Manual de Usuario

HARD - TCP/IP STACK

MCE WebServer BOARD

Telemetría y Control remoto a través de Ethernet.

mc electronics®

Introducción

El MCE WebServer es una herramienta de desarrollo que se puede controlar a través de Internet. Permite monitorear parámetros a distancia (como la temperatura y humedad de una sala, la apertura de una puerta, la velocidad de un motor, etc).

Además cuenta con 10 salidas digitales para controlar dispositivos externos. Por ejemplo se puede encender un motor, activar una alarma, cerrar una compuerta, apagar una luz, etc.

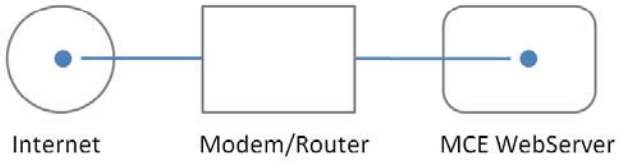
El usuario final accede a una dirección web del tipo (http://) donde puede controlar la placa desde una interface gráfica.

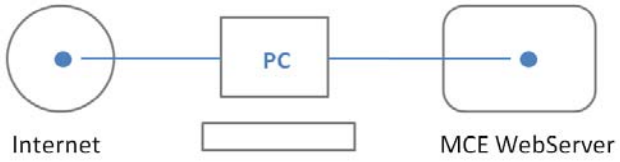
Información actualizada on-line:
http://www.mcelectronics.com.ar/desarrollos/web

mc electronics

Conexión a la red

Se conecta directamente a un modem o a un router con un cable cat. 5e. recto.
En este caso, se habilita DHCP (cliente) para obtener una dirección IP automáticamente.
También soporta conexión a PC mediante un cable cat. 5e. cruzado.





Conexión a PC: Conectar un cable UTP/STP cruzado al terminar RJ45.
Conexión a Router: Conectar un cable UTP/STP recto al terminar RJ45.

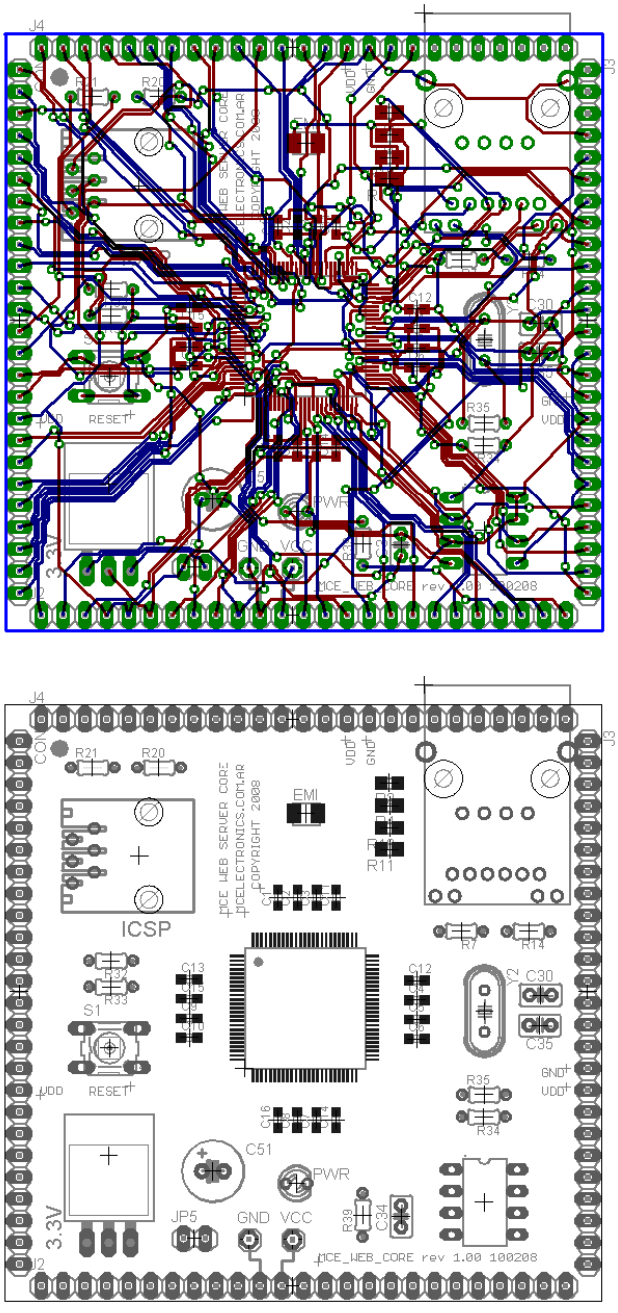
En el explorador Web poner http://mcewebserver o la IP: 169.254.1.1 en el caso de conexión directa a la PC. Si se utiliza un router con DHCP activo, la IP aparece en el display.

