI esclavo se utiliza para la comunicación del módulo con el maestro, que en este caso, es el microcontrolador. Pero en este modo de comunicación se utiliza el `INT_NX` como pin de interrupción. Entonces, cuando los datos están disponibles para el maestro, la línea `INT_NX` del módulo se pone a nivel bajo. La línea `INT_NX` se pondrá en el valor **1**, después de que los datos sean transferidos al buffer del maestro. La velocidad máxima del reloj (CLK) es de **25 MHz**.

Control del módulo por el puerto SPI

Cuando enviamos los comandos de control al módulo, debemos tener en claro que este trabaja en el formato **big endian**. Esto quiere decir que primero debemos enviar el bit más significativo y, luego, los menos significativos, como lo indica la comunicación por SPI. Los datos serán recogidos por el módulo después de que se active la señal de **Chip Select** correspondiente y la señal de reloj tenga un flanco de subida.

Cuando operamos con el módulo ZG2100M, nos encontramos con dos regiones o espacios de trabajo que podemos manipular. Una es el espacio de acceso a los registros, que nos permite leer y modificar los registros internos del módulo. La otra está dedicada a la interfaz **FIFO** (*First In, First Out*, primero en entrar, primero en salir). Veremos más adelante en detalle cómo trabajar en estas zonas.

La comunicación del módulo con el microcontrolador se realiza por medio de SPI

ACCESO AL ESPACIO DE REGISTROS

Desde el dispositivo maestro, tenemos acceso al espacio de registro del **módulo ZeroG**, para lo cual tenemos que enviar una **trama de datos** a través de la interfaz SPI (**Figura 6**). Cada trama de datos que enviemos debe mantener el pin **Chip Select** a nivel bajo, para habilitar el módulo y mantener la señal de clock funcionando. El primer byte que enviamos contiene la siguiente información:

- **F**: selecciona cuál es la zona a la que vamos a acceder, si es el espacio del FIFO o el de registros. Si este bit está en **1**, le estamos indicando al módulo que el acceso es hacia el **espacio FIFO**. En cambio, si transmitimos un **0**, indicamos que vamos a trabajar con el espacio de registros.
- **R**: es el bit de **lectura/escritura**. Si el bit se pone a **1**, indicamos que la operación es de lectura. En cambio, si colocamos un **0**, hacemos referencia a una operación de escritura.
- **ADDR0**: es la dirección inicial para el envío de datos. Este valor solo se utiliza cuando accedemos

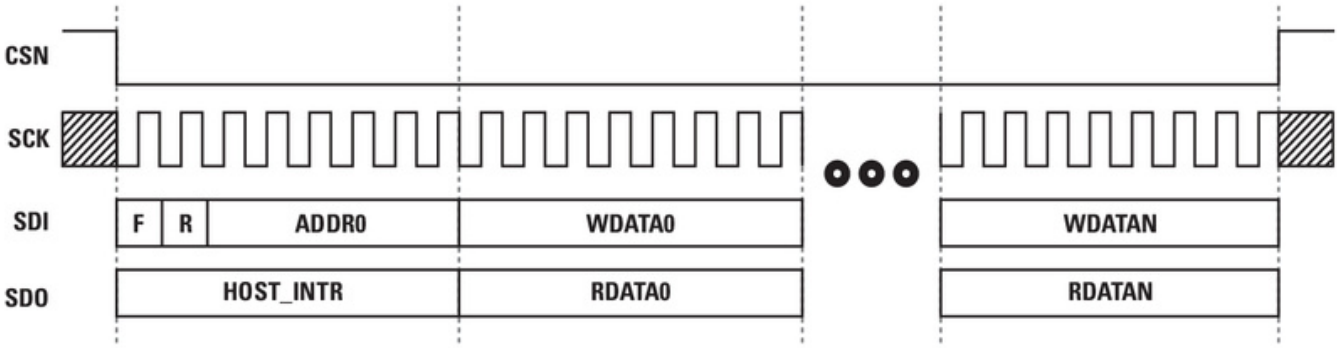


FIGURA 6. **Formato de las tramas para acceder al espacio de registros del módulo ZeroG.**

al espacio de registro, y es ignorado cuando accedemos al FIFO.

- **WDATAn**: son los bytes de escritura. Solo se utilizan durante la escritura, mientras que en la operación de lectura son ignorados.
- **RDATAn**: son los bytes de lectura. Son siempre válidos, tanto en la lectura como en la escritura.
- **HOST_INTR**: registro de interrupción de 8 bits.

INTERFAZ FIFO

En la **Figura 7** vemos el formato de la trama de datos que debemos enviar a la interfaz FIFO para leer o escribir datos. Básicamente, aparecen los mismos campos de control que en el acceso a los registros, salvo que en esta ocasión, vemos un campo de comandos (FCMD) y el campo RSV al inicio de la trama. FCMD especifica la tarea que queremos realizar con la pila y tiene un tamaño de 3 bits. Los comandos básicos que podemos enviar al módulo pueden verse en el siguiente código:

```
-FCMD[2:0] :
0x0 - RFIFO_CMD: comando de lectura
0x1 - WCONT: Continuar con la escritura
0x2 - WSTART0: Comenzar escritura
de paquetes
0x3 - WSTART1: Comenzar escritura
de paquetes
0x4 - WEND CMD: Fin de la escritura
0x5 - REND CMD: Fin de la lectura.
```

El campo **RSV** está reservado. Para obtener más información acerca de cómo comunicarse con el módulo, tendremos que consultar en las hojas de datos del dispositivo.



CONSIDERACIONES DE LAYOUT

Cuando colocamos dispositivos de **RF** en nuestros desarrollos, debemos tomar importantes precauciones a la hora de diseñar el **layout**.

Debido a que los sistemas RF trabajan en altas frecuencias, el diseño de las pistas de las placas influye en el funcionamiento del dispositivo. Este diseño afecta a los sistemas RF debido a los efectos de reflexión que pueden existir en una línea, y que dependen de las discontinuidades de la impedancia que puede mostrar hacia el dispositivo.

También es preciso considerar que los dispositivos que se encuentran alrededor de un módulo RF pueden ocasionar interferencias en nuestras

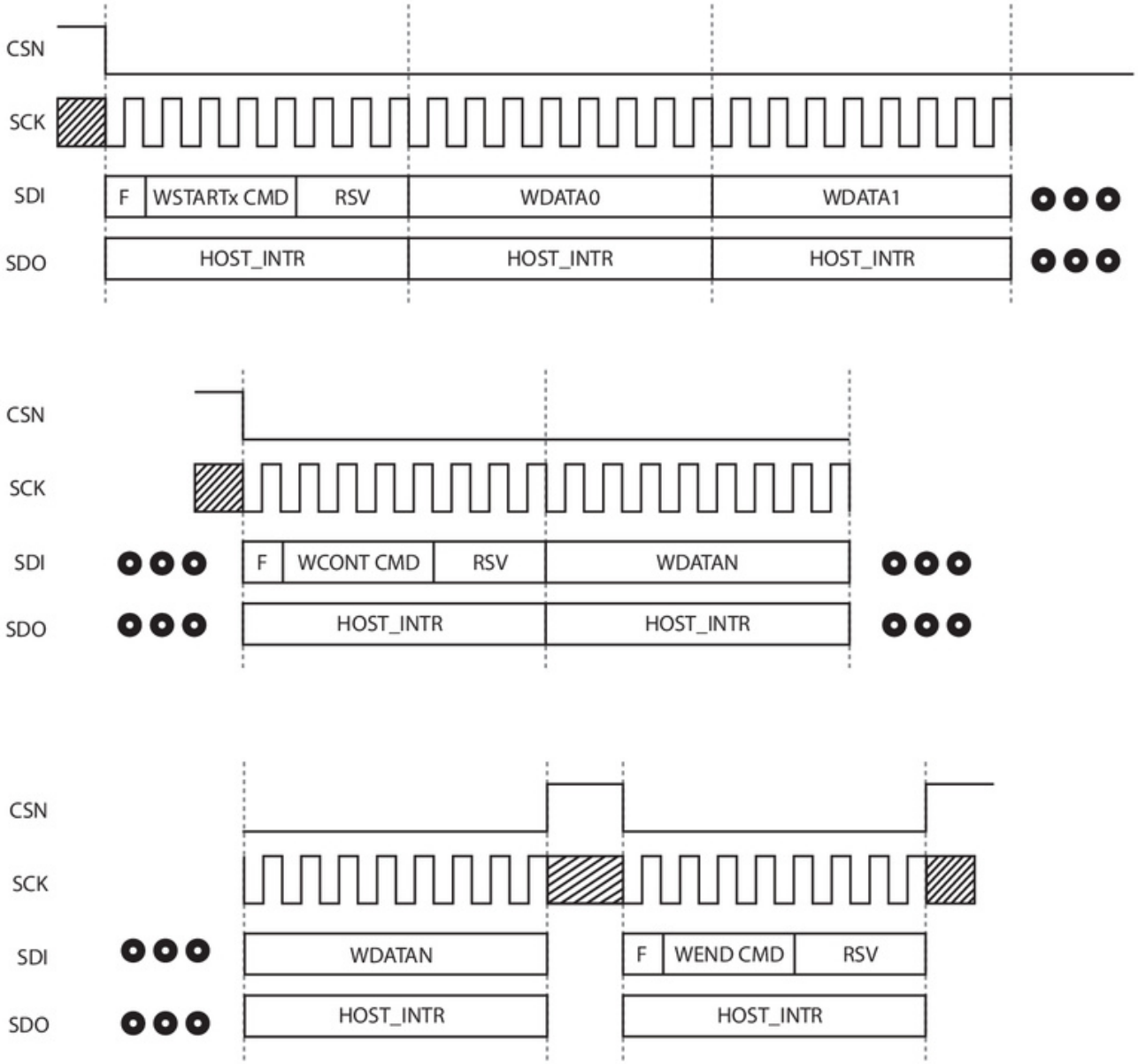


FIGURA 7. **Formato de las tramas de escritura hacia la interfaz FIFO.**

CIFRADO DE DATOS

Los sistemas de seguridad más comunes utilizan protocolos de cifrado de datos, que se encargan de codificar la información transmitida. Así, las redes inalámbricas pueden proteger la información que por ellas circula y resguardar su confidencialidad.

aplicaciones. Es por eso que debemos asegurarnos de que aquellos que emiten un campo magnético, como bobinas y transformadores, estén alejados una cierta distancia del módulo. Entonces, cuando trabajamos en el diseño de circuitos RF, necesitaremos contar con las herramientas y el software adecuados al diseño de estos dispositivos.

En el mercado encontramos muchas empresas que se dedican a solucionar los problemas de interferencia. También debemos tener en cuenta que todo equipo que emita radiación electromagnética está regulado por normas para evitar que interfieran con otros dispositivos y proteger al usuario del efecto nocivo de estas radiaciones.

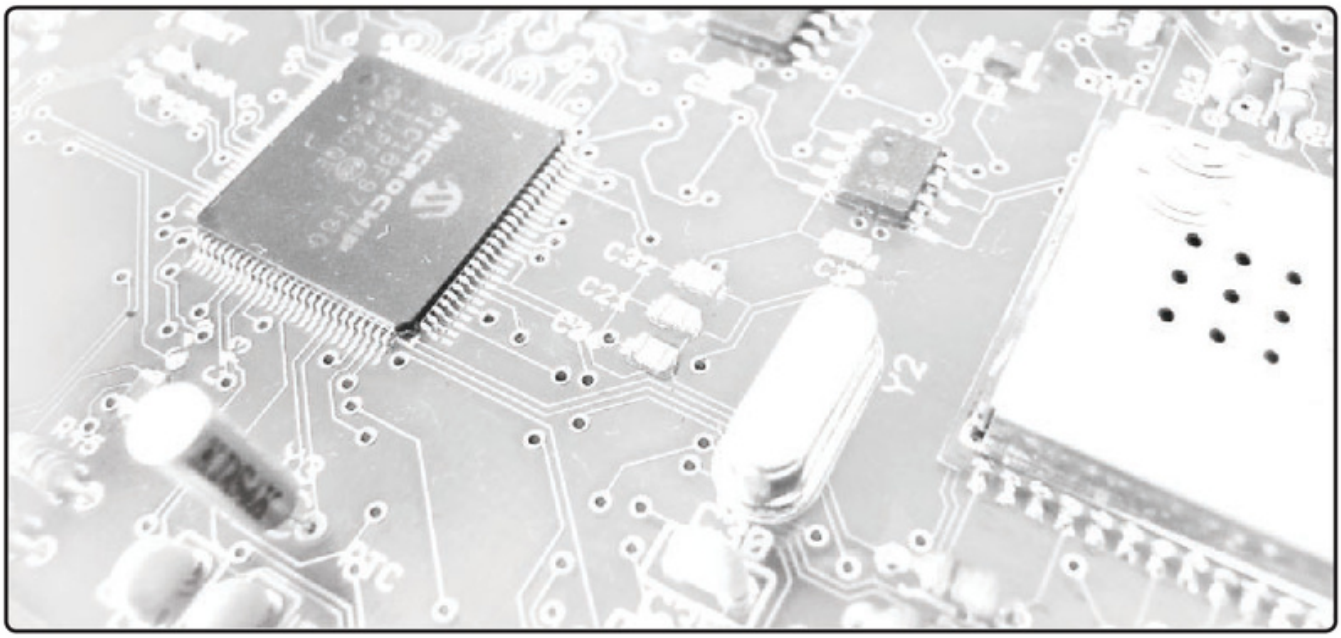
Debemos tener en cuenta todas estas consideraciones en el momento de insertar el módulo de conectividad WiFi, ya que trabaja en zonas de altas frecuencias. Pero, además, tenemos la ventaja de que el módulo **ZG2100M** incorpora en su PCB

En el diseño de circuitos RF, necesitamos un software de diseño adecuado para estos dispositivos

todos los componentes de RF y amplificadores de banda base para establecer la comunicación, lo que nos evita estar lidiando con el diseño tan complicado de la pistas de RF. Solo debemos tomar algunas precauciones que nos indica el fabricante para colocar el módulo sobre el PCB.

EJEMPLO DE DISEÑO DEL PCB

En la **Figura 8** podemos ver un ejemplo de cómo colocar el módulo sobre el PCB. En esa zona, el fabricante indica que no se deben trazar pistas por debajo del módulo, y tampoco deben encintrarse dispositivos cercanos.



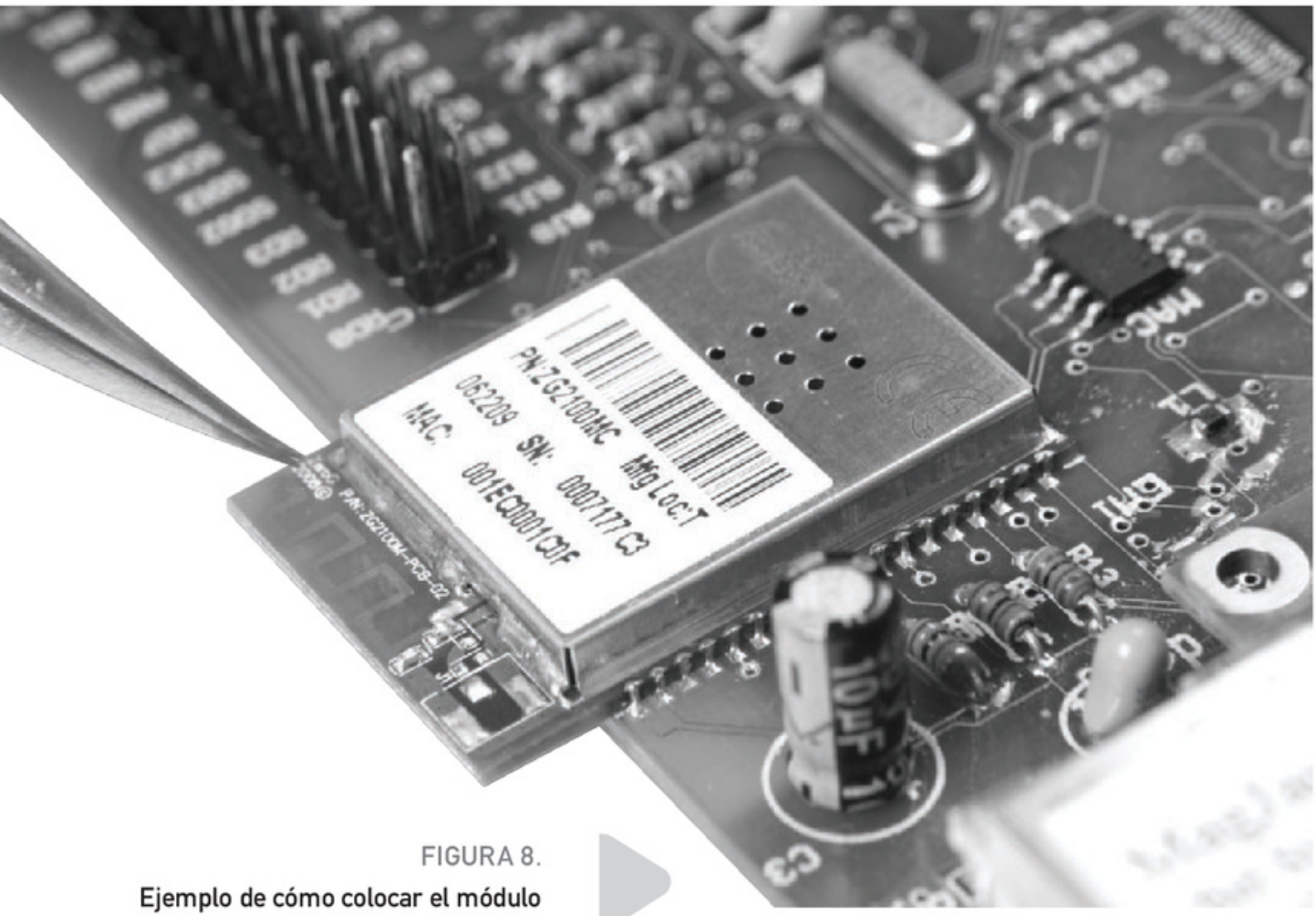


FIGURA 8.
Ejemplo de cómo colocar el módulo de conectividad Wi-Fi para evitar las interferencias indeseadas.

En la parte inferior del módulo, podremos ver algunos puntos de prueba que coloca el fabricante y varias vías que interconectan las capas del PCB. Debemos asegurarnos de que las vías que queden debajo del módulo tengan una máscara antisoldante.

La **Figura 9** muestra una zona del PCB donde no se coloca ninguna pista y tampoco, ningún plano de masa. Como esta zona es muy sensible a las interferencias por la proximidad de la antena, debemos tomar esta precaución al momento de diseñar el PCB que alojará el módulo.