Información actualizada on-line:
http://www.mcelectronics.com.ar/desarrollos/web

mc electronics

Layout de componentes - CORE

WEBCORE - REV 100208

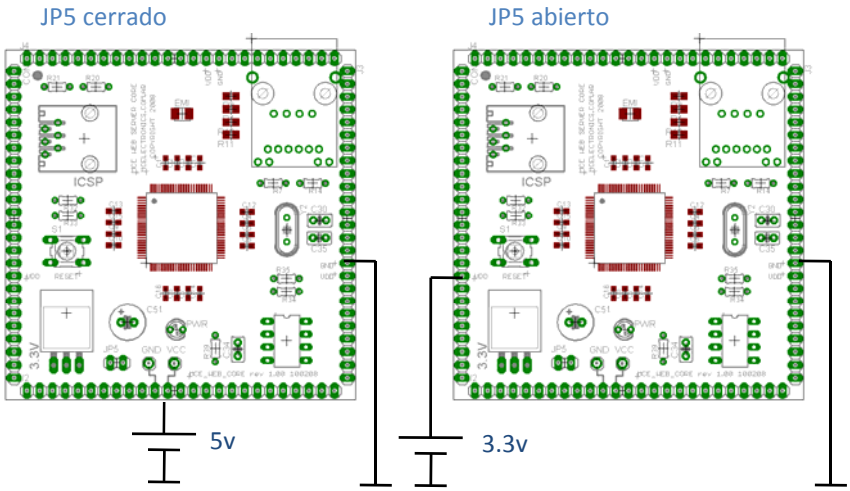


Dimensiones: 7.0 x 7.0 x 2.2 cm.
Peso: 36 g.

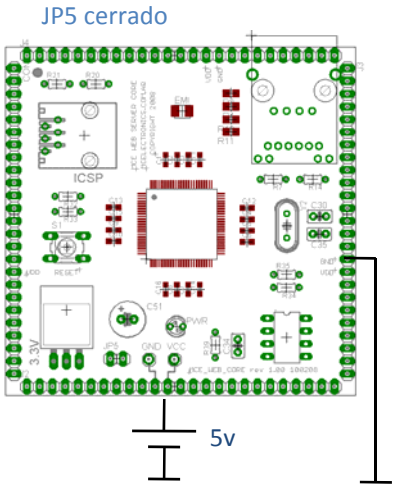
Layout de componentes - CORE

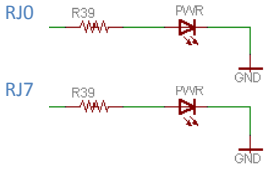
WEBCORE - REV 100208

Puede alimentarse con 5v directamente sobre los TestPoints o bien anular el regulador abriendo JP5 y alimentar la placa con 3.3v sobre VDD.



El stack trae cargado un ejemplo que muestra el estado de los 8 leds del puerto J. Para implementarlo, hay que agregar un led en cada pin de RJ0 aRJ7:

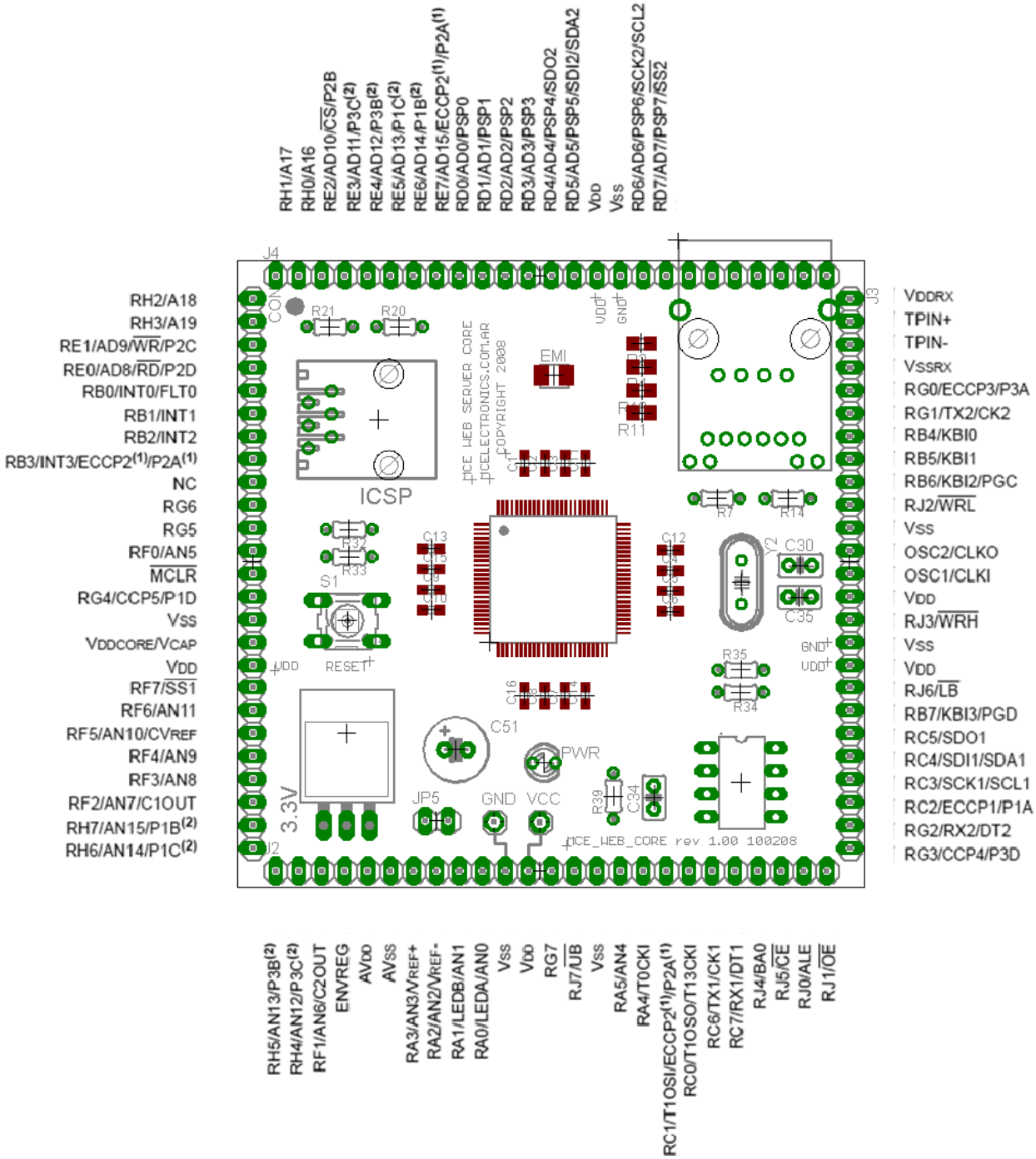




Como ejemplo de entradas preconfiguradas se pueden conectar 4 pulsadores a: RB0, RB1, RB2 Y RB3.

PIN-OUT (CORE)