En el mercado encontramos muchas empresas que se dedican a solucionar los problemas de interferencia

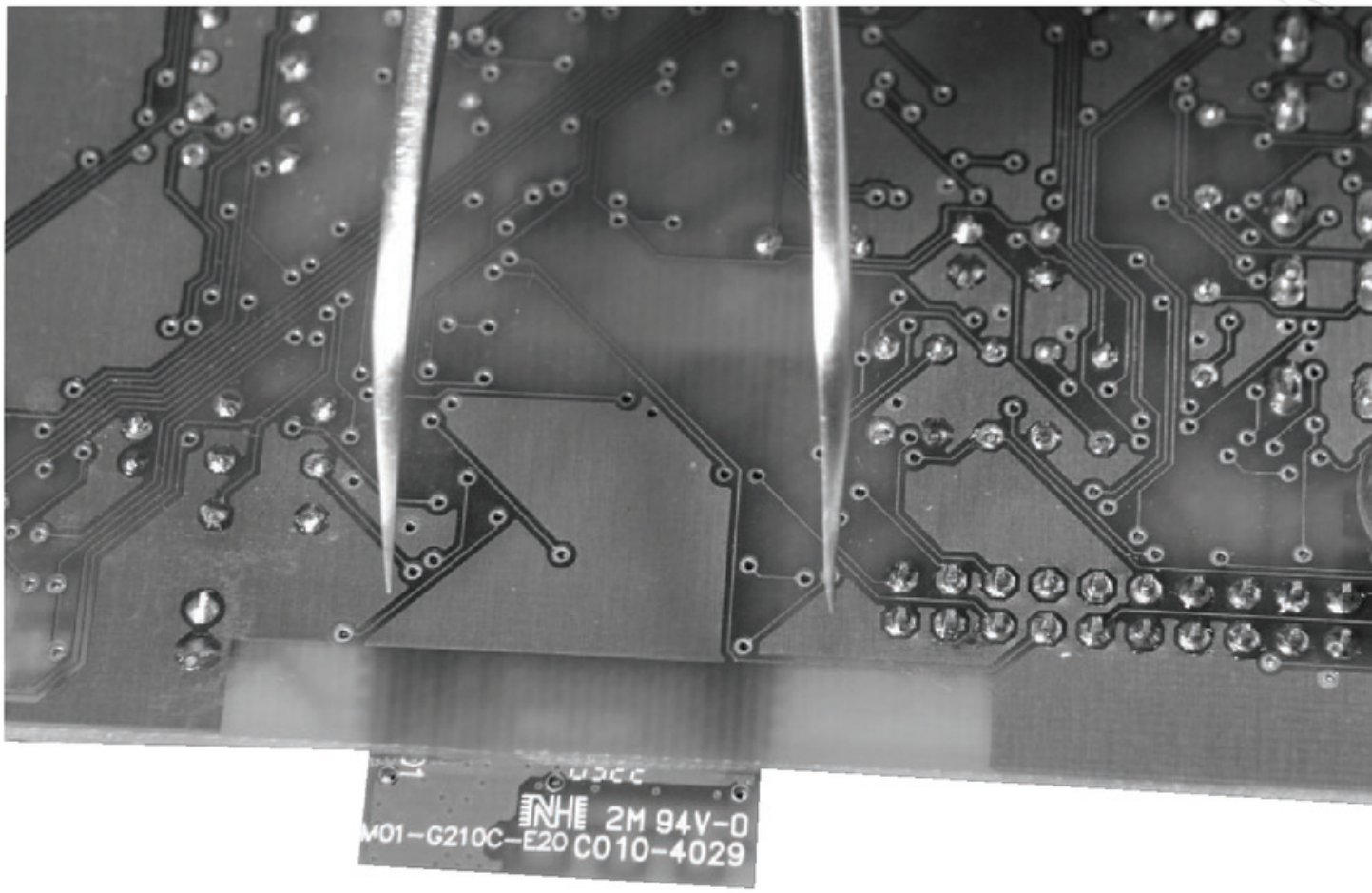


FIGURA 9. **Las pistas y los planos de masa pueden interferir en el correcto funcionamiento de la antena; por eso se delimita una zona donde no colocamos nada.**

Configuración del stack TCP/IP para Wi-Fi

Descubrimos cuáles son las ventajas de contar con conectividad inalámbrica y las diferentes tecnologías que brindan soluciones a esta necesidad, como ZigBee, Bluetooth o Wi-Fi.

Las redes Wi-Fi han extendido su espacio de uso, y hoy es posible encontrarlas en aplicaciones auxiliares, como sensores, sistemas de seguridad integrados, electrodomésticos inteligentes y una gran variedad de aplicaciones relacionadas con dicho mundo.

En este esfuerzo por agregar conectividad en sus aplicaciones, Microchip ofrece para sus líneas de microcontroladores soluciones basadas en esta familia de productos Wi-Fi I/O de ZeroG.

MICROCHIP CONFIGURATION WIZARD PARA WIFI

En capítulos anteriores vimos cómo utilizar esta herramienta para configurar los distintos aspectos y funciones que presenta el **stack de TCP/IP** para los microcontroladores PIC. Ahora, estudiaremos de qué manera configurarlo para que nuestras aplicaciones cuenten con conectividad WiFi.

Comenzamos ejecutando el programa Microchip Configuration Wizard y procedemos a configurar el stack de TCP/IP. Primero, indicamos el directorio donde se encuentra el proyecto general. Si descargamos la librería de soluciones de Microchip, podemos buscarlo donde se instala por defecto la carpeta **TCPIP WiFi Demo App**. Allí encontramos los archivos fuente y las plantillas de las páginas web que emplearemos.

Recordemos que la elección de las distintas funciones que puede ejecutar el stack TCP/IP de Microchip se realiza desde el archivo **TCPIPConfig.h**. La utilidad del programa Microchip Configuration Wizard reside en la facilidad de llevar a cabo estos cambios por medio de una interfaz gráfica, pero en realidad, no se está haciendo otra cosa que modificar este archivo. Podemos ver el contenido de este archivo ingresando en la carpeta **TCPIP WiFi Demo App** (Figura 10). A continuación, nos encontramos con la página donde podemos seleccionar las diferentes funciones que soporta el **stack**. Allí debemos asegurarnos de que estén seleccionadas las pestañas para las funciones de servidor web y el cliente de e-mail. Luego, en la opción de conectividad inalámbrica (*wireless*), elegimos el módulo que venimos estudiando, el **ZG2100** (Figura 11).

Es importante observar que, a diferencia de lo que hacíamos con Ethernet, el microcontrolador solo se

Las redes WiFi extendieron su uso y hoy es posible encontrarlas en sistemas de seguridad integrados

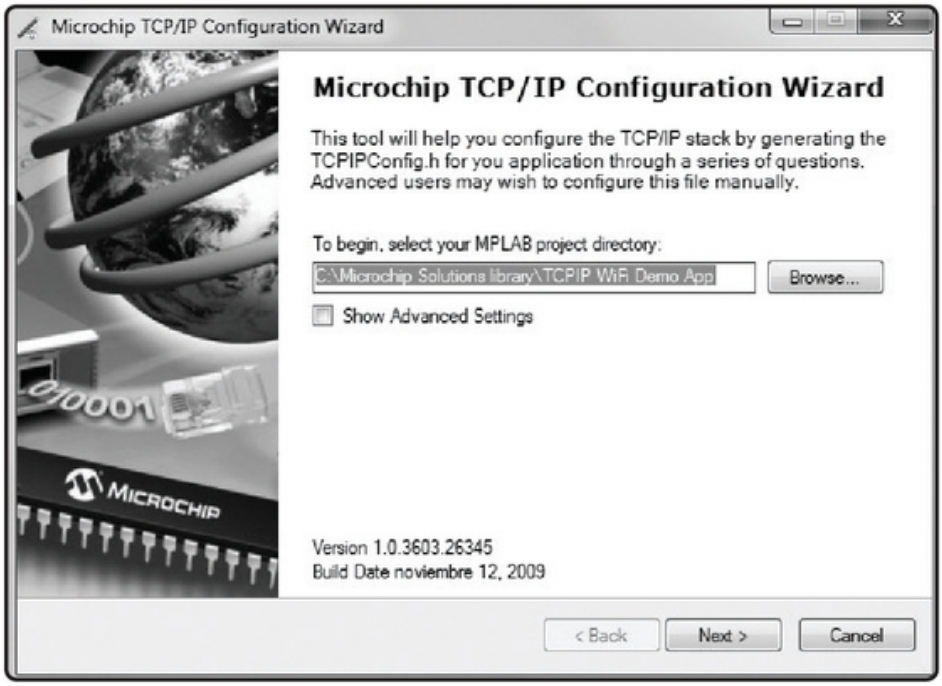


FIGURA 10.
Ejecutamos el Microchip Configuration Wizard **para configurar el stack de TCPIP, y lo haremos sobre el proyecto** `TCPIP WiFi Demo App`.

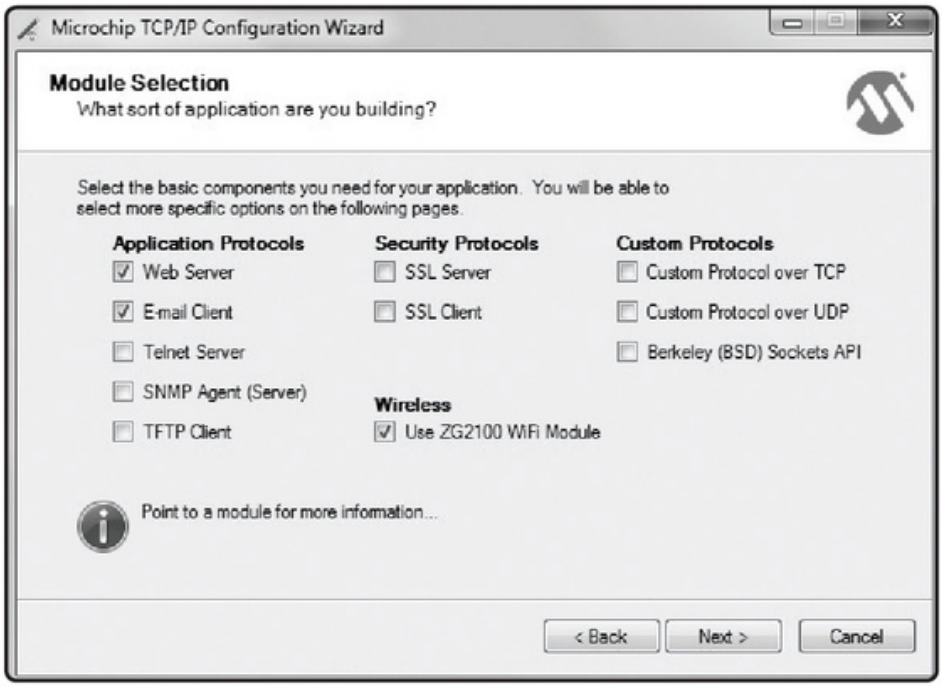


FIGURA 11.
Selección de las funciones que se utilizarán del stack y activación del casillero `wireless`.

encarga de transportar el paquete en los protocolos de niveles superiores, y el módulo **Wi-Fi** de las capas 1 y 2 del modelo OSI; es decir, que tiene el manejo de la capa de enlace y de la capa física. En la próxima pantalla de configuración, seleccionamos **`TCP Performant Test`**, que es la utilidad que nos permite generar **pings**, de gran utilidad a la hora de encontrar errores en las redes.

ASIGNACIÓN DE LA DIRECCIÓN MAC PREGRABADA

En el siguiente cuadro de diálogo indicamos el nombre de host con el cual se identificará el servidor en la red donde esté trabajando. Luego, tenemos que señalar la dirección MAC del dispositivo.

Recordemos que esta debe ser única para nuestro equipo; de lo contrario, podrían generarse errores en la red. En este aspecto debemos destacar cuáles son las fuentes donde el stack puede tomar una dirección MAC. Básicamente, existen **tres**. Una de ellas es la dirección **MAC** del módulo **ZeroG**, grabada desde fábrica, de modo que nos garantiza que sea única. Las direcciones MAC se encuentran organizadas de modo tal de evitar que dos dispositivos tengan la misma. Es así que la **IEEE** asigna a los fabricantes los primeros **tres octetos** (24 bits) que componen la MAC, llamados **OUI** (Organizationally Unique Identifier), ver **Figura 12**.

La MAC que posee el módulo **ZeroG** puede verse en la etiqueta ubicada en la parte superior de este. Si la revisamos, podemos ver que la **OUI** que identifica a los dispositivos de la empresa ZeroG es el siguiente: **00:1E:C0**.

Cada OUI garantiza al fabricante la posibilidad de registrar **2^24** dispositivos. Otro ejemplo es la OUI de las herramientas de Microchip, que es **00:04:A3** y se muestra por defecto en el **Configuration Wizard**.

Una **segunda fuente** desde la cual el stack TCP/IP puede obtener una dirección MAC es una imagen de código programado. La **tercera** es un valor que se almacena en la memoria **EEPROM**.

En tiempo de ejecución, el stack crea una estructura en memoria **RAM** donde se almacena la dirección MAC válida que se utilizará en el período de

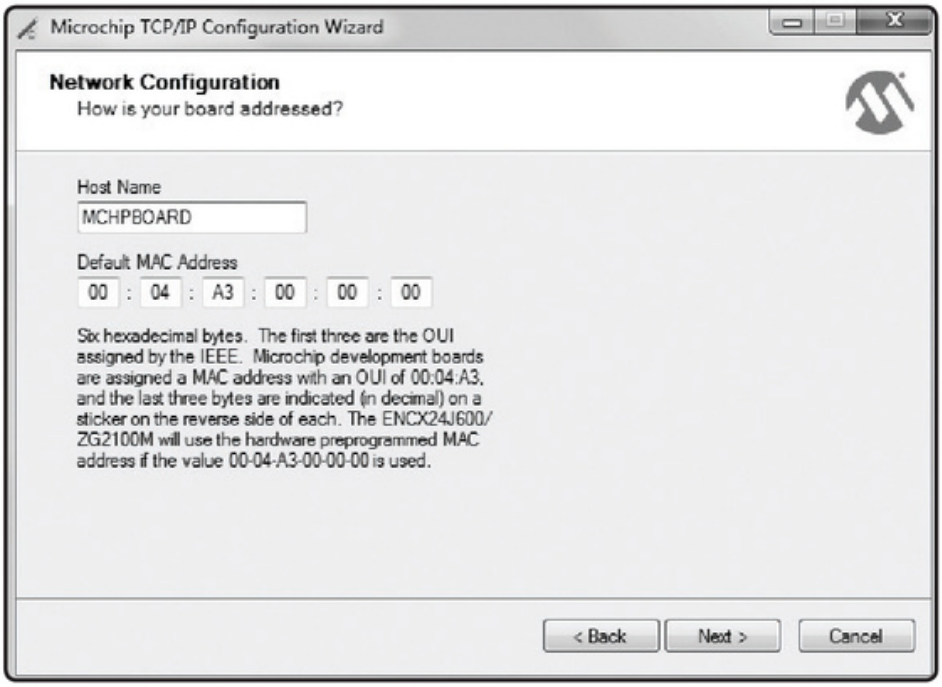


FIGURA 12. **Asignación de la dirección MAC que utilizará el dispositivo; recordemos que debe ser única en la red.**

sesiones. Entonces, el código revisa si hay una estructura de datos validos en la **EEPROM**. Si existe, se utilizan estos valores, y se ignoran los que puedan estar programados dentro del código en tiempo de compilación. Pero si no se encuentran la MAC en la EEPROM, se emplea la que se coloca durante la configuración del stack, es decir, la especificada en el **Configuration Wizard**.

Si el valor que colocamos en esa zona del cuadro de diálogo es **00:04: A3: 00:00:00**, le estamos indicando al stack que la MAC por utilizar es la que está grabada en el módulo de **conectividad ZG2100**. De lo contrario, se utilizará la que le indiquemos en la configuración del stack.

PRIORIDAD DE LAS DIRECCIONES

En términos de prioridad de las direcciones que se utilizarán para el dispositivo, la **MAC** que se encuentra grabada en la **EEPROM** es la primera, y ningún otro valor tendrá precedencia sobre ella (**Figura 13**).

La dirección MAC que se encuentra grabada en la EEPROM es la que siempre tiene mayor prioridad

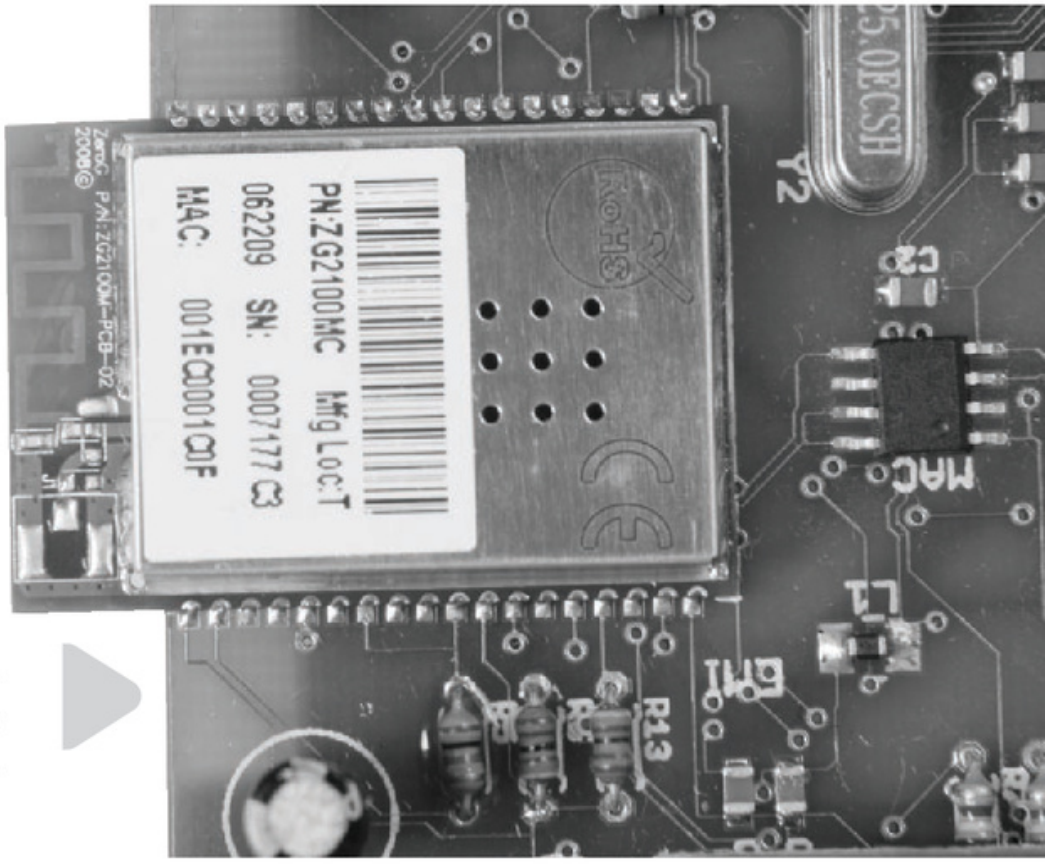


FIGURA 13.
La dirección MAC del módulo, está especificada en la etiqueta.