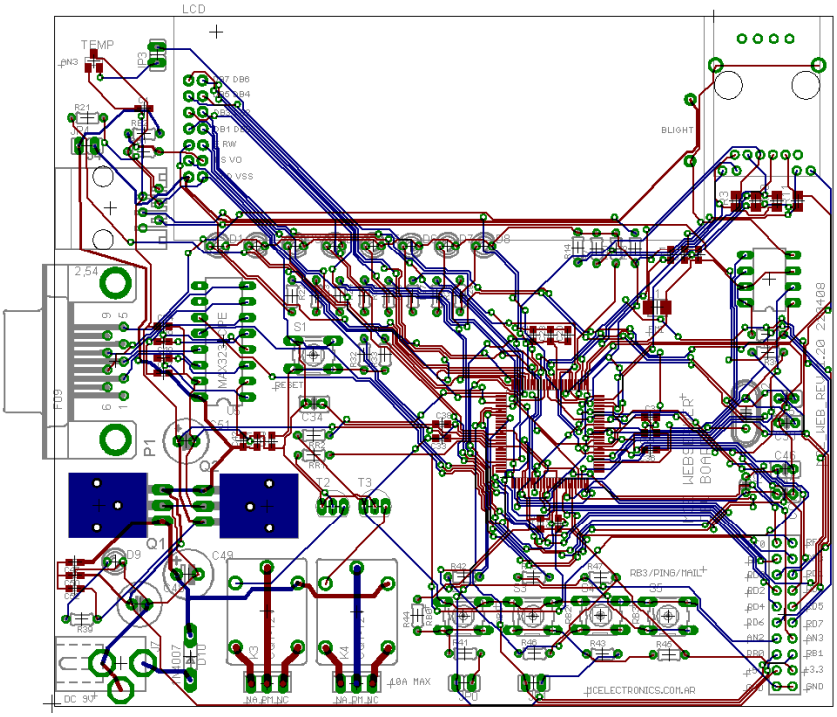
WEBCORE - REV 100208

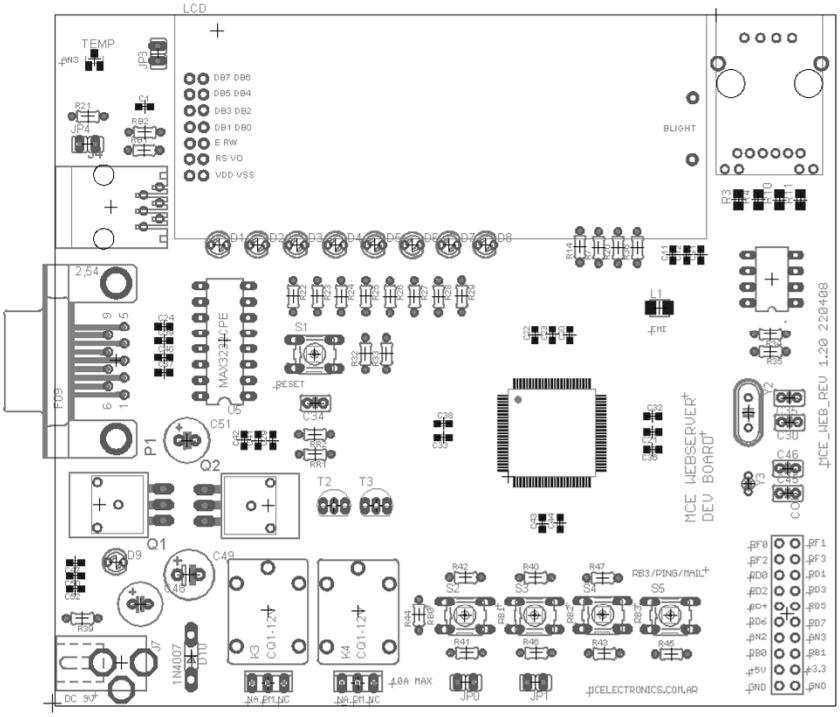


Convenientemente se adoptó la misma disposición de pines que en el PIC 18F97J60

Layout de componentes - DEV

WEBDEV - REV 220408





Layout de componentes - DEV

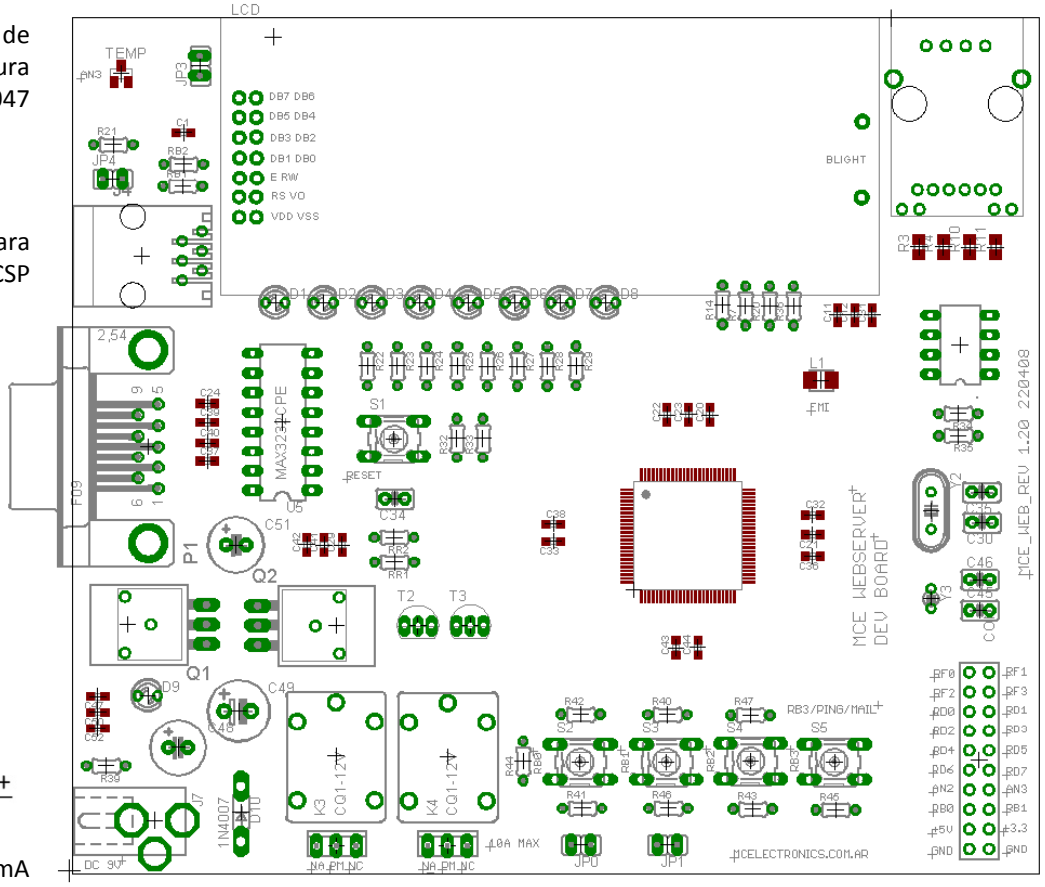
WEBDEV - REV 220408

Conector RJ45

Sensor de Temperatura TC1047

RJ11 para ICSP

RS232 (a la PC)



Memoria SPI de 1 Mbit (25LC1024)

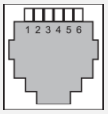
Header de Expansión

9V DC - 500mA

Salida por Relay hasta 10A

Pulsadores (simulan entradas dig.)

RJ11 para ICSP (pin-out)



1-VPP
2-VDD
3-GND
4-PGD
5-PGC
6-NC

Tabla de Jumpers

JP0	Habilita S2 (cerrado)
JP1	Habilita S3 (cerrado)
JP3	Habilita sensor temp. (cerrado)
JP4	Enciende Backlight (cerrado)

TCP/IP Stack - Introducción

Esta versión modificada está basada en la distribución original: Microchip TCP/IP stack v4.18. La idea era hacerla compatible con el MCE WebServer para que acepte memorias de 1 Mbit como la 25LC1024.

Hay que tener en cuenta que se puede compilar el stack para ser usado en micros que no tienen controlador ethernet como el 18F4620 (en este caso hay que agregarle un ENC28J60 de Microchip) y para aquellos que si traen un controlador integrado como el 18F97J60.

Que necesito ?

Para comenzar a trabajar con el stack necesitas instalar estos programas (incluidos en el DVD).

Microchip MPLAB 8.X
Microchip C18 Student Edition.
Microchip MPFS.
MCE TCP/IP Stack 4.18 (Incluye el WebSite)

Todos los ejemplos del sitio están compilados con el C18 para el PIC18F9J60 que utiliza el MCE WebServer.

Dentro del MCE TCP/IP Stack abrir la carpeta TCPIP Demo App que contiene el proyecto TCPIP Demo App-C18

Este es el proyecto general que contiene los encabezados y los archivos .c del stack, a continuación enumeramos los archivos principales.

TCPIPConfig.h

El archivo TCPIPConfig.h se utiliza para habilitar o deshabilitar funciones del stack como DNS, FTP, SNTP, SSL y UART dependiendo de las características de nuestra aplicación y la memoria disponible en el micro. Por ejemplo, para habilitar el cliente NTP, debemos incluir la sentencia: #define STACK_USE_SNTP_CLIENT

Dentro de TCPIPConfig.h es posible especificar si la pagina web va a estar en la memoria de programa o en la memoria EEPROM externa.

Si el website va a estar alojado en la memoria de programa debemos comentar la sentencia: #define MPFS_USE_EEPROM e incluir el archivo MPFSImg2.c en el proyecto. Ver Microchip MPFS.

También se puede definir la dirección MAC a través de MY_DEFAULT_MAC_BYTE5 y el nombre de la placa mediante MY_DEFAULT_HOST_NAME (ej: http://mcewebserver).

MainDemo.c

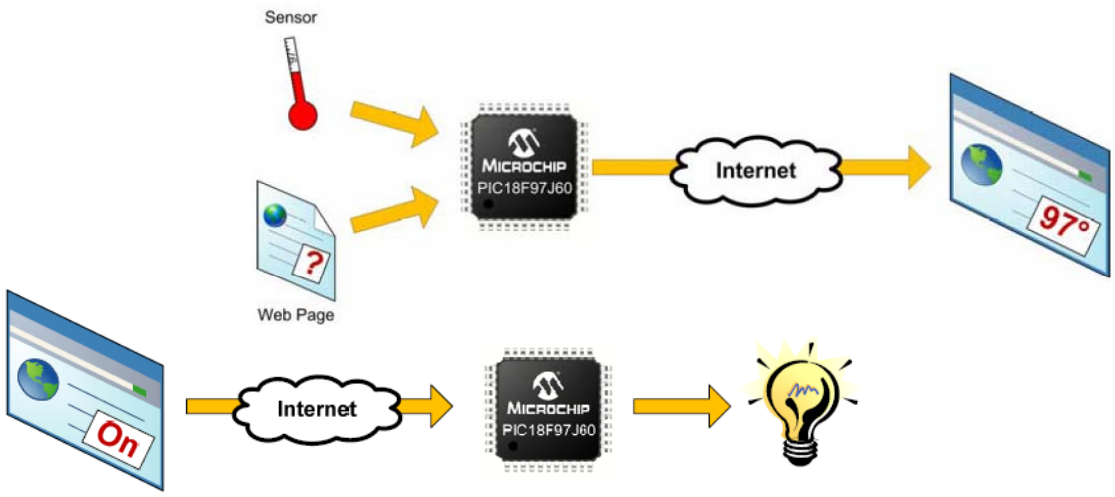
En el archivo MainDemo.c encontramos el mapeo de puertos, la configuración de los registros de la USART, los ADC y la inicialización de la placa. Por ejemplo ADCON0=0x0D; // Línea 1471- Para seleccionar AN3 como entrada analógica.

CustomHTTPApp.c

Por último, el tercer archivo que necesitamos para una aplicación básica es CustomHTTPApp.c, en el cual se encuentran las funciones callback, es decir aquellas que interactúan con la pagina web. Ver Funciones.

Funciones

En esta sección vamos a ver como interactuar con el micro desde la pagina web. Voy a desarrollar un ejemplo para monitorear el estado de un puerto y luego le vamos a agregar control remoto, es decir, cambiar el estado del puerto desde la interface web.



Monitoreo a distancia:
Para monitorear el estado de un puerto desde la web necesitamos hacer tres cosas:

Mapear e inicializar el puerto en el micro. (HardwareProfile.h, MainDemo.c)
Crear una función callback. (CustomHTTPApp.c)
Insertar una variable dinámica en la pagina web. (status.xml, index.html)

Luego compilamos el sitio web con MPFS2.exe, compilamos el proyecto con C18 desde el MPLAB y por último grabamos el firmware y la web en la memoria externa.

HardwareProfile.h

En este archivo simplemente asociamos un puerto con un nombre fácil de recordar, por ejemplo si colocamos un LED en RF3 (pin 22 del PIC18F97J60):

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

MainDemo.c

Vamos a definir RF3 como salida e inicializarlo en cero. Este proceso está fuera del loop principal, por lo que solo se va a ejecutar al reiniciar el micro:

```
static void InitializeBoard(void)
{

   LED_TRIS = 0;
   LED_IO = 0;
    //Continua la inicialización de los otros puertos.
}
```

CustomHTTPApp.c

La función callback contiene el nombre y el valor de la variable dinámica (led) que va a ser mostrada en la web.
Un error muy común es enviar directamente al buffer ethernet el estado del led mediante la variable int LED_IO.
Hay que tener en cuenta que SOLO SE PUEDEN ENVIAR STRINGS POR ETHERNET, por lo que en numerosas ocasiones usamos itoa o ftoa para convertir los tipos de dato.

```
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 lugar de usar los conversores de tipo, creamos dos variables auxiliares ESTADO_LED_UP y ESTADO_LED_DOWN para mostrar el estado del led mediante ON, OFF.

status, index.html

Hay que incluir en el sitio web la variable dinámica led, para esto vamos a modificar status.xml e index.html:

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>
```

status, index.html (cont.)