CONEXIÓN A UNA RED WIFI EXISTENTE

En el siguiente cuadro de diálogo del Configuration Wizard, tenemos las opciones para la conexión a la red inalámbrica. En **SSID** (*Service Set IDentifier*) colocamos el nombre de la red a la que se conectará el servidor web. Si ya existe una red, debemos indicar aquí el nombre que la identifica, dato disponible desde el *access point* de la red. Si no tenemos una red preexistente, colocamos el nombre que queremos darle. En este lugar, también podemos indicar cuál es la estructura de la red a la cual se va a conectar.

Habíamos visto que teníamos, básicamente, dos estructuras para la conexión: ad-hoc y de infraestructura. El módulo permite la conexión en ambos tipos de redes. Por defecto, el stack viene configurado para conectarnos a una red de infraestructura. Si en realidad queremos conectarnos a una red ad-hoc debemos seleccionar **Include Ad-HOC Manager**.

CANAL DE RADIO FRECUENCIA

En la misma pantalla (**Figura 14**) podemos elegir el canal de radiofrecuencia que utilizará el módulo. En la imagen vemos que, por defecto, están

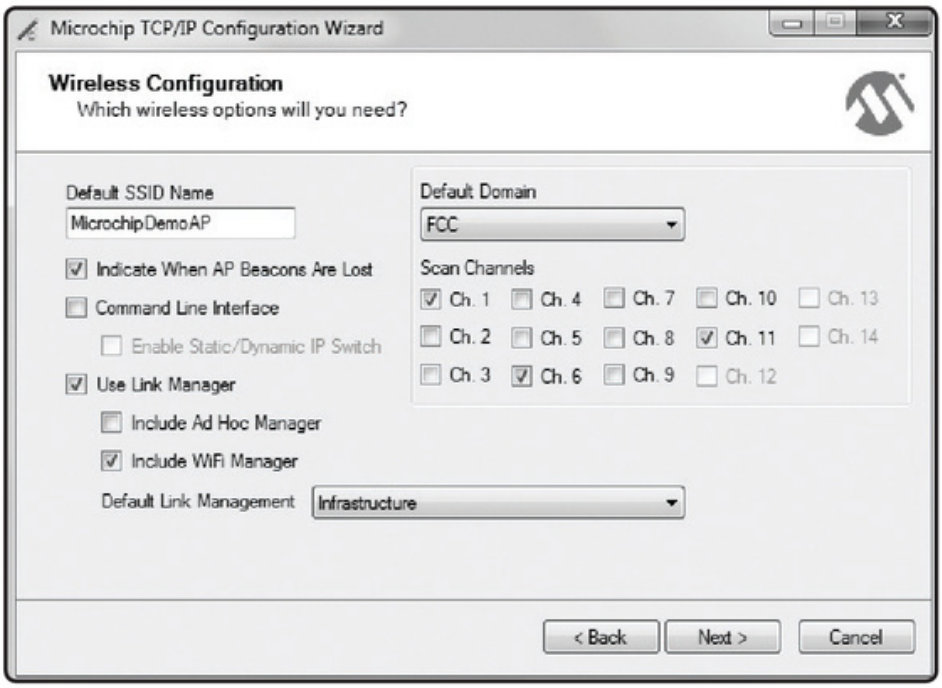


FIGURA 14.
Cuadro de diálogo donde colocaremos el nombre de la red a la que se conectará la aplicación.

SOBRE PROTOCOLOS

La mayoría de los protocolos que trabajan en la capa 2 del modelo OSI usan una de las tres numeraciones manejadas por el IEEE: **MAC-48**, **EUI-48** y **EUI-64**. Estas han sido diseñadas para ser identificadores globalmente únicos.

seleccionados tres canales: 1, 6 y 11. El módulo ZeroG realizará, entonces, el análisis sobre ellos para detectar actividades de RF. Es importante saber que, a mayor cantidad de canales, más se demorará el módulo en conectarse.

SEGURIDAD Y ENCRIPTACIÓN DE DATOS

En el cuadro de diálogo **Wireless Security** podemos adicionar seguridad a la conexión que establecerá nuestra aplicación con el access point. Podemos elegir entre WEP o WPA.

Seguridad WEP

WEP (*Wired Equivalent Privacy*) es un mecanismo de encriptación para las redes inalámbricas. La mayoría de los access points aplican dos tipos de seguridad WEP: la **de 64 bits** utiliza una clave de **40 bits** (WEP-40), en tanto que la de **128 bits** emplea una clave de **104 bits** (WEP-104).

Por defecto, el stack viene configurado para conectarnos a una red de infraestructura

En la forma más básica, las claves WEP son valores hexadecimales de solo 5 bytes para WEP-40 y de 13 bytes para WEP-104 (**Figura 15**). Algunos access points tratan de incrementar el carácter aleatorio de la clave WEP mediante la adición de una capa extra que convertirá una contraseña **ASCII** en una clave **hexadecimal**. El módulo ZeroG requerirá una clave hexadecimal, sin importar qué camino se utilice para generarla. Para habilitar la seguridad WEP entre la placa de aplicación y el access point, habrá que configurar este último para dicha seguridad, y el stack de la placa de desarrollo, con las claves WEP correctas.

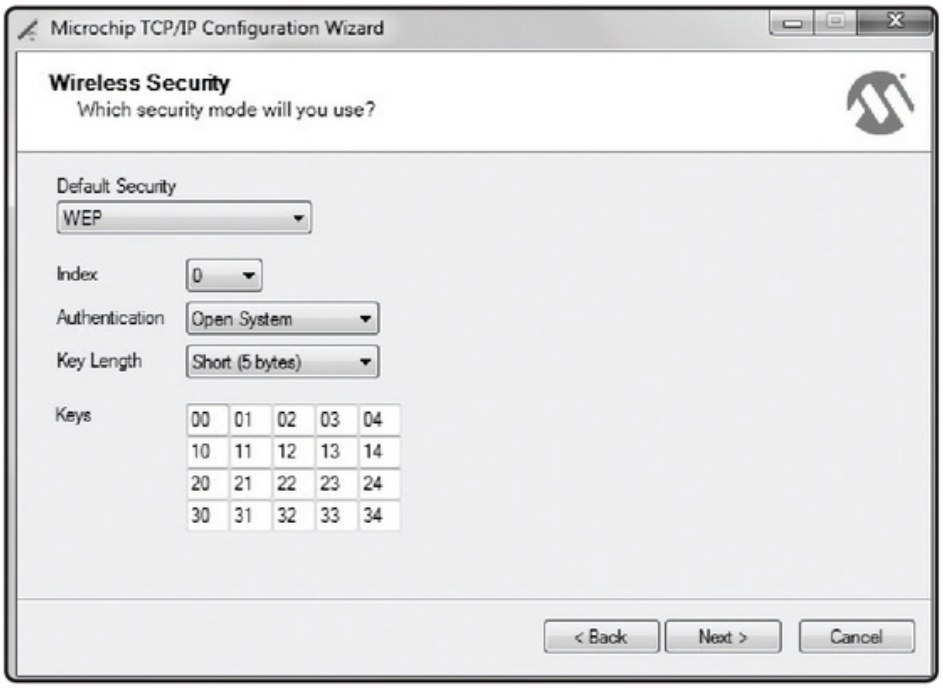


FIGURA 15.
Configuración para la seguridad WEP, donde se definen las claves de 5 bytes. Estas deben ser las mismas que posee el router o access point.

Seguridad WPA/WPA2

WPA y WPA2 son modos de seguridad que implementan la especificación 802.11i. Son más seguros que las redes de encriptación WEP. Utilizan el SSID y la contraseña de usuario para generar el PSK (*Pre-Shared Key*), donde las estaciones y el access point tienen, previamente, una clave compartida.

El módulo ZeroG posee dos maneras diferentes de calcular la PSK para su uso en WPA/WPA2. En una de ellas, el usuario final puede programar la clave en hexadecimal de 32 bytes en el código fuente o en el Configuration Wizard (Figura 16). En la otra, el módulo ZeroG puede calcular la clave hexadecimal de 32 bytes internamente. Debemos tener en cuenta que si el módulo ZeroG tiene que calcular la clave, esto puede añadir un tiempo extra de 30 a 45 segundos en el tiempo de conexión, para dar lugar a que el chip calcule el valor.

Configuración del router

Para que nuestra aplicación pueda conectarse a la red, debemos configurar el router para que exista una coherencia con la configuración de la aplicación; de lo contrario, no podremos conectarnos a la red. Es importante aclarar en esta situación particular que los términos access point y router serán utilizados como sinónimos, debido a que estos dos elementos de la red se encuentran combinado en un solo equipo.

Si contamos con una red hogareña con Wi-Fi, debemos acceder al router y revisar algunas configuraciones para establecer la comunicación con nuestra aplicación. Comenzamos por acceder a la página principal del router, para lo cual abrimos el navegador y colocamos la dirección http://192.168.1.1.

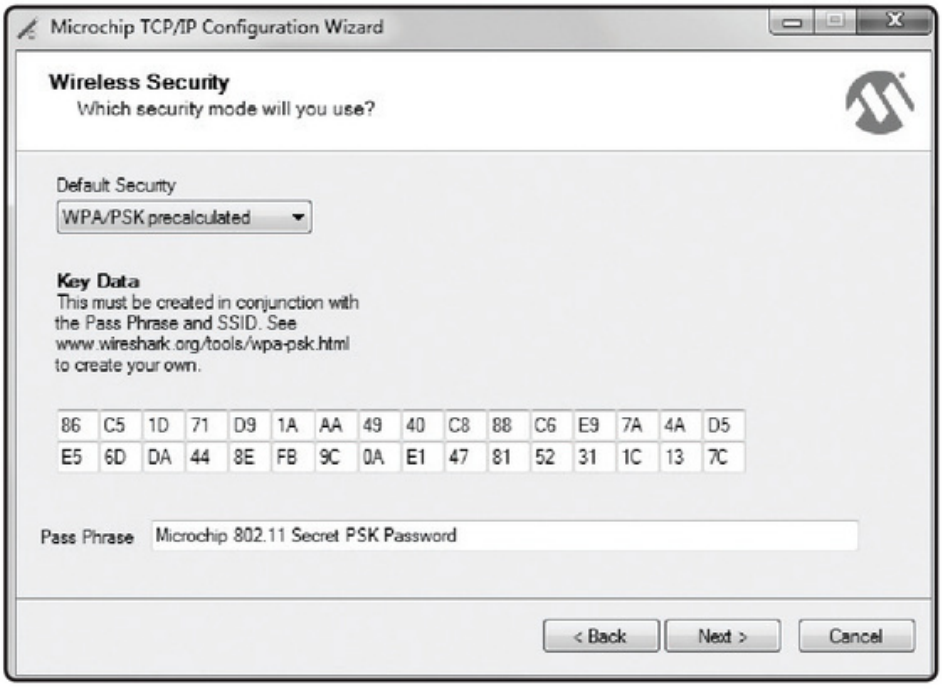


FIGURA 16.
En la seguridad WPA podemos indicarle el valor de la clave o decirle al módulo que lo calcule.

Esta se utiliza por defecto en muchos routers hogareños, ya que se encuentra dentro del rango de direcciones privadas que especifica IP. Si el router del cual disponemos no utiliza esta dirección para ingresar en su página principal, debemos acudir al manual de usuario del dispositivo. Veremos que se abrirá el asistente de configuración web del router. Luego, colocamos el nombre de usuario y la contraseña.

Si contamos con una red hogareña con Wi-Fi, debemos acceder al router y revisar algunas configuraciones

CONFIGURACIONES BÁSICAS

Cuando accedemos al router, aparece la imagen que vemos en la Figura 17. La mayoría de las configuraciones básicas que vienen por defecto en el router sirven para nuestros ejemplos. Lo único que vamos a verificar es que el router esté configurado como un servidor DHCP. Para hacerlo, vamos al casillero **Local DHCP Server** y lo colocamos en **Enable** para habilitarlo. Con esto permitiremos que el router realice la gestión y la asignación de las direcciones IP de todos los equipos que estén conectados en la red.



FIGURA 17.
Pantalla principal del router, donde vemos las configuraciones básicas.

SOBRE LOS ROUTERS

Utilizamos un router de la marca Linksys; si contamos con uno diferente, tal vez las capturas de pantallas que mostramos en estas páginas sean distintas. De todas maneras, las configuraciones básicas están soportadas por la mayoría de los dispositivos de ruteo hogareños.

Configuración de la red inalámbrica

A continuación, seleccionamos la solapa **Wireless** para configurar los aspectos de la red inalámbrica. Allí, podemos colocar el nombre de la red en la solapa **SSID**. En este caso, ponemos **mobile**, que es el nombre de la red que estamos utilizando actualmente en estos ensayos. También podemos elegir la velocidad de comunicación y el canal RF donde trabajará el router. El modo de trabajo de la red lo dejamos en **Mixed**.

Configuración de la seguridad

Una vez que configuramos los aspectos generales de la red, procedemos a establecer la seguridad. Cuando estudiamos la configuración del **stack**, sabemos que podemos elegir entre dos modos de seguridad, **WEP** y **WPA** (**Figura 18**).

Recordemos que WPA posee más seguridad que WEP, pero queda en nosotros elegir el que queremos para nuestras aplicaciones. En este caso, vamos a seleccionar la seguridad WEP. Para hacerlo, tenemos que elegir la solapa **Wireless Security** y, en el modo de seguridad, elegir **WEP**.

En cuanto al tipo de autenticación, lo dejamos en sistema abierto (*Open System*). Luego, seleccionamos qué tipo de seguridad **WEP** queremos. Habíamos visto que esta puede tener una encriptación de 64 bits, con claves de **40 bits**, o de **128 bits**, con una clave de **124 bits**. En este caso, vamos a seleccionar la seguridad WEP de 64 bits.

En la parte inferior de la página de configuración de seguridad del router, vemos las claves generadas por el dispositivo para la autenticación de los equipos que se conectan a la red. Estas tienen un tamaño de **5 bytes**. Vemos por la configuración del router que la clave por defecto es la que se encuentra en el **índice 1**. Ella debe coincidir con la que se encuentra en el **índice cero** de la configuración del



FIGURA 18. **Configuraciones de seguridad de la red inalámbrica, donde elegimos una de tipo WEP.**

stack de TCP/IP. Esta diferencia en el índice se debe a que en el router se comienza a contar desde el 1 (uno), y en el stack, a partir del 0 (cero). De todas maneras, a lo que debemos prestar atención es que la clave que está escrita en el primer renglón sea igual a la que se configuró en el stack (Figura 19).

MÓDULOS DE ENTRENAMIENTO

Un desarrollador que tiene conocimientos de programación en microcontroladores PIC puede crear aplicaciones con conectividad Wi-Fi de manera rápida, aprovechando las ventajas que ofrece un módulo de entrenamiento. Estos módulos nos permiten hacer desarrollos de manera mucho más rápida, porque nos dan la posibilidad ensayar y depurar código con un hardware que sabemos que funciona bien.

Además, con estos módulos de enseñanza contamos con la documentación necesaria para realizar aplicaciones con conectividad de manera eficiente.

Una vez que configuramos los aspectos generales de la red, procedemos a establecer la seguridad

PLACA MCE WEBSERVER WIFI

Esta placa de entrenamiento (Figura 20) incorpora en el mismo PCB un microcontrolador PIC18F97J60 y el módulo de conectividad ZG2100M (Figura 21), por lo que es ideal para ejercitar los temas que vimos en esta obra. Además, no solo cuenta con conectividad inalámbrica; gracias a que incorpora un microcontrolador PIC con drivers para Ethernet, también podemos conectarnos a una red cableada.

Esto nos da mucha flexibilidad en el momento de aprender las diferentes opciones de conectividad.

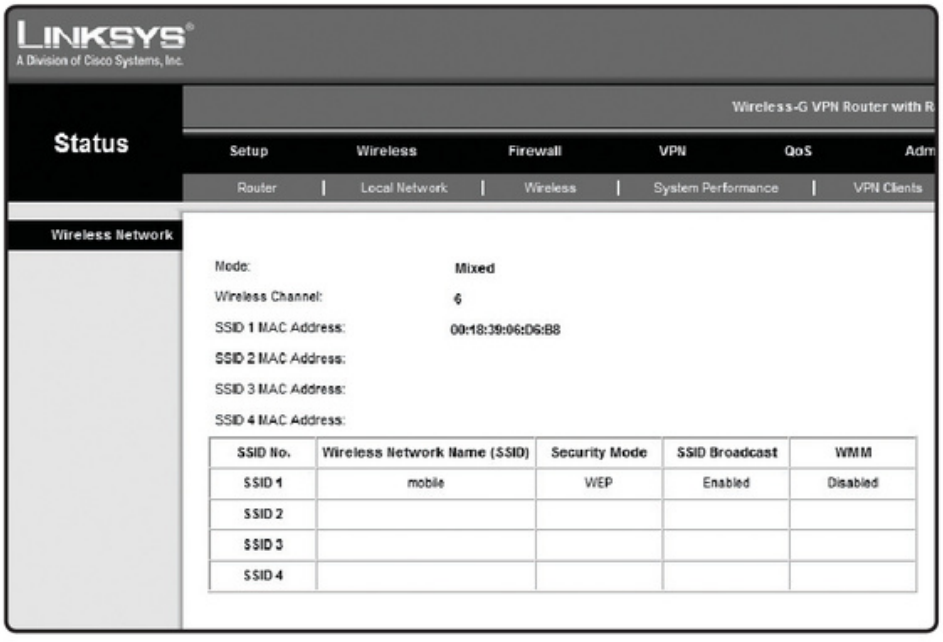


FIGURA 19.
Configuración final del router para la red inalámbrica.

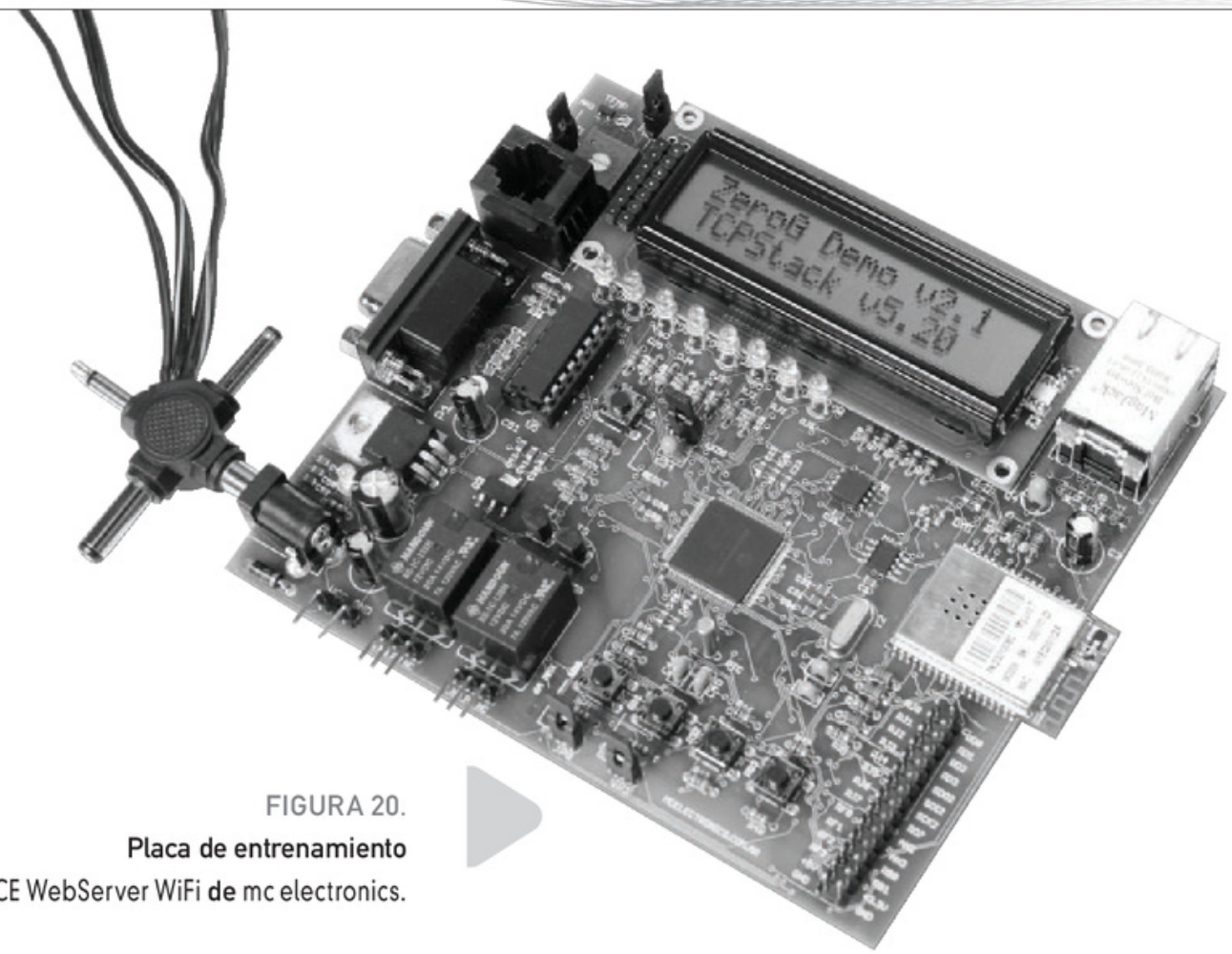


FIGURA 20.
Placa de entrenamiento
MCE WebServer WiFi **de** mc electronics.

La placa cuenta con varios periféricos para realizar un sistema integral de monitoreo y control a distancia, además de ofrecer una interfaz para el usuario con un LCD incorporado.

Posee una memoria **EEPROM 25LC1024** externa para alojar una página web, y una **MAC Address** Chip que viene con una MAC pregrabada. Esto nos asegura que la aplicación tenga una dirección única.

En cuanto al software, utiliza el stack TCP/IP versión 5.2 corriendo en el PIC, ya preparado para conectarse a una red WiFi. Más información sobre este tema, en **www.mcelectronics.com.ar**.

mc electronics®

CONTROL REMOTO DE UN MOTOR A TRAVÉS DE WIFI

Con todo lo que aprendimos hasta el momento sobre conectividad con PIC, vamos a concluir realizando el control de un motor a través de una red inalámbrica, en el **Paso a paso 1**. Esto nos da la posibilidad de que el operador monitoree el estado de una planta desde una PC a distancia, que puede encontrarse en otra habitación, en otro edificio o en otra ciudad.

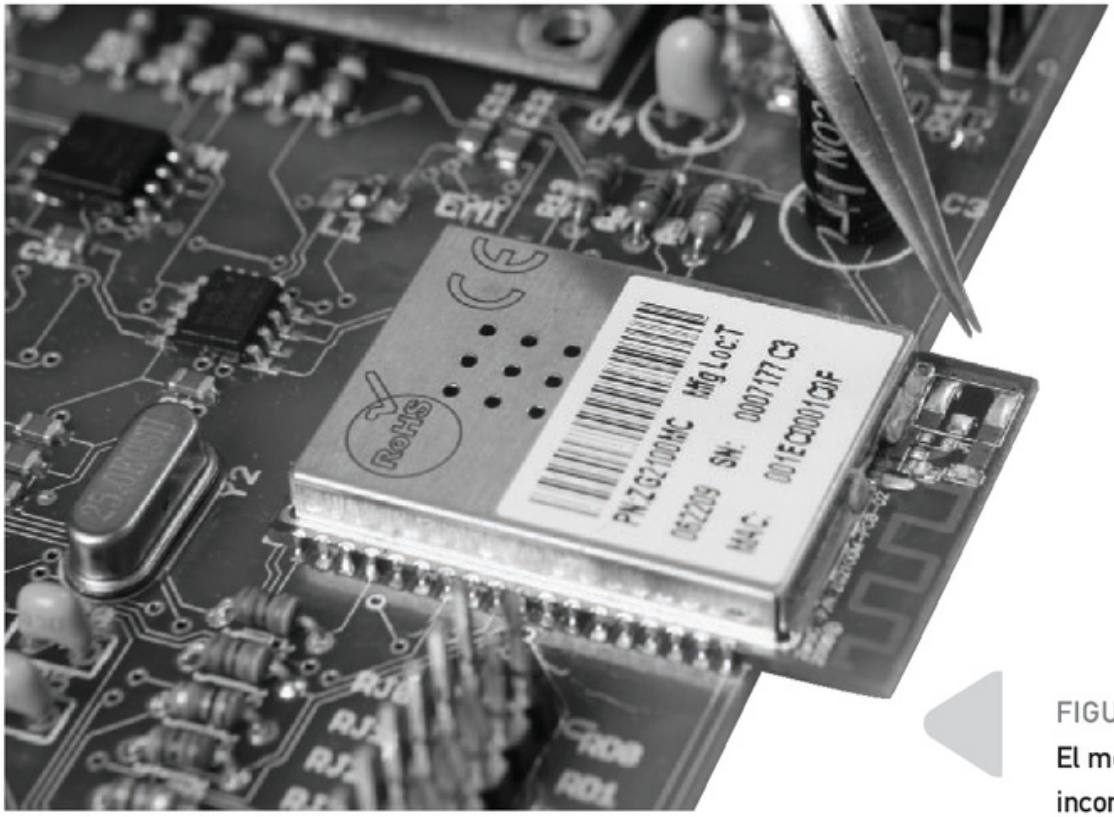
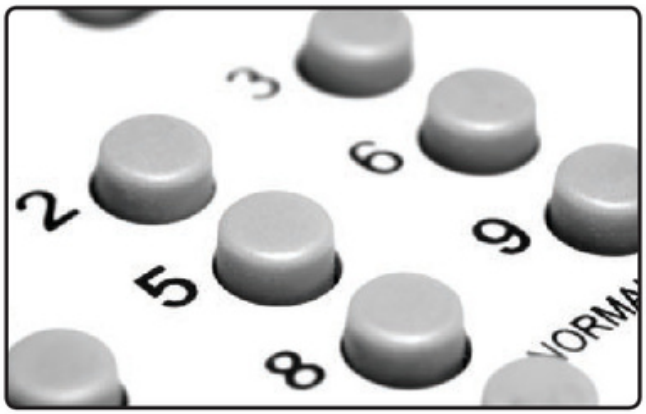


FIGURA 21.
El módulo WI-FI ZG2100, **incorpora la placa.**

El control de un motor de esta manera tiene muchas aplicaciones en el ámbito industrial, debido a que posee infinidad de aplicaciones.

En la **Figura 22**, podemos ver el esquema de la práctica donde se muestra el PIC conectado al módulo ZG2100 a través de la conexión SPI. Además, figuran los pines de control que accionarán el relay que manejará el encendido del motor.



802.11B

En abril del año 2000, se estableció la primer norma 802.11b, que utilizaba la norma la banda de 2,4 GHz y alcanzaba una velocidad de 11 Mbps. Fue creada y mantenida por la IEEE (*Institute of Electrical and Electronics Engineers*) LAN/MAN.

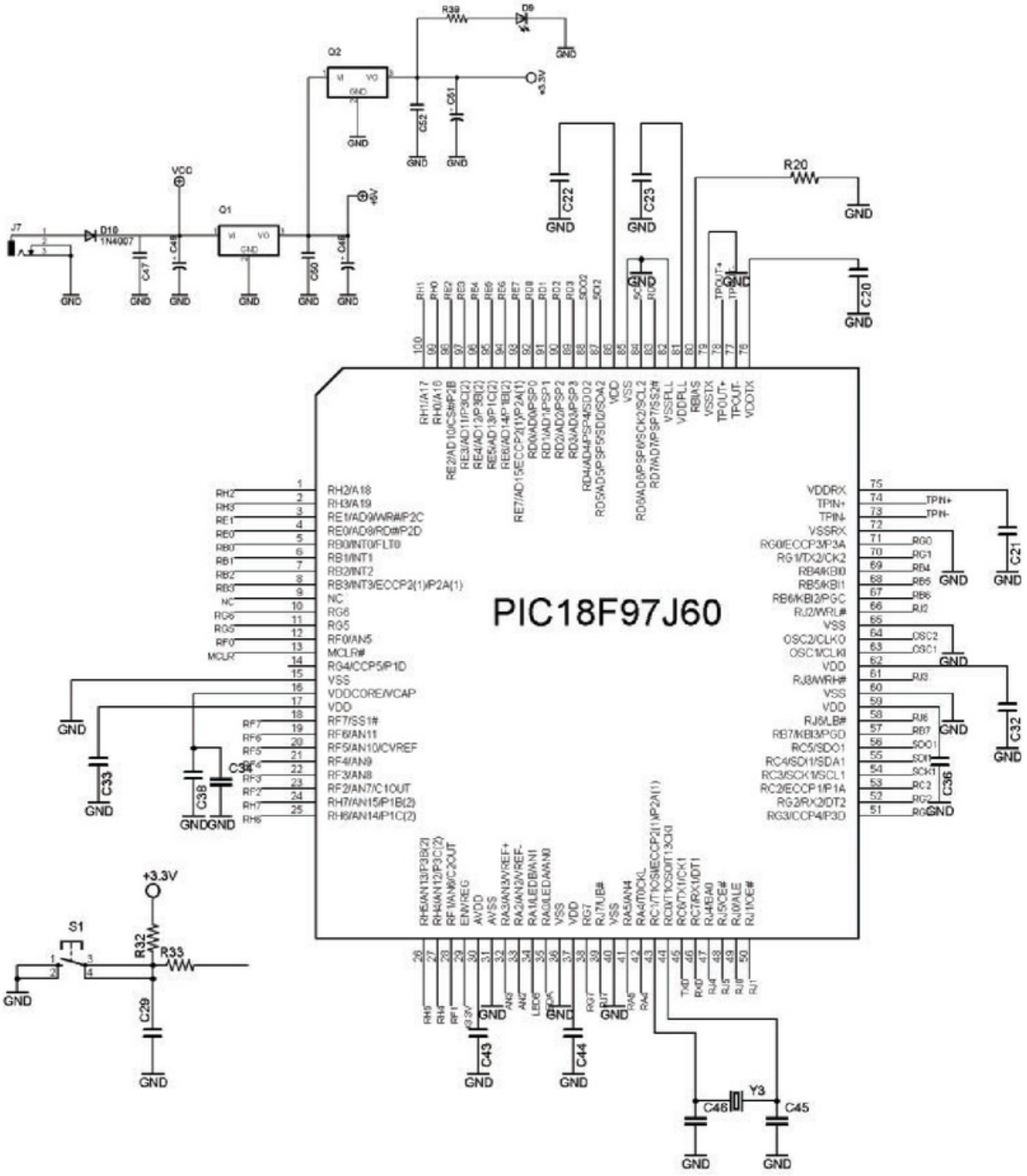
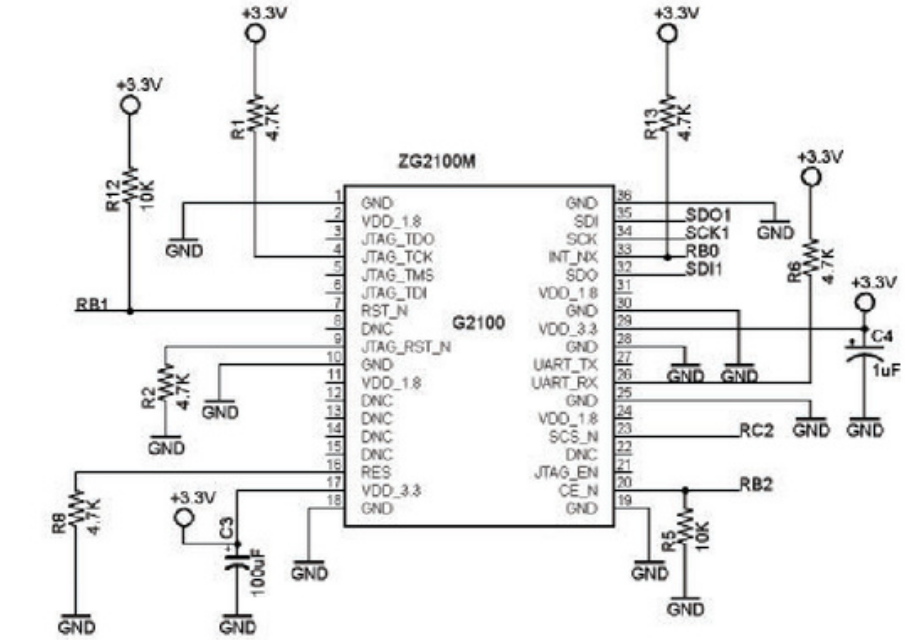
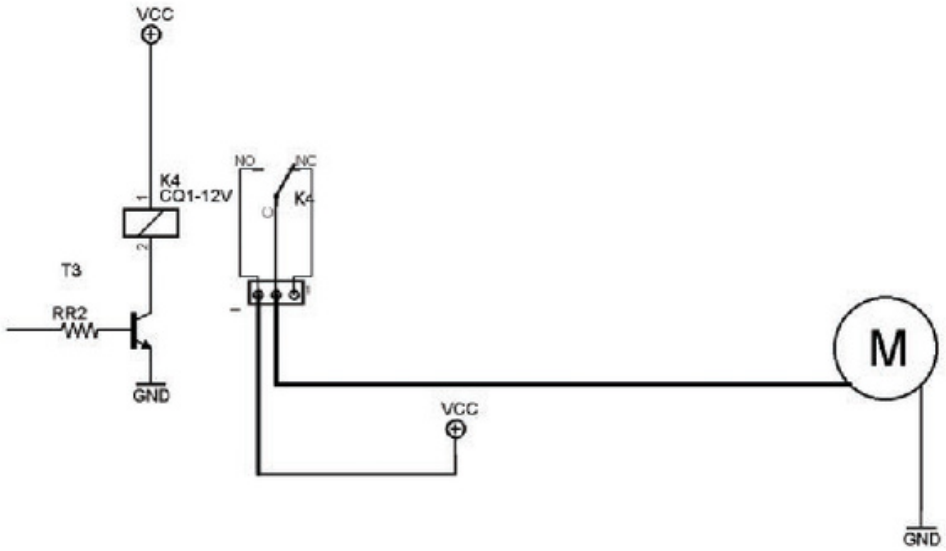


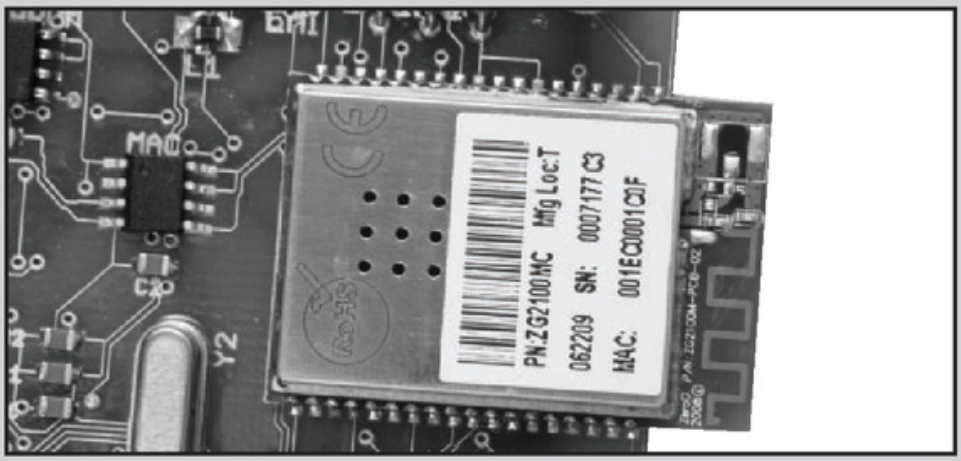
FIGURA 22. **Circuito de alimentación: tanto el PIC como el módulo de conectividad se alimentan a 3,3 V.**


+3.3V
R1
4.7K
R12
10K
R13
4.7K
ZG2100M
G2100
GND
VDD_1.8
JTAG_TDO
JTAG_TCK
JTAG_TMS
JTAG_TDI
RST_N
DNC
JTAG_RST_N
VDD_3.3
UART_TX
UART_RX
SCS_N
JTAG_EN
CE_N
SDO1
SCK1
RB0
SDI1
RB1
RC2
RB2
R2
4.7K
R6
4.7K
C4
1uF
R8
4.7K
C3
100uF
R5
10K


VCC
K4
CQ1-12V
NO
NC
T3
RR2
M
GND

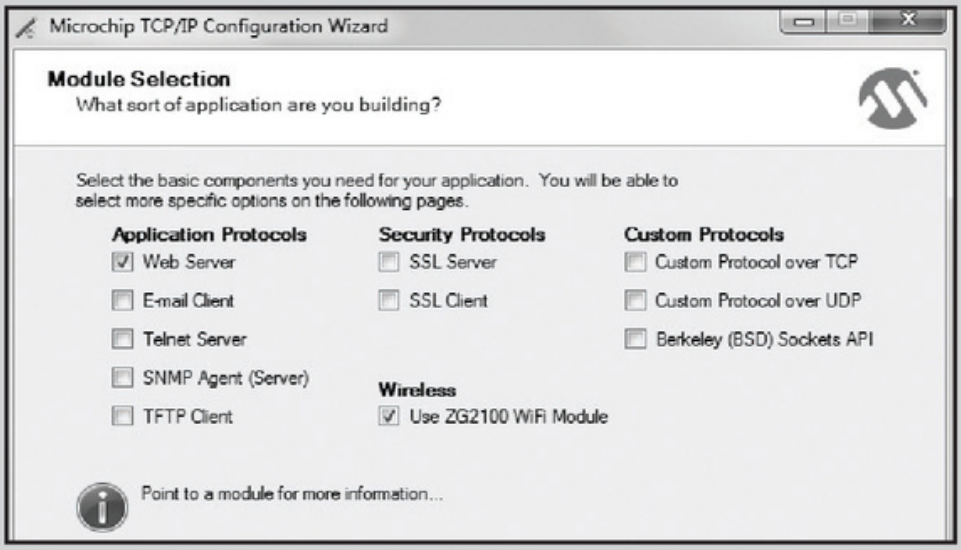
PASO A PASO /1
Control remoto de un motor a través de Wi-Fi

1



En el momento de diseñar el PCB que contiene los dispositivos, debe recordar las consideraciones que vio sobre el layout. No hay que ubicar componentes cerca de las proximidades de la antena Wi-Fi, y también es importante colocar bien alejado (lo más posible) el relay del módulo ZG2100.