Entre <> se indica el nombre del divisor que se va a utilizar en el index para definir la posición de la variable.

Index.html finalmente es el archivo que ve el usuario final. Donde aparezca <“led”> el webserver lo va a reemplazar con ON, OFF según el valor de LED_IO.

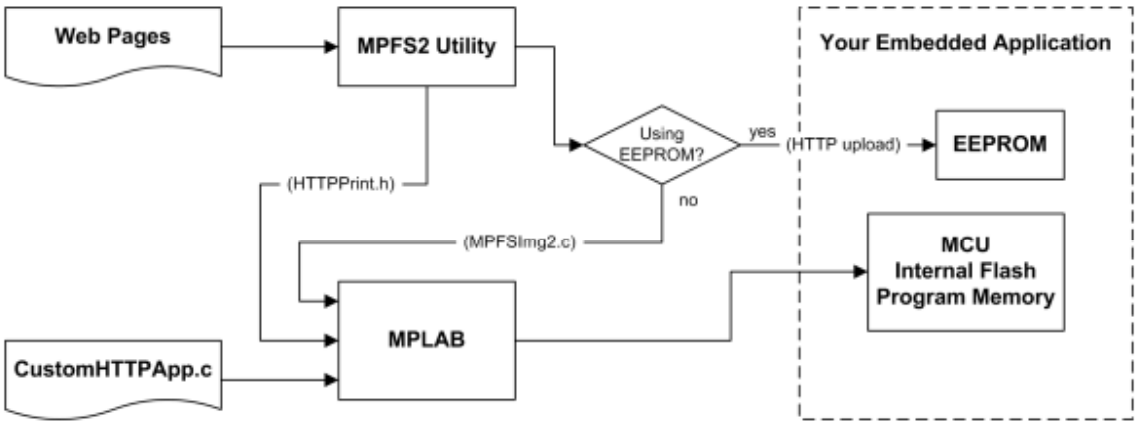
```
<span id=”led">?</span>  
```

Es importante detacar que este ejemplo utiliza AJAX para actualizar la información sin tener que apretar F5 en el explorador. Por lo tanto debemos agregar la siguiente sentencia al final del index:

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

MPFS

Microchip PIC File System es el sistema de archivos que utiliza el micro para leer la pagina web. Una vez que tenemos nuestro site terminado con los archivos html, jpg o gif debemos convertirlos al sistema MPFS para luego, poder cargarlos en el micro.



La aplicación MPFS.exe convierte el conjunto de archivos que componen la pagina web a dos formatos posibles:

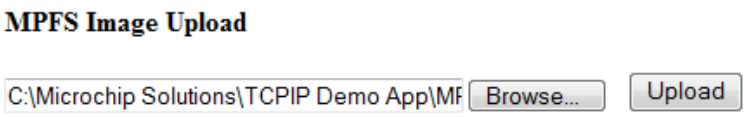
.bin (para cargar el website en la memoria EEPROM externa)
.c (para cargar el website en la memoria de programa)

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

MPFSUPLOAD

Hay dos formas de almacenar la pagina web:

1 - En la EEPROM, mediante mpfsupload. Para esto ejecutamos en el explorador de internet: http://mcewebserver/mpfsupload o bien http://169.254.1.1/mpfsupload y cargamos el archivo MPFSImg2.bin generado previamente con MPFS.exe



2 - En la memoria de programa, para esto necesitamos compilar todo el proyecto nuevamente (a menos que tengamos un bootloader).