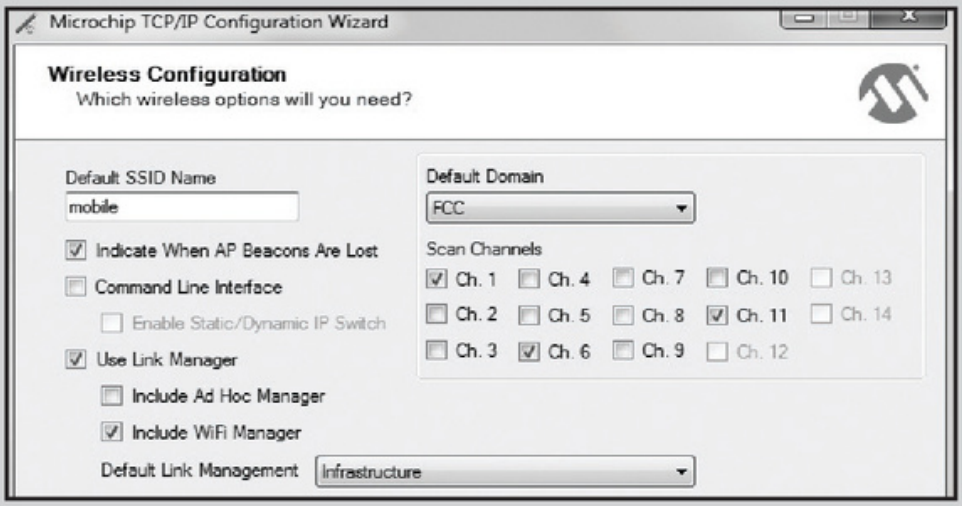
2



Configure el stack de TCP/IP para conectarlo a la red Wi-Fi. En el directorio, seleccione la carpeta **`TCPIP WIFI Demo App`**, y utilice las funcionalidades de Web Server y la conexión mediante el módulo ZG2100 que estudió en esta obra.

PASO A PASO /1 (cont.)

3



Luego, en el cuadro de dirección MAC, deje la que viene por defecto, para indicar que se utilizará la que está grabada en el módulo. A continuación, coloque el SSID de la red donde trabaja; en este caso, mobile.

4



Así como indica el nombre de la red en el stack, en el router debe tener una red con el mismo SSID. Esto es importante, porque si los SSID son diferentes, no habrá comunicación alguna entre el módulo y el router.

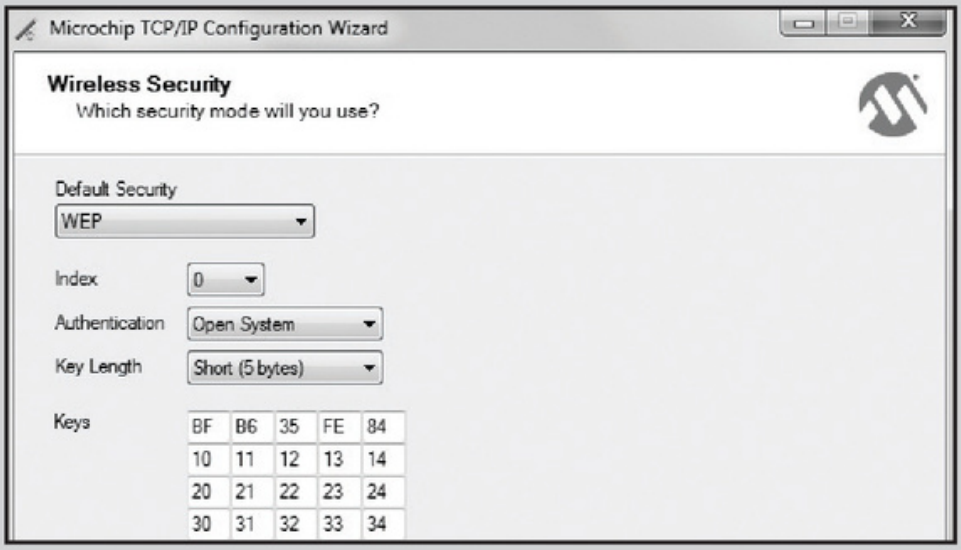
PASO A PASO /1 (cont.)

5



Establezca la seguridad en su red inalámbrica. Para eso, en el router vaya hacia la zona de seguridad inalámbrica y seleccione **Seguridad WEP** en el formato de 64 bits. Esto le entregará las claves de autenticación del dispositivo.

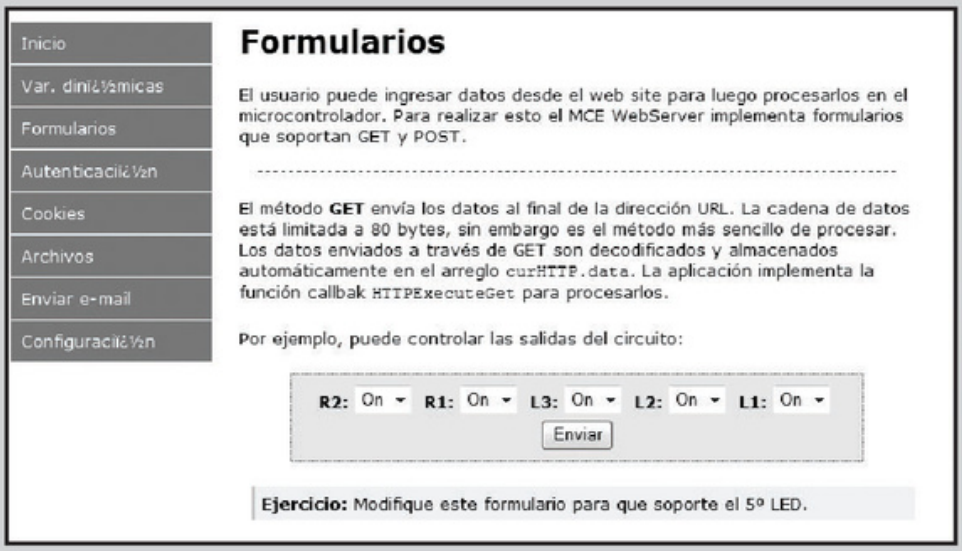
6



La clave que obtenga del router deberá copiarla en el stack de TCP/IP.
Si pone el índice en **0**, copiará la clave que se encuentra en el primer renglón del router, al primer renglón que tiene de la configuración del stack.

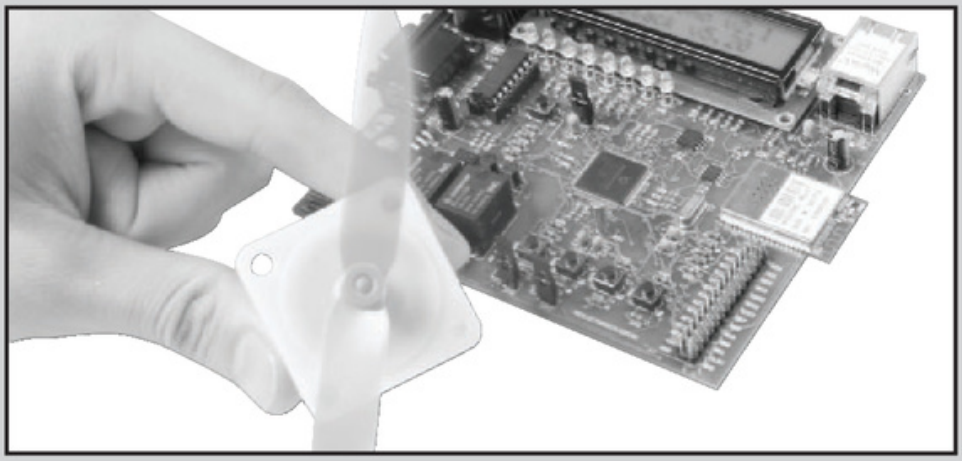
PASO A PASO /1 (cont.)

7



Este es el formulario web para el control del relay al cual se conecta el motor. Va a utilizar lo que aprendió del método GET para el manejo de este elemento. Usará el mismo formulario web que vio en el capítulo anterior, con las mismas funciones **callback** para la excitación de los pines que controlan la bobina de accionamiento.

8



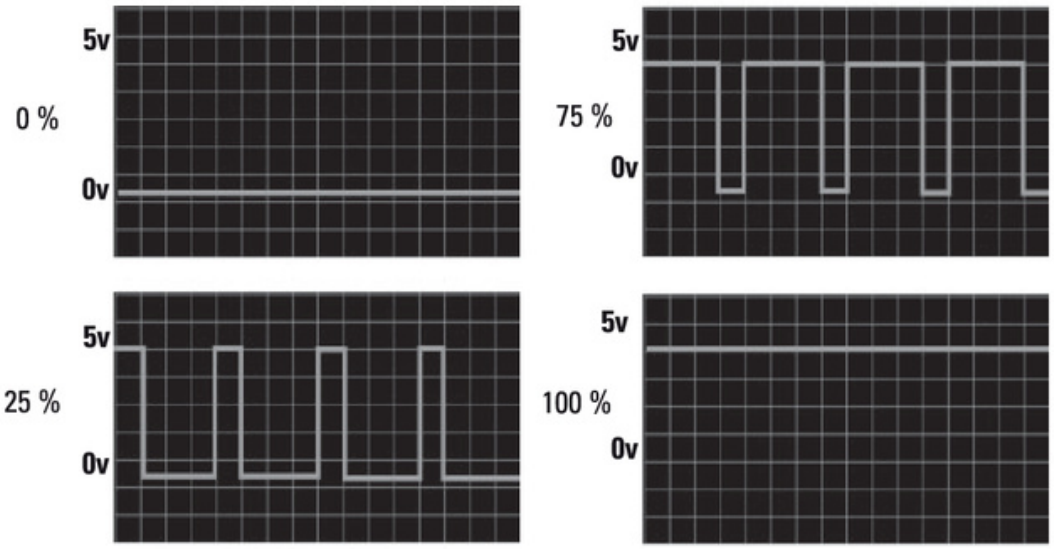
Luego de compilar el proyecto, grabe el código en el PIC y pruebe el accionamiento del relay; esto debe hacer funcionar el motor. Puede poner algunas aspas para apreciar su movimiento.

INFOGRAFÍA 2: CONTROL DE MOTOR MEDIANTE PWM

Una forma eficiente de controlar un motor es utilizar Modulación por ancho de pulso (PWM, siglas del inglés Pulse-Width Modulation). Existen muchas técnicas para lograr nuestro objetivo; en nuestro caso generaremos las señales PWM mediante un software cargado en la memoria de un microcontrolador ya que son adecuadas a los niveles de tensión que el motor necesita para funcionar mediante una placa de control basada en transistores tipo MOSFET.

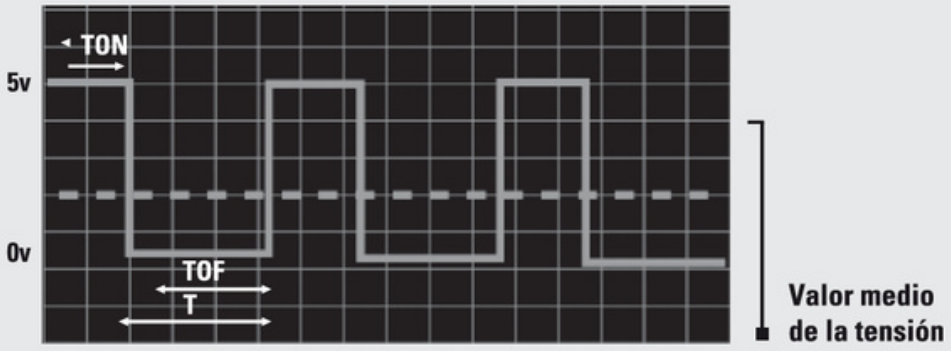
Ciclo de trabajo

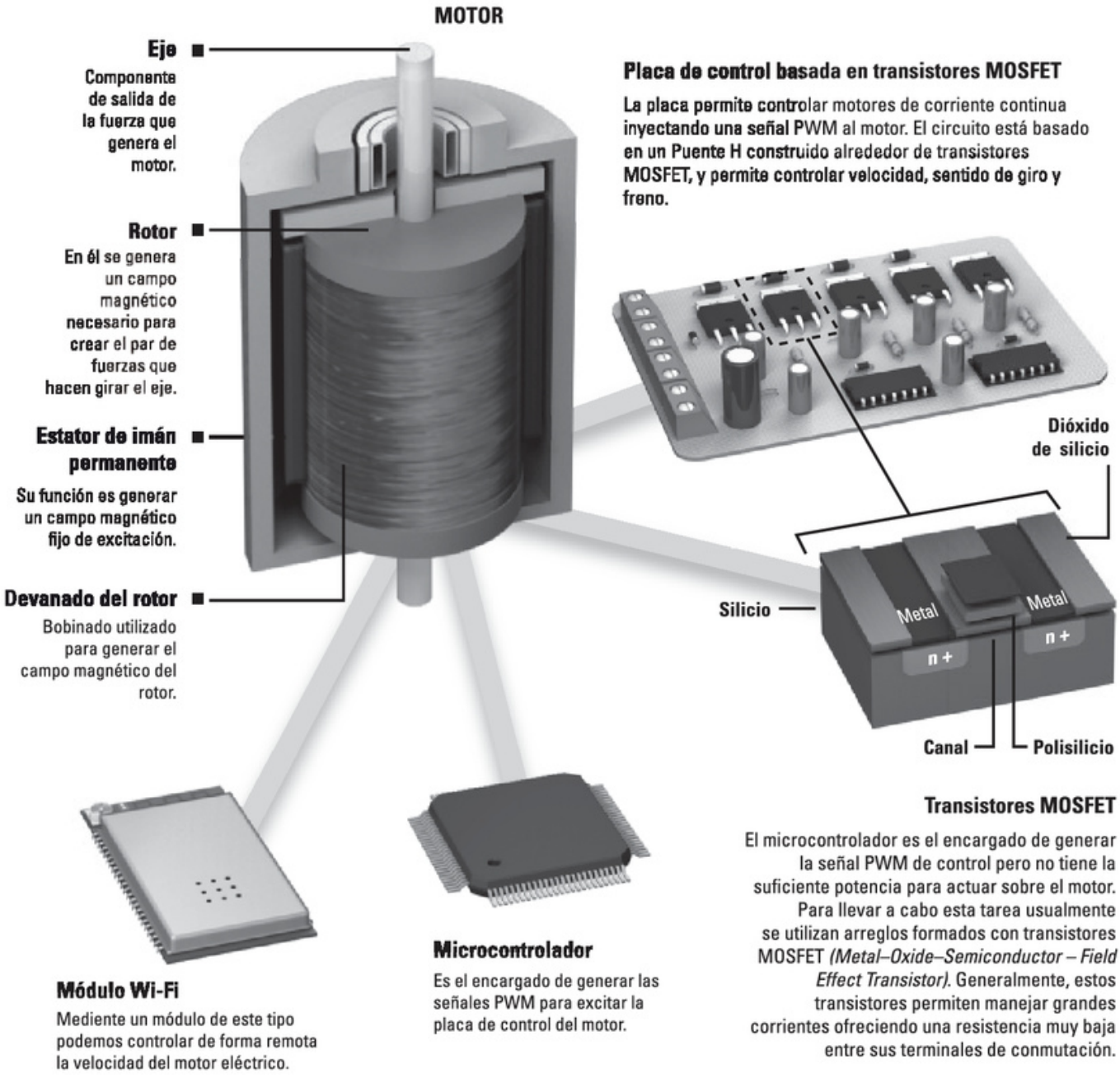
Recibe este nombre la relación de tiempos entre el estado alto y bajo de la señal utilizada. Se expresa como un porcentaje entre el período y el ancho positivo del pulso. A medida que el ciclo de trabajo se acerque al 100%, la velocidad del motor aumentará, y viceversa.



Valor medio

El valor medio de la tensión que se aplica al motor se calcula mediante la fórmula: Vm= Ton / T, siendo Ton el tiempo que la señal está a nivel alto, y T es el período de la señal (Ton + Toff).




MOTOR
Eje
Componente de salida de la fuerza que genera el motor.
Rotor
En él se genera un campo magnético necesario para crear el par de fuerzas que hacen girar el eje.
Estator de imán permanente
Su función es generar un campo magnético fijo de excitación.
Devanado del rotor
Bobinado utilizado para generar el campo magnético del rotor.
Placa de control basada en transistores MOSFET
La placa permite controlar motores de corriente continua inyectando una señal PWM al motor. El circuito está basado en un Puente H construido alrededor de transistores MOSFET, y permite controlar velocidad, sentido de giro y freno.
Dióxido de silicio
Silicio
Metal
Metal
n +
n +
Canal
Polisilicio
Transistores MOSFET
El microcontrolador es el encargado de generar la señal PWM de control pero no tiene la suficiente potencia para actuar sobre el motor. Para llevar a cabo esta tarea usualmente se utilizan arreglos formados con transistores MOSFET (Metal–Oxide–Semiconductor – Field Effect Transistor). Generalmente, estos transistores permiten manejar grandes corrientes ofreciendo una resistencia muy baja entre sus terminales de conmutación.
Microcontrolador
Es el encargado de generar las señales PWM para excitar la placa de control del motor.
Módulo Wi-Fi
Mediente un módulo de este tipo podemos controlar de forma remota la velocidad del motor eléctrico.

Multiple **choice**

1 ¿Cuál es la norma que estandariza las redes Wi-Fi?

a- IEEE 802.11.
b- Bluetooth.
c- Infrarrojo.
d- API.

2 ¿Cuál es el límite de la velocidad del módulo ZG2100M?

a- 1 Mbps.
b- 2 Mbps.
c- 3 Mbps.
d- 4 Mbps.

3 ¿Hasta cuántos pines de conexión cuenta el módulo ZG2100M?

a- 6.
b- 12.
c- 24.
d- 36.

4 ¿Cuántos bits tiene la clase de seguridad del tipo de seguridad WEP de 64 bits?

a- 40.
b- 64.
c- 104.
d- 128.

5 ¿Cuántos bits tiene la clase de seguridad del tipo de seguridad WEP de 128 bits?

a- 40.
b- 64.
c- 104.
d- 128.

6 ¿Qué especificación implementan WPA y WPA2?

a- 802.11x.
b- 802.11b.
c- 802.11e.
d- 802.11i.

Respuestas: 1-a, 2-b, 3-d, 4-a, 5-c, 6-d.

Apéndice A

Robótica



En este apéndice, explicaremos las nociones básicas de la robótica.

RedUSERS.com

Anatomía de los robots

Cuando escuchamos la palabra **robot**, muchas veces pensamos en esas películas que nos han sorprendido por ostentar máquinas que realizan acciones similares o superiores a las capacidades del ser humano. Desde hace muchos años, los humanos han tratado de reproducir el comportamiento y la forma de los seres vivos, creando distintos modelos de robots y dotándolos de sistemas cada vez más avanzados.

En **1923**, **Karel Capek** escribió la obra **Rossum's Universal Robot (R.U.R.)**, en donde surgió por primera vez la palabra robot, la cual deriva de **robotnik**, que en checo significa "siervo". A través de los años, se han construido muchos aparatos mecánicos; **Leonardo da Vinci** dedicó parte de su vida a desarrollar un león animado, aves voladoras, vehículos autopropulsados, y tantos otros. Los robots nacieron de la necesidad y del progreso tecnológico recién en **1960**, cuando **George Devol**, considerado padre de la robótica, desarrolló el primer robot industrial.

Estos aparatos pueden variar en cuanto a su forma, pero todos tienen una característica en común: la inteligencia necesaria para realizar tareas en diversos entornos, entendiéndose por inteligencia a la habilidad que poseen para resolver un problema.



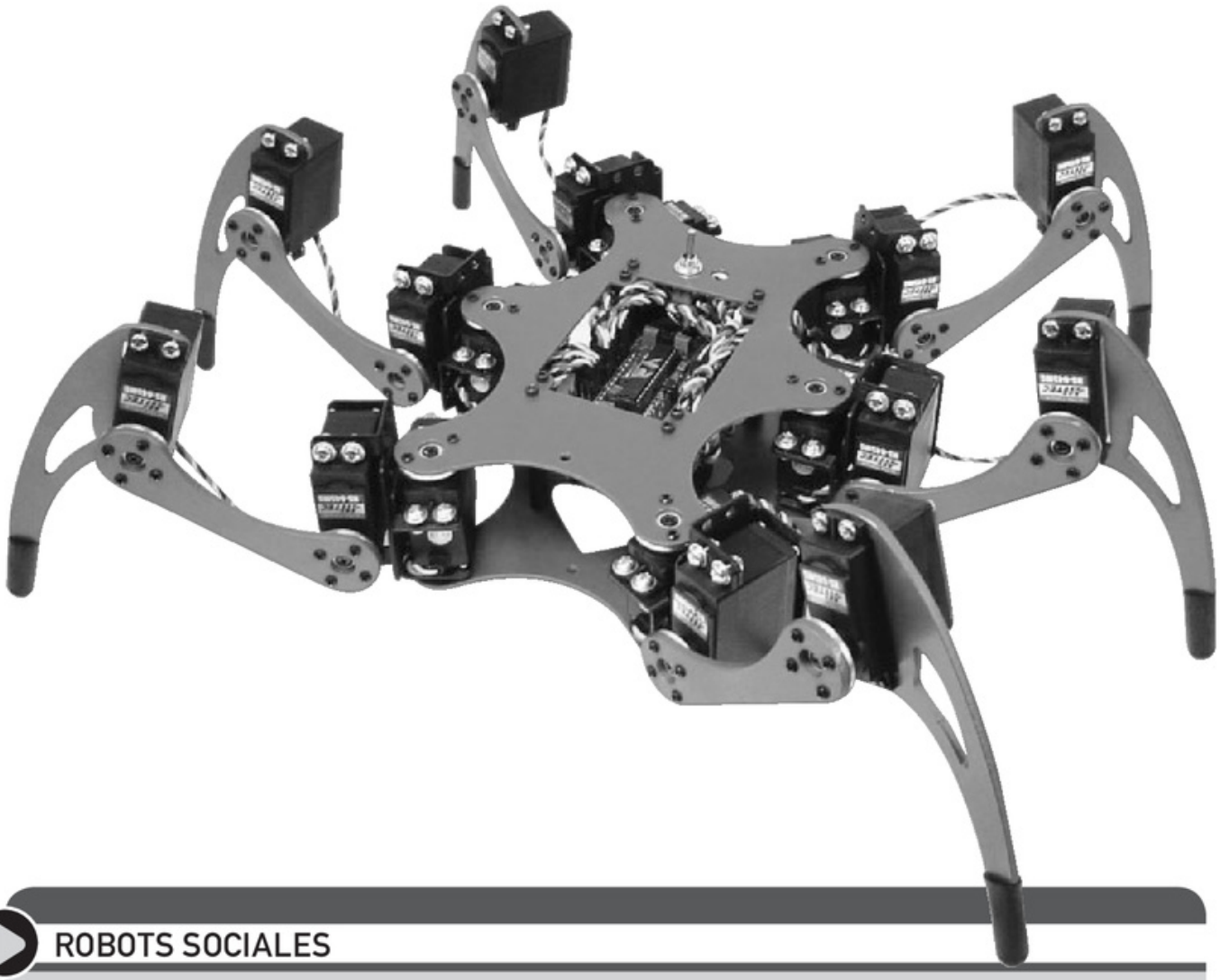
Los robots son muy utilizados en tareas en las que se requiere alta exactitud

MÚLTIPLES DISCIPLINAS

Podemos decir que la robótica demanda a los diseñadores el conocimiento de diversas áreas para su desarrollo e implementación, como electrónica, sistemas computacionales, mecánica y análisis de la relación costo-beneficio, entre otras.

El **Instituto Norteamericano de Robótica** define al robot industrial como un manipulador multifuncional y reprogramable, diseñado para desplazar materiales, piezas, herramientas o dispositivos especiales, mediante movimientos programables y variables que permiten llevar a cabo diversas tareas.

Los robots son muy utilizados en tareas en las que se requiere alta exactitud, abaratamiento de costos, ensamblaje, embalaje, u otras que resultan peligrosas o repetitivas, incluso, sucias para las personas. Por tal motivo, existe la tendencia a buscar formas cada vez más humanoides para sus estructuras y, así, llegar a reemplazar al ser humano.



ROBOTS SOCIALES

Un robot social interactúa y se comunica con las personas, siguiendo patrones, comportamientos y normas sociales. Posee una apariencia humanoide, pues es necesario que tenga la habilidad de ubicarse dentro de la sociedad.

Otros impulsores importantes de los robots son las aplicaciones militares. Estas fomentan el desarrollo de robots cada día más complejos y demandan un avance tecnológico a veces solo superado por la ficción.

TIPOS Y CARACTERÍSTICAS DE LOS ROBOTS

Enumeraremos a continuación los distintos tipos de robots y las características más sobresalientes que se han alcanzado en los últimos años. En la actualidad, existe una amplia variedad de clasificaciones de robots, entre las que podemos encontrar: **aviones autónomos, segway, mayordomos, vehículos terrestres autónomos, humanoides, colaborativos, tipo Rover, bípedos, ápodos, polimórficos, con forma de animales** y otros. Según sus diferentes capacidades, los robots se clasifican en:

- **Androides bípedos**: son robots que parecen y actúan como los humanos. Tratan de imitar el comportamiento del hombre. Actualmente, su utilidad es solo para experimentación. En este modelo, la principal limitante es la implementación del equilibrio a la hora del desplazamiento.
- **Zoomórficos**: su sistema de locomoción imita al de los animales. La aplicación de estos robots es en el estudio de terrenos accidentados, volcanes y exploración de planetas, ya que son hábiles para sortear obstáculos. Pueden subclasificarse en caminadores y no caminadores; estos últimos son muy poco evolucionados.
- **Móviles**: poseen una gran capacidad de desplazamiento y, también, un alto nivel de inteligencia. El desplazamiento se logra mediante plataformas o carros con sistemas de locomoción rodantes. Tienen gran aplicación en líneas de montajes industriales, donde hay que asegurar el transporte de piezas de un punto a otro. Por lo general, son guiados por telemando o por la información recibida del entorno mediante sensores.



Los robots se clasifican según las capacidades que pueden desarrollar y su morfología



- **Poliarticulados**: se caracterizan por agrupar robots primordialmente sedentarios, que poseen un número reducido de articulaciones para mover sus elementos terminales. Su utilidad principal es en el área industrial, donde desplazan elementos que requieren cuidados especiales.

Aplicaciones de los robots

En el desarrollo de los robots existen **dos factores** esenciales que limitan la construcción. Por un lado, los económicos, ya que su fabricación involucra altos costos. Por el otro, los tecnológicos, que dependerán de la tecnología del momento.

Los campos de aplicación de los robots son diversos. En la **construcción** se los usa como ejecutores de tareas de transporte, montaje, soldado, dimensionamiento, exploración e investigación, pues presentan la ventaja de poder resistir a medios hostiles para el ser humano. En el **entretenimiento**, posibilitan la recreación de situaciones ficticias y efectos especiales. En la actualidad, el mayor campo de aplicación es la **automatización industrial**, pues existe gran interés para mejorar, agilizar y aumentar la producción de distintos procesos.

GENERACIONES DE ROBOTS

Los robot industriales se subclasifican según su grado de complejidad en diferentes generaciones.

Los de **primera generación** son aquellos que trabajan como repetidores de las extremidades superiores del hombre. Se consideran esclavos mecánicos, ya que se controlan por medio de la intervención directa. Podemos mencionar casos como manipulación de materiales radiactivos, exploración submarina y obtención de muestras, entre otras especialidades.

Los de **segunda generación** funcionan automáticamente, sin la intervención del hombre. Poseen posiciones predefinidas en donde se realizan tareas específicas. Este tipo de robot se caracteriza por ejecutar movimientos repetitivos.





La robótica demanda a los diseñadores el conocimiento de diversas áreas para su desarrollo

Los robots tienen la capacidad de una rápida reprogramación, lo cual les otorga una gran versatilidad. Su campo de aplicación se expande no solo a la manipulación de materiales, sino también a la colaboración en procesos de manufactura, como en la soldadura continua o de punto y en tareas de pintado. Los robots de **tercera generación** son dispositivos que se construyen con el objetivo de que sean capaces de elegir la mejor forma de hacer la tarea encomendada, teniendo en cuenta el entorno. Para lograr su misión, es necesario que posean alguna interacción con el medio y los objetos. Los dos requerimientos más comunes son la capacidad de reconocer objetos en el espacio y la posibilidad de adoptar distintas trayectorias para obtener el objetivo deseado.

INTELIGENCIA DE UN ROBOT

Como mencionamos anteriormente, la inteligencia de un robot se basa en la habilidad que posee para resolver problemas. Esto se consigue mediante la toma de decisiones de acciones predefinidas (giro, retroceso, etc.), a través del análisis de la información

EL COMIENZO DE LAS LEYES DE LA ROBÓTICA

En 1942, el famoso escritor **Isaac Asimov** enunció en sus novelas de ciencia ficción tres leyes aplicables a la robótica que con el tiempo se convirtieron en cuatro y representan el código moral que deberían seguir.

recogida del medio utilizando sensores y transductores. Podemos decir que, aunque no efectúe tareas complicadas, una heladera es un robot, porque mantiene la temperatura en su interior a un valor seleccionado; las decisiones de prender o apagar se evalúan utilizando la información recibida por los sensores.

Si la inteligencia es la aplicación de la lógica a la resolución de problemas, entonces, al surgir un problema, se generará un estímulo que deberá ser analizado para obtener alternativas de solución. Estas se someterán a un análisis y se seleccionará la más adecuada para su ejecución.

En la actualidad, existen sistemas de procesamiento cada vez más sofisticados que permiten que el robot tome la decisión, ejecute acciones, aprenda de sus errores y acumule experiencia.



Componentes de un robot

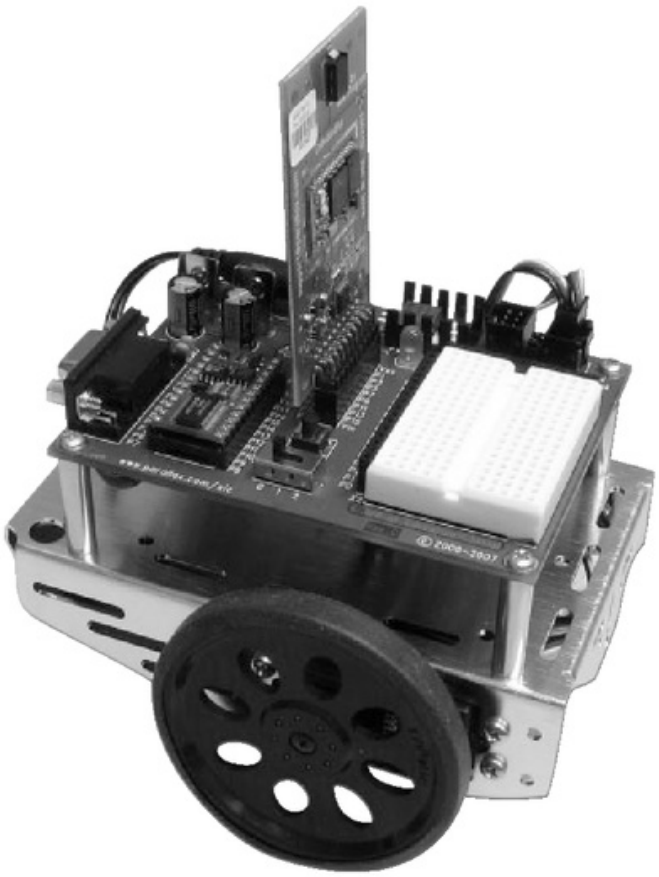
El **manipulador**, mecánicamente, es el componente principal que dota al robot de sus movimientos y define lo que llamamos la **zona de trabajo**.

LAS LEYES DE LA ROBÓTICA EN NUESTROS DÍAS

En la actualidad, el proyecto **Physical Human-Robot Interaction Dependability and Safety**, financiado por la **Unión Europea**, pretende forzar a los robots a cumplir las leyes de la robótica enunciadas por **Isaac Asimov** en **1942**.

Está formado por una serie de elementos estructurales sólidos, unidos mediante articulaciones que permiten un movimiento relativo entre dos piezas consecutivas. Cada articulación da al robot un grado de libertad.

La mayoría de los robots guarda similitud con la anatomía de las extremidades superiores del hombre.



Para hacer referencia a los elementos que componen al robot, muchas veces se usan términos como cintura, hombro, brazo, codo, muñeca y antebrazo. Las articulaciones utilizadas en un manipulador pueden ser:

- **Lineales**: los elementos estructurales se apoyan sobre un deslizador lineal y se desplazan a lo largo del recorrido mediante tornillos sinfín o cilindros.
- **Rotacionales**: conformadas por una unión giratoria que, a menudo, es manejada por motores eléctricos con transmisiones.
- **Por cilindros hidráulicos y uso de palancas**.

EL CONTROLADOR

Es el que maneja cada uno de los movimientos del manipulador, las acciones, los cálculos de posición y procesamiento de la información. Está continuamente enviando y recibiendo información de otros componentes que interactúan con él o que trabajan conformando un equipo. En función de los parámetros que se regulan, existen varios tipos de controladores:

- **De posición:** se encargan de controlar la posición del elemento terminal.
- **Cinemáticos:** tienen la capacidad de actuar sobre la posición y la velocidad.
- **Dinámicos:** además de actuar sobre la regulación de la velocidad y la posición, controlan las

DISPOSITIVOS DE ENTRADA/SALIDA

Los **periféricos E/S** son un conjunto de dispositivos pertenecientes a un sistema de procesamiento, que se utilizan para comunicarse con otros sistemas. Una entrada es una señal recibida por la unidad, mientras que las salidas son las enviadas por esta.

propiedades dinámicas del manipulador y de los elementos asociados.