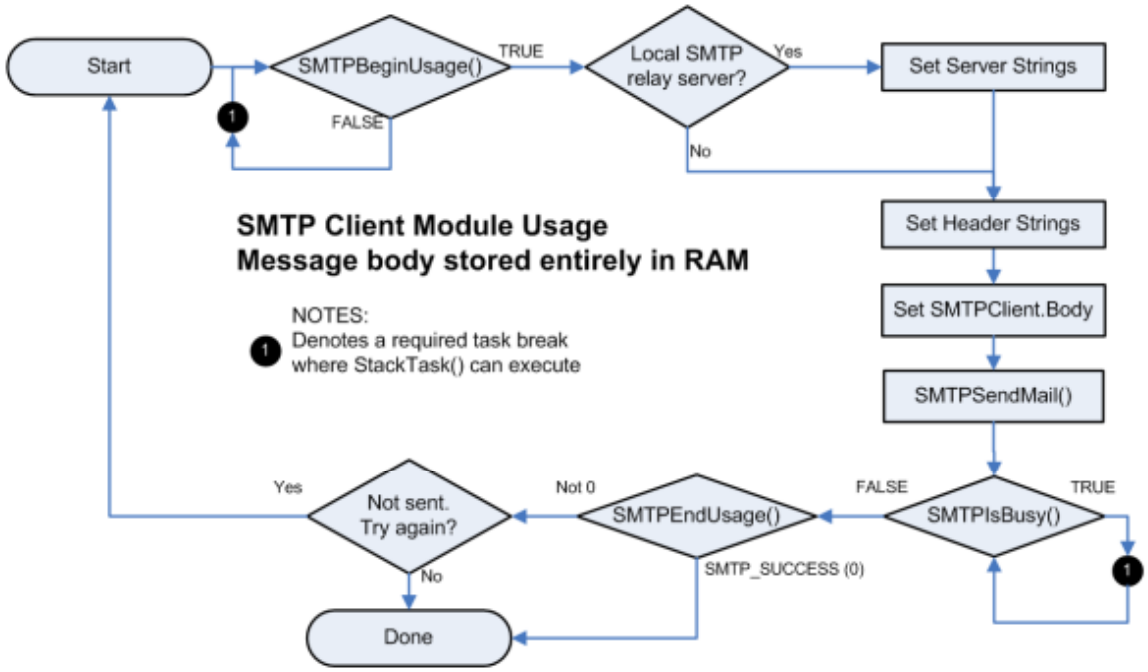
Debemos incluir en el proyecto el archivo MPFSImg2.c generado con MPFS.exe y comentar la sentencia #define MPFS_USE_EEPROM en el archivo TCPIPConfig.h.

De esta forma estaríamos presidiendo de la memoria EEPROM externa con el consecuente ahorro de dinero y espacio físico en la placa. Sin embargo sólo es posible con paginas web pequeñas ya que la memoria de programa del 18F97J60 queda prácticamente completa con todas las funciones del stack habilitadas.

SMTP

Para poder enviar emails desde el MCE WebServer debemos habilitar el cliente SMTP, dejando de comentar la sentencia STACK_USE_SMTP_CLIENT en el archivo *TCPIPConfig.h*.

El proceso se puede invocar en cualquier momento, por ejemplo si sube la temperatura, si se presiona un pulsador, etc. El mail del ejemplo envía el estado de todas las variables a la dirección que se especifique.



Antes de intentar enviar un mail con la función SMTPSendMail() debemos verificar si no hay otro mail en proceso, para eso utilizamos en primer término SMTPBeginUsage() como indica el diagrama.

Si nuestro servidor de correo saliente requiere autorización, utilizamos SMTPClient.Server, SMTPClient.Username y SMTPClient.Password para setear los parámetros.

SMTP (cont.)

Por ultimo incluimos los campos “De”, “Para”, “Asunto” y “Mensaje” antes de llamar a SMTPSendMail().

Ejemplo completo:

```
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@mcelectronics.com.ar>";
SMTPClient.ROMPointers.From = 1;
SMTPClient.Subject.szROM = (ROM BYTE*)"Mensaje de prueba";
SMTPClient.ROMPointers.Subject = 1;
SMTPClient.Body.szRAM = RAMStringBody;
SMTPSendMail();
MailState++;
}
```

Impreso en papel reciclado.
Buenos Aires - Argentina
Agosto 2009

MCE WebServer Board
Manual REV: 110809H

MCE110809H

Austria 1760 - OF 8
Ciudad de Buenos Aires (1425).
BA. Argentina.

(011) 6091-4922/4581
www.mcelectronics.com.ar
info@mcelectronics.com.ar