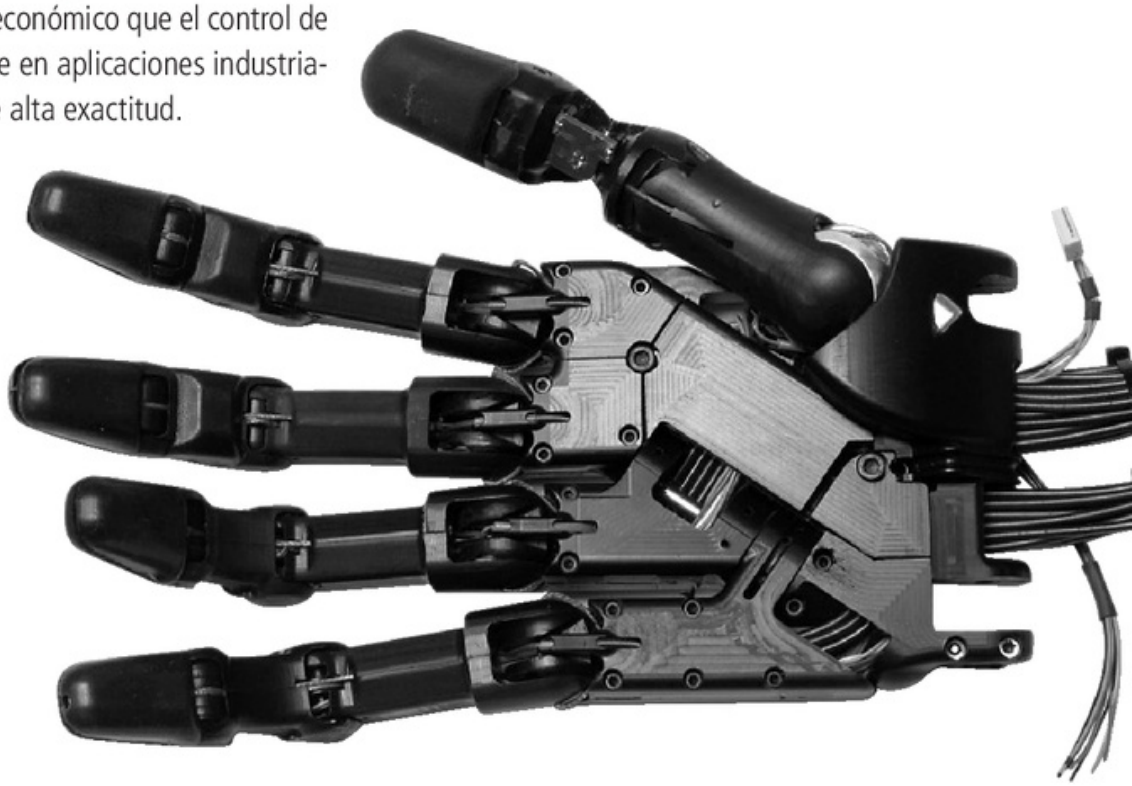
- **Adaptativos:** engloban las regulaciones anteriores y se ocupan de controlar la variación de las características del manipulador según la posición.

OTRAS CLASIFICACIONES

Otra clasificación en cuanto al control abarca: **control de lazo abierto** y **control de lazo cerrado**. El primero no posee una muestra de lo que sucede en el entorno, lo que da lugar a muchos errores. Aunque es más simple y económico que el control de lazo cerrado, no se admite en aplicaciones industriales en las que se requiere alta exactitud.

El manipulador es el componente principal que provee al robot de sus movimientos

La mayoría de los robots que se utilizan hoy en día con fines industriales se controlan mediante un proceso de lazo cerrado, es decir, a través de un lazo de realimentación.

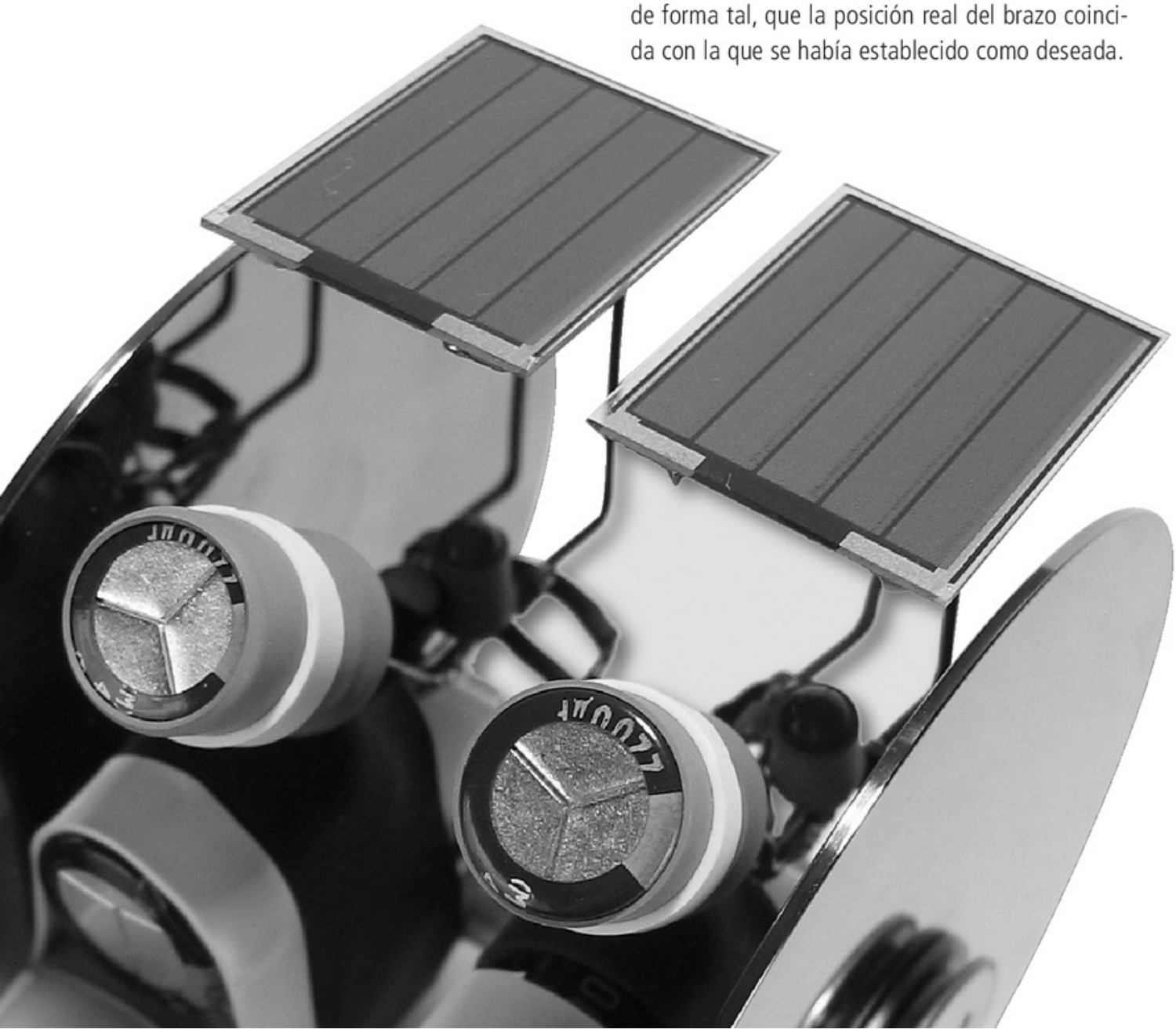


DISPOSITIVOS DE ENTRADA/SALIDA II

Los **dispositivos de E/S** pueden ser utilizados por otro sistema o por una persona para comunicarse con una computadora. Teclados y mouse son considerados dispositivos de entrada, mientras que monitores e impresoras son de salida.

Para realizar este tipo de control, se utiliza un **sensor** o **transductor**, que transforma una magnitud, por ejemplo, física en una eléctrica, que ingresa al controlador por medio de los dispositivos de adquisición de datos. Estos dispositivos están compuestos por circuitos acondicionadores de señal y tarjetas de entradas de información. También se utilizan tarjetas de drivers (salida de datos), a las cuales se conectan los actuadores.

Otra característica de este tipo de tarjetas es que permiten la comunicación con otras máquinas-herramientas, al tiempo que proveen de un aislamiento entre ellos, a través de **optoacopladores**. Una vez que la información de la posición, proporcionada por el sensor, ingresa en el programa que está en ejecución en el controlador, se compara con el valor inicial deseado. En función del error obtenido entre los dos valores, el controlador, siguiendo una secuencia de programación definida, actuará de forma tal, que la posición real del brazo coincida con la que se había establecido como deseada.



Servicios **al lector**



Encontraremos información adicional relacionada con el contenido que servirá para complementar lo aprendido.

RedUSERS.com

Índice **temático**

#

802.11B	161

A

AJAX	087
Androides bípedos	174
Asimov, Isaac	176/177
Atmel AVR	040

C

Cabecera HTTP	032
Caché de DNS	027
Callback	84/124
Capek, Karel	172
CGI	049
Chip select	140
Compilar stack	078/079
Concepto de Token	017
Control de lazo abierto	179
Control de lazo cerrado	179
Control de motor mediante PWM	168/169
Control remoto	119/120
Controladores cinemáticos	178
Controladores de posición	178
Controladores dinámicos	178
Costates	068
CustomHTTPApp.c	084

D

Da Vinci, Leonardo	172
Datagrama UDP	028
Devol, George	172
Diseño del PCB	147/148/149
Dispositivos de red	015/016/017

E

Especificaciones en HTML	038
ESTADO_LED_DOWN	084/085
ESTADO_LED_UP	084/085
Estándar 802.11	136/137
Ethernet con PIC	036/037/038



F

Formularios web	122
Freescale	040
Ftoa	085

G

Google	031

H

HardwareProfile.h	083
HOST_INTR	145
HTTP_IO_DONE	126
HTTPExecuteGet()	124/126
HTTPGetArg()	125

I

Instituto Norteamericano de Robótica	172
Interfaz FIFO	145
IP ESPECIAL	021
Itoa	085

L

LAN	012
Lenguaje HTML	032/033
lights_chk(0)	124
lights_chk(1)	124



M

MainDemo.c	083
MAN	012
Manipulador	177
Manipuladores lineales	178
Manipuladores por cilindros hidráulicos y uso de palancas	178
Manipuladores rotacionales	178
MCE Web Server CORE	119
Método GET	121/123
Microchip Configuration Wizard	072/073/150/151/152/153/154
Modelo OSI	018/019/020
Módulo ZG2100M	137/138/139/140
MPFSGetFilename()	126

P

Periféricos E/S	178/179
Physical Human-Robot Interaction Dependability and Safety	177

Prótocolo TCP	029/030/031
Prótocolo UDP	028
Puente serie-ethernet	095

R

RAMStringTo[]	095
RDATAn	145
Resolución por DNS	046
Robot social	173
Robots móviles	174
Robots poliarticulados	174
Robots zoomórficos	174
Rossum's Universal Robot	172

S

Servidor HTTP	047/048/049
Servidores DNS	026
Servidores web dedicados	114/115
Sistema de archivos MPFS	086/087/088/089/090
Sistema de nombre de dominio	025/026
sock_read()	045
sock_write()	045
SSI	049
Stack TCP/IP	040
status.xml	086
strcmppgm2ram()	126



T

TCP/IP en Rabbit	038
tcp_listen()	045
tcp_open()	045
TCPIPConfig.h	083
Telemetría de variables analógicas	108
Texas Instruments	040
Topología anillo	014
Topología árbol	012
Topología bus	013
Topología estrella	014
Topología malla	013

U

udp_open()	045
urHTTPExecuteGet()	123

V

Variables dinámicas	109/110/111/112/113

W

WAN	012
WDATAn	145
Web Server vía Telnet	032
Webpreview	126
WEP	158
WIZnet	040
WPA	158

CLAVES PARA COMPRAR
UN LIBRO DE COMPUTACIÓN

1 SOBRE EL AUTOR Y LA EDITORIAL

Revise que haya un cuadro "sobre el autor", en el que se informe sobre su experiencia en el tema. En cuanto a la editorial, es conveniente que sea especializada en computación.

2 PRESTE ATENCIÓN AL DISEÑO

Compruebe que el libro tenga guías visuales, explicaciones paso a paso, recuadros con información adicional y gran cantidad de pantallas. Su lectura será más ágil y atractiva que la de un libro de puro texto.

3 COMPARE PRECIOS

Suele haber grandes diferencias de precio entre libros del mismo tema; si no tiene el valor en tapa, pregunte y compare.

4 ¿TIENE VALORES AGREGADOS?

Desde un sitio exclusivo en la Red, un Servicio de Atención al Lector, la posibilidad de leer el sumario en la Web para evaluar con tranquilidad la compra, y hasta la presencia de adecuados índices temáticos, todo suma al valor de un buen libro.

5 VERIFIQUE EL IDIOMA

No solo el del texto; también revise que las pantallas incluidas en el libro estén en el mismo idioma del programa que usted utiliza.

usershop.redusers.com
VISITE NUESTRO SITIO WEB

» Vea información más detallada sobre cada libro de este catálogo.
» Obtenga un capítulo gratuito para evaluar la posible compra de un ejemplar.
» Conozca qué opinaron otros lectores.
» Compre los libros sin moverse de su casa y con importantes descuentos.
» Publique su comentario sobre el libro que leyó.
» Manténgase informado acerca de las últimas novedades y los próximos lanzamientos.

TAMBIÉN PUEDE CONSEGUIR NUESTROS LIBROS EN KIOSCOS O PUESTOS DE PERIÓDICOS, LIBRERÍAS, CADENAS COMERCIALES, SUPERMERCADOS Y CASAS DE COMPUTACIÓN.



LLEGAMOS A TODO EL MUNDO VÍA OCA * Y DHL **

* SOLO VÁLIDO EN LA REPÚBLICA ARGENTINA // ** VÁLIDO EN TODO EL MUNDO EXCEPTO ARGENTINA

usershop.redusers.com usershop@redusers.com + 54 (011) 4110-8700



Administrador de Redes Windows

En esta obra presentamos los fundamentos para la instalación, configuración y puesta en marcha de una red corporativa. El libro está centrado en Windows Small Business Server 2011, sistema operativo pensado para pequeñas y medianas empresas que necesitan llevar adelante soluciones de networking a bajo costo.

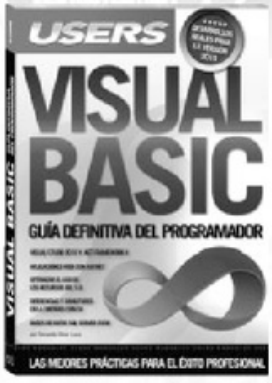
→ COLECCIÓN: MANUALES USERS
→ 320 páginas / 978-987-1773-80-0



HTML 5

En esta obra presentamos un nuevo paradigma de Internet que cambia de manera sustancial todo lo que conocíamos sobre el diseño y desarrollo web. A lo largo de sus capítulos, aprenderemos sobre los estándares web que existen, y conoceremos las diferencias entre las versiones anteriores y la actual del lenguaje.

→ COLECCIÓN: MANUALES USERS
→ 320 páginas / 978-987-1773-79-4



Visual Basic

Este libro está escrito para aquellos usuarios que quieran aprender a programar en VB.NET. Desde el IDE de programación hasta el desarrollo de aplicaciones del mundo real en la versión 2010 de Visual Studio, todo está contemplado para conocer en profundidad VB.NET al finalizar la lectura.

→ COLECCIÓN: MANUALES USERS
→ 352 páginas / 978-987-1773-57-2



Microcontroladores

Este manual es ideal para aquellos que quieran iniciarse en la programación de microcontroladores. A través de esta obra, podrán conocer los fundamentos de los sistemas digitales, aprender sobre los microcontroladores PIC 16F y 18F, hasta llegar a conectar los dispositivos de forma inalámbrica, entre muchos otros proyectos.

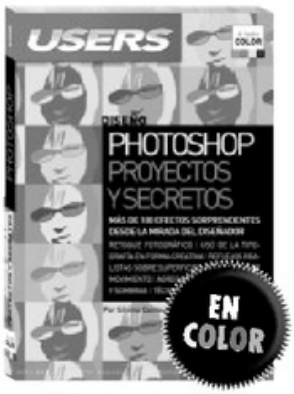
→ COLECCIÓN: MANUALES USERS
→ 320 páginas / 978-987-1773-56-5



Programador.NET

Este libro está dirigido a todos aquellos que quieran iniciarse en el desarrollo bajo lenguajes Microsoft. A través de los capítulos del manual, aprenderemos sobre POO y la programación con tecnologías .NET, su aplicación, cómo interactúan entre sí y de qué manera se desenvuelven con otras tecnologías existentes.

→ COLECCIÓN: MANUALES USERS
→ 352 páginas / 978-987-1773-26-8



Photoshop: proyectos y secretos

En esta obra aprenderemos a utilizar Photoshop, desde la original mirada de la autora. Con el foco puesto en la comunicación visual, a lo largo del libro adquiriremos conocimientos teóricos, al mismo tiempo que avanzaremos sobre la práctica, con todos los efectos y herramientas que ofrece el programa.

→ COLECCIÓN: MANUALES USERS
→ 320 páginas / 978-987-1773-25-1

¡Léalo antes Gratis!

En nuestro sitio, obtenga GRATIS un capítulo del libro de su elección antes de comprarlo.



WordPress

Este manual está dirigido a todos aquellos que quieran presentar sus contenidos o los de sus clientes a través de WordPress. En sus páginas el autor nos enseñará desde cómo llevar adelante la administración del blog hasta las posibilidades de interacción con las redes sociales.

→ COLECCIÓN: MANUALES USERS
→ 352 páginas / 978-987-1773-18-3



Administrador de servidores

Este libro es la puerta de acceso para ingresar en el apasionante mundo de los servidores. Aprenderemos desde los primeros pasos sobre la instalación, configuración, seguridad y virtualización; todo para cumplir el objetivo final de tener el control de los servidores en la palma de nuestras manos.

→ COLECCIÓN: MANUALES USERS
→ 352 páginas / ISBN 978-987-1773-19-0



Windows 7: Trucos y secretos

Este libro está dirigido a todos aquellos que quieran sacar el máximo provecho de Windows 7, las redes sociales y los dispositivos ultraportátiles del momento. A lo largo de sus páginas, el lector podrá adentrarse en estas tecnologías mediante trucos inéditos y consejos asombrosos.

→ COLECCIÓN: MANUALES USERS
→ 352 páginas / ISBN 978-987-1773-17-6



Desarrollo PHP + MySQL

Este libro presenta la fusión de dos de las herramientas más populares para el desarrollo de aplicaciones web de la actualidad: PHP y MySQL. En sus páginas, el autor nos enseñará las funciones del lenguaje, de modo de tener un acercamiento progresivo, y aplicar lo aprendido en nuestros propios desarrollos.

→ COLECCIÓN: MANUALES USERS
→ 432 páginas / ISBN 978-987-1773-16-9



Excel 2010

Este manual resulta ideal para quienes se inician en el uso de Excel, así como también para los usuarios que quieran conocer las nuevas herramientas que ofrece la versión 2010. La autora nos enseñará desde cómo ingresar y proteger datos hasta la forma de imprimir ahorrando papel y tiempo.

→ COLECCIÓN: MANUALES USERS
→ 352 páginas / ISBN 978-987-1773-15-2



Técnico Hardware

Esta obra es fundamental para ganar autonomía al momento de reparar la PC. Aprenderemos a diagnósticar y solucionar las fallas, así como a prevenirlas a través del mantenimiento adecuado, todo explicado en un lenguaje práctico y sencillo.

→ COLECCIÓN: MANUALES USERS
→ 320 páginas / ISBN 978-987-1773-14-5



PHP Avanzado

Este libro brinda todas las herramientas necesarias para acercar al trabajo diario del desarrollador los avances más importantes incorporados en PHP 6. En sus páginas, repasaremos todas las técnicas actuales para potenciar el desarrollo de sitios web.

→ COLECCIÓN: MANUALES USERS
→ **400** páginas / ISBN 978-987-1773-07-7



AutoCAD

Este manual nos presenta un recorrido exhaustivo por el programa más difundido en dibujo asistido por computadora a nivel mundial, en su versión 2010. En sus páginas, aprenderemos desde cómo trabajar con dibujos predeterminados hasta la realización de objetos 3D.

→ COLECCIÓN: MANUALES USERS
→ **384** páginas / ISBN 978-987-1773-06-0



Windows Seven Avanzado

Esta obra nos presenta un recorrido exhaustivo que nos permitirá acceder a un nuevo nivel de complejidad en el uso de Windows 7. Todas las herramientas son desarrolladas con el objetivo de acompañar al lector en el camino para ser un usuario experto.

→ COLECCIÓN: MANUALES USERS
→ **352** páginas / ISBN 978-987-1773-08-4



Photoshop

En este libro aprenderemos sobre las más novedosas técnicas de edición de imágenes en Photoshop. El autor nos presenta de manera clara y práctica todos los conceptos necesarios, desde la captura digital hasta las más avanzadas técnicas de retoque.

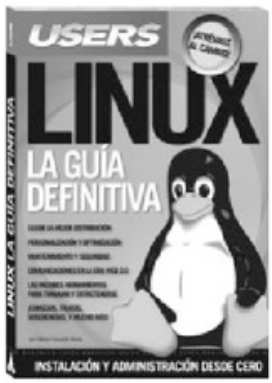
→ COLECCIÓN: MANUALES USERS
→ **320** páginas / ISBN 978-987-1773-05-3



Grabación y producción de música

En este libro repasaremos todos los aspectos del complejo mundo de la producción musical. Desde las cuestiones para tener en cuenta al momento de la composición, hasta la mezcla y el masterizado, así como la distribución final del producto.

→ COLECCIÓN: MANUALES USERS
→ **320** páginas / ISBN 978-987-1773-04-6



Linux

Este libro es una completa guía para migrar e iniciarse en el fascinante mundo del software libre. En su interior, el lector conocerá las características de Linux, desde su instalación hasta las opciones de entretenimiento, con todas las ventajas de seguridad que ofrece el sistema.

→ COLECCIÓN: MANUALES USERS
→ **320** páginas / ISBN 978-987-26013-8-6

¡Léalo antes Gratis!

En nuestro sitio, obtenga GRATIS un capítulo del libro de su elección antes de comprarlo.



Premiere + After Effects

Esta obra nos presenta un recorrido detallado por las aplicaciones audiovisuales de Adobe: Premiere Pro, After Effects y Soundbooth. Todas las técnicas de los profesionales, desde la captura de video hasta la creación de efectos, explicadas de forma teórica y práctica.

→ COLECCIÓN: MANUALES USERS
→ 320 páginas / ISBN 978-987-26013-9-3



Office 2010

En este libro aprenderemos a utilizar todas las aplicaciones de la suite, en su versión 2010. Además, su autora nos mostrará las novedades más importantes, desde los minigráficos de Excel hasta Office Web Apps, todo presentado en un libro único.

→ COLECCIÓN: MANUALES USERS
→ 352 páginas / ISBN 978-987-26013-6-2



Excel Paso a Paso

En esta obra encontraremos una increíble selección de proyectos pensada para aprender mediante la práctica la forma de agilizar todas las tareas diarias. Todas las actividades son desarrolladas en procedimientos paso a paso de una manera didáctica y fácil de comprender.

→ COLECCIÓN: PASO A PASO
→ 320 páginas / ISBN 978-987-26013-4-8



C#

Este libro es un completo curso de programación con C# actualizado a la versión 4.0. Ideal tanto para quienes desean migrar a este potente lenguaje, como para quienes quieran aprender a programar desde cero en Visual Studio 2010.

→ COLECCIÓN: MANUALES USERS
→ 400 páginas / ISBN 978-987-26013-5-5



200 Respuestas Seguridad

Esta obra es una guía básica que responde, en forma visual y práctica, a todas las preguntas que necesitamos contestar para conseguir un equipo seguro. Definiciones, consejos, claves y secretos, explicados de manera clara, sencilla y didáctica.

→ COLECCIÓN: MANUALES USERS
→ 320 páginas / ISBN 978-987-26013-1-7



Funciones en Excel

Este libro es una guía práctica de uso y aplicación de todas las funciones de la planilla de cálculo de Microsoft. Desde las funciones de siempre hasta las más complejas, todas presentadas a través de ejemplos prácticos y reales.

→ COLECCIÓN: 200 RESPUESTAS
→ 368 páginas / ISBN 978-987-26013-0-0

usershop.redusers.com



Proyectos con Windows 7

En esta obra aprenderemos cómo aprovechar al máximo todas las ventajas que ofrece la PC. Desde cómo participar en las redes sociales hasta las formas de montar una oficina virtual, todo presentado en 120 proyectos únicos.

→ COLECCIÓN: MANUALES USERS
→ **352** páginas / ISBN 978-987-663-036-8



PHP 6

Este libro es un completo curso de programación en PHP en su versión 6.0. Un lenguaje que se destaca tanto por su versatilidad como por el respaldo de una amplia comunidad de desarrolladores, que lo convierten en un punto de partida ideal para quienes comienzan a programar.

→ COLECCIÓN: MANUALES USERS
→ **368** páginas / ISBN 978-987-663-039-9



200 Respuestas: Blogs

Esta obra es una completa guía que responde a las preguntas más frecuentes de la gente sobre la forma de publicación más poderosa de la Web 2.0. Definiciones, consejos, claves y secretos, explicados de manera clara, sencilla y didáctica.

→ COLECCIÓN: 200 RESPUESTAS
→ **320** páginas / ISBN 978-987-663-037-5



Hardware paso a paso

En este libro encontraremos una increíble selección de actividades que abarcan todos los aspectos del hardware. Desde la actualización de la PC hasta el overclocking de sus componentes, todo en una presentación nunca antes vista, realizada íntegramente con procedimientos paso a paso.

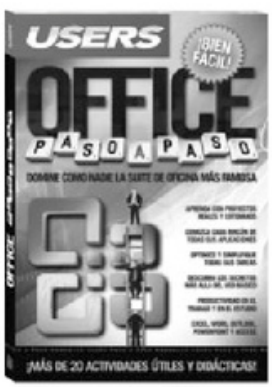
→ COLECCIÓN: PASO A PASO
→ **320** páginas / ISBN 978-987-663-034-4



200 Respuestas: Windows 7

Esta obra es una guía básica que responde, en forma visual y práctica, a todas las preguntas que necesitamos conocer para dominar la última versión del sistema operativo de Microsoft. Definiciones, consejos, claves y secretos, explicados de manera clara, sencilla y didáctica.

→ COLECCIÓN: 200 RESPUESTAS
→ **320** páginas / ISBN 978-987-663-035-1



Office paso a paso

Este libro presenta una increíble colección de proyectos basados en la suite de oficina más usada en el mundo. Todas las actividades son desarrolladas con procedimientos paso a paso de una manera didáctica y fácil de comprender.

→ COLECCIÓN: PASO A PASO
→ **320** páginas / ISBN 978-987-663-030-6

¡Léalo antes Gratis!

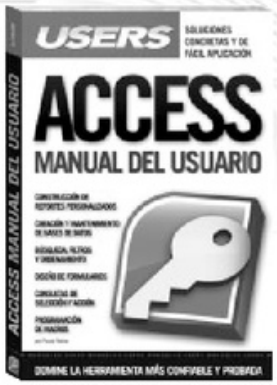
En nuestro sitio, obtenga GRATIS un capítulo del libro de su elección antes de comprarlo.



101 Secretos de Hardware

Esta obra es la mejor guía visual y práctica sobre hardware del momento. En su interior encontraremos los consejos de los expertos sobre las nuevas tecnologías, las soluciones a los problemas más frecuentes, cómo hacer overclocking, modding, y muchos más trucos y secretos.

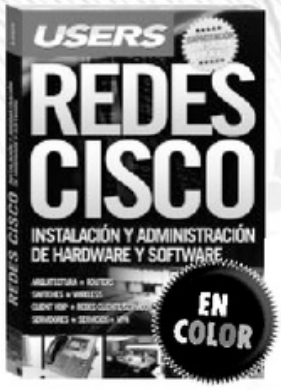
→ COLECCIÓN: MANUALES USERS
→ 352 páginas / ISBN 978-987-663-029-0



Access

Este manual nos introduce de lleno en el mundo de Access para aprender a crear y administrar bases de datos de forma profesional. Todos los secretos de una de las principales aplicaciones de Office, explicados de forma didáctica y sencilla.

→ COLECCIÓN: MANUALES USERS
→ 320 páginas / ISBN 978-987-663-025-2



Redes Cisco

Este libro permitirá al lector adquirir todos los conocimientos necesarios para planificar, instalar y administrar redes de computadoras. Todas las tecnologías y servicios Cisco, desarrollados de manera visual y práctica en una obra única.

→ COLECCIÓN: MANUALES USERS
→ 320 páginas / ISBN 978-987-663-024-5



Proyectos con Office

Esta obra nos enseña a usar las principales herramientas de Office a través de proyectos didácticos y útiles. En cada capítulo encontraremos la mejor manera de llevar adelante todas las actividades del hogar, la escuela y el trabajo.

→ COLECCIÓN: MANUALES USERS
→ 352 páginas / ISBN 978-987-663-023-8



Dreamweaver y Fireworks

Esta obra nos presenta las dos herramientas más poderosas para la creación de sitios web profesionales de la actualidad. A través de procedimientos paso a paso, nos muestra cómo armar un sitio real con Dreamweaver y Fireworks sin necesidad de conocimientos previos.

→ COLECCIÓN: MANUALES USERS
→ 320 páginas / ISBN 978-987-663-022-1



Excel revelado

Este manual contiene una selección de más de 150 consultas de usuarios de Excel y todas las respuestas de Claudio Sánchez, un reconocido experto en la famosa planilla de cálculo. Todos los problemas encuentran su solución en esta obra imperdible.

→ COLECCIÓN: MANUALES USERS
→ 336 páginas / ISBN 978-987-663-021-4



Robótica avanzada

Esta obra nos permitirá ingresar al fascinante mundo de la robótica. Desde el ensamblaje de las partes hasta su puesta en marcha, todo el proceso está expuesto de forma didáctica y sencilla para así crear nuestros propios robots avanzados.

→ COLECCIÓN: MANUALES USERS
→ **352** páginas / ISBN 978-987-663-020-7



Windows 7

En este libro encontraremos las claves y los secretos destinados a optimizar el uso de nuestra PC tanto en el trabajo como en el hogar. Aprenderemos a llevar adelante una instalación exitosa y a utilizar todas las nuevas herramientas que incluye esta versión.

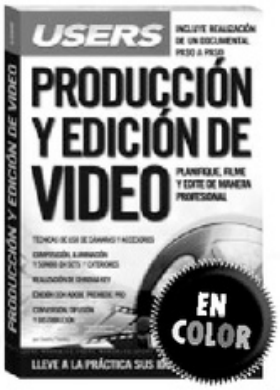
→ COLECCIÓN: MANUALES USERS
→ **320** páginas / ISBN 978-987-663-015-3



De Windows a Linux

Esta obra nos introduce en el apasionante mundo del software libre a través de una completa guía de migración, que parte desde el sistema operativo más conocido: Windows. Aprenderemos cómo realizar gratuitamente aquellas tareas que antes hacíamos con software pago.

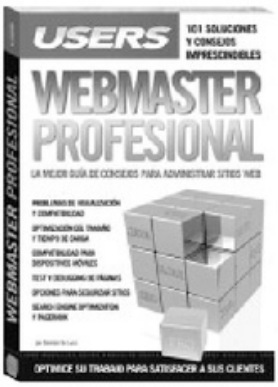
→ COLECCIÓN: MANUALES USERS
→ 336 páginas / ISBN 978-987-663-013-9



Producción y edición de video

Un libro ideal para quienes deseen realizar producciones audiovisuales con bajo presupuesto. Tanto estudiantes como profesionales encontrarán cómo adquirir las habilidades necesarias para obtener una salida laboral con una creciente demanda en el mercado.

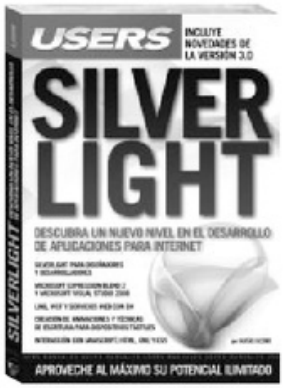
→ COLECCIÓN: MANUALES USERS
→ 336 páginas / ISBN 978-987-663-012-2



Webmaster Profesional

Esta obra explica cómo superar los problemas más frecuentes y complejos que enfrenta todo administrador de sitios web. Ideal para quienes necesiten conocer las tendencias actuales y las tecnologías en desarrollo que son materia obligada para dominar la Web 2.0.

→ COLECCIÓN: MANUALES USERS
→ 336 páginas / ISBN 978-987-663-011-5



Silverlight

Este manual nos introduce en un nuevo nivel en el desarrollo de aplicaciones interactivas a través de Silverlight, la opción multiplataforma de Microsoft. Quien consiga dominarlo creará aplicaciones visualmente impresionantes, acordes a los tiempos de la incipiente Web 3.0.

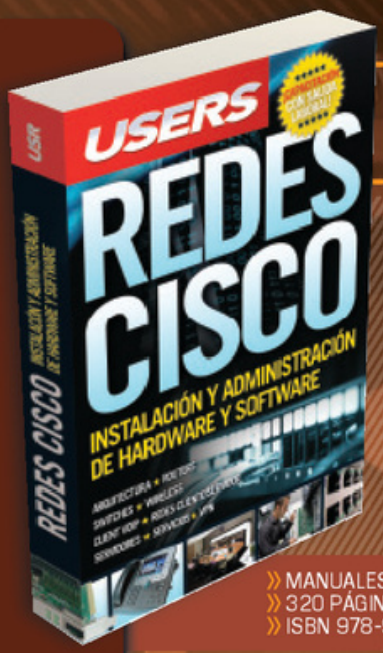
→ COLECCIÓN: MANUALES USERS
→ 352 páginas / ISBN 978-987-663-010-8

CONÉCTESE
CON LOS MEJORES LIBROS DE COMPUTACIÓN

LLEGAMOS A TODO EL MUNDO
VÍA OCA* Y DHL**

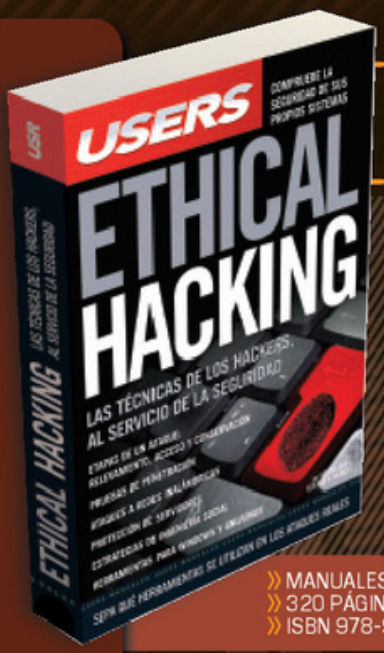
usershop.redusers.com
usershop@redusers.com

*SÓLO VÁLIDO EN LA REPÚBLICA ARGENTINA // **VÁLIDO EN TODO EL MUNDO EXCEPTO ARGENTINA



PLANIFICACIÓN, INSTALACIÓN Y ADMINISTRACIÓN DE REDES COMPLEJAS

» MANUALES USERS
» 320 PÁGINAS
» ISBN 978-987-663-024-5



CONOZCA LAS TÉCNICAS DE LOS HACKERS Y CONSIGA UN SISTEMA SEGURO

» MANUALES USERS
» 320 PÁGINAS
» ISBN 978-987-1347-93-3



ADMINISTRACIÓN PROFESIONAL Y AVANZADA PARA REDES COMPLEJAS

» MANUALES USERS
» 336 PÁGINAS
» ISBN 978-987-1347-55-1



INSTALACIÓN Y VIRTUALIZACIÓN DE SERVIDORES CORPORATIVOS

» MANUALES USERS
» 352 PÁGINAS
» ISBN 978-987-1773-19-0

NETWORKING CON MICROCONTROLADORES

En esta obra nos introduciremos en los conceptos de networking y protocolos de comunicación, y los pondremos en relación con los microcontroladores, para así poder acceder remotamente a los equipos: desde las primeras configuraciones hasta la programación de los dispositivos.

DENTRO DEL LIBRO ENCONTRARÁ

■ Networking con TCP/IP ■ Modelo OSI ■ Conexión a un servidor web ■ Ethernet con PIC ■ Rabbit ■ Rabbit Web ■ Multitarea cooperativo ■ Networking con PIC ■ Agregar servicios al stack ■ Envío de e-mails ■ Acceso desde Internet ■ Telemetría y control remoto Ethernet con PIC ■ Encendido de una lámpara de forma remota ■ Wi-Fi con PIC y ZeroG ■ Configuración del stack



ADEMÁS

ELECTRÓNICA PRÁCTICA
Aprenda a analizar, simular y construir circuitos

MICROCONTROLADORES
Funcionamiento, programación y aplicaciones prácticas

PROYECTOS CON MICROCONTROLADORES
Aprenda a desarrollar sus propias aplicaciones

SOBRE LA COLECCIÓN: ELECTRÓNICA

■ **Aprendizaje guiado mediante explicaciones claras y concisas** ■ **Proyectos prácticos basados en necesidades reales** ■ **Consejos de los profesionales** ■ **Infografías y procedimientos paso a paso** ■ **Producciones fotográficas profesionales**

MICROCONTROLLLERS NETWORKING

In this manual we will take a look at networking and **communications** protocols, to enhance and connect microcontrollers with complex networks. In it´s chapters we will find interesting projects to remotely control routines and hardware.

NIVEL DE USUARIO

BÁSICO	INTERMEDIO	AVANZADO	EXPERTO
		■	

ISBN 978-987-1773-24-4

9 789871 773244 >

RedUSERS.com

Nuestro sitio reúne a la mayor comunidad de tecnología en América Latina. Aquí podrá comunicarse con lectores, editores y autores, y acceder a noticias, foros y blogs constantemente actualizados. Además, podrá descargar material adicional de los libros y capítulos gratuitos, o conocer nuestras otras publicaciones y acceder a comprarlas desde cualquier parte del mundo.

Si desea más información sobre el libro: Servicio de atención al lector **usershop@redusers.com**