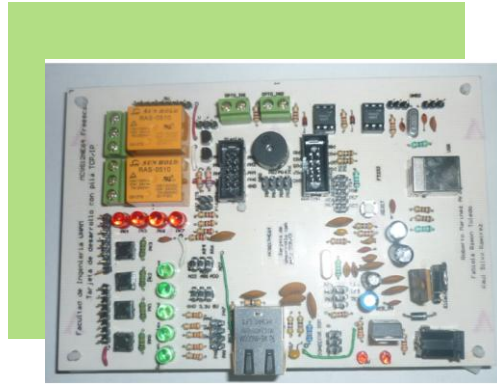


## Capítulo 6

# Pruebas de la tarjeta de desarrollo

---



- 6.1 Configuración del sistema de desarrollo**
  - 6.1.1 Conexión del puerto USB
  - 6.1.2 Conexión del programador
  - 6.1.3 Conexión a la red local
  - 6.1.4 Alimentación
- 6.2 Programación del microcontrolador mediante Ambiente de Desarrollo Integrado (IDE)**
  - 6.2.1 Programación mediante puerto BDM
  - 6.2.2 Programación mediante el *Monitor Serie*
- 6.3 Velocidad de operación del software**
  - 6.3.1 Frecuencia de la señal de reloj
  - 6.3.2 Temporizadores
- 6.4 Pruebas de dispositivos de entrada y salida de datos**
  - 6.4.1 Botones, leds y buzzer
  - 6.4.2 Teclado y display
- 6.5 Pruebas de dispositivos de monitoreo y control**
  - 6.5.1 Fotorresistencia y relevadores
  - 6.5.2 Entradas optoacopladas
- 6.6 Pruebas de comunicación**
  - 6.6.1 Comunicación serie
  - 6.6.2 Comunicación de protocolos de red

En los capítulos 3 y 5 se describió el hardware y software del sistema de control a distancia o tarjeta de desarrollo. En el presente capítulo se realizan una serie de pruebas que demuestran la funcionalidad y alcance del sistema de desarrollo.

En el capítulo 7 se desarrolla una aplicación demostrativa de un proceso de control a distancia, haciendo uso de las funcionalidades que se comprueban a continuación.

## 6.1 Configuración del sistema de desarrollo

Para utilizar el sistema o tarjeta de desarrollo y llevar a cabo las pruebas de funcionalidad es necesario el siguiente material:

- Tarjeta de desarrollo con microcontrolador MC9S12NE64 mostrada en el apéndice D.
- Computadora personal con:
  - S.O. Windows 2000 o superiores
  - Puerto USB
  - Tarjeta de red con Conector RJ45
  - Software *CodeWarrior Studio V5.9* o superior
  - Software controlador del convertidor FT232BM (Serie a USB).
- Cable de red cruzado UTP5 (para conectar directamente la tarjeta a la computadora) o dos cables de red uno a uno y una interfaz de red, como un hub o un switch.
- Módulo de Programación y Depuración mediante el puerto BDM (PYDBDM) para el MC9S12NE64. Se utilizó el *USB Multilink Interface* de la compañía *P&E Microcomputer Systems*.
- Cable USB con conector tipo B macho en un extremo y tipo A macho en el otro extremo.

Y realizar las conexiones siguientes:

- Conexión de la tarjeta de desarrollo a una red LAN, formada por una computadora y el sistema de control, o bien, entre un pequeño grupo de computadoras.
- Conexión del módulo de programación y depuración para el MC9S12NE64 (PYDBDM), conectado a la computadora a través del puerto USB y al microcontrolador a través del puerto BDM del sistema de desarrollo.
- Conexión del puerto USB de la computadora con el puerto USB del sistema de control.

### 6.1.1 Conexión del puerto USB

El microcontrolador cuenta con dos módulos de comunicación seriales asíncronos, de los cuales se utiliza uno para conectar un circuito de interfaz USB-RS232 (FT232BM) que permite comunicar el microcontrolador del sistema de desarrollo con una computadora a través de un cable USB (ver tema 3.3.1).

El puerto USB en el sistema de desarrollo tiene tres usos:

1. Cuando se conecta el cable USB al sistema se aprovecha la energía que la computadora provee a través del puerto USB para alimentar a la tarjeta de desarrollo.
2. El puerto USB se emplea para programar el microcontrolador directamente a través del programa *Monitor Serie* instalado previamente en el microcontrolador (apéndice B *Monitor Serie*), sin necesidad de conectar el módulo programador o depurador (PYDBDM)
3. La comunicación serie entre la computadora y el microcontrolador se realiza a través de esta interfaz por medio de los puertos USB de ambos dispositivos. Esta comunicación permite enviar mensajes de depuración del microcontrolador a la computadora, como los mensajes que se utilizan para verificar el estado de los paquetes en las funciones de la pila TCP/IP (recepción de paquetes, checksum correcto, paquete enviado, entre otros).

La conexión entre el puerto USB del microcontrolador y el puerto USB de la computadora se ilustran en la Figura 6.1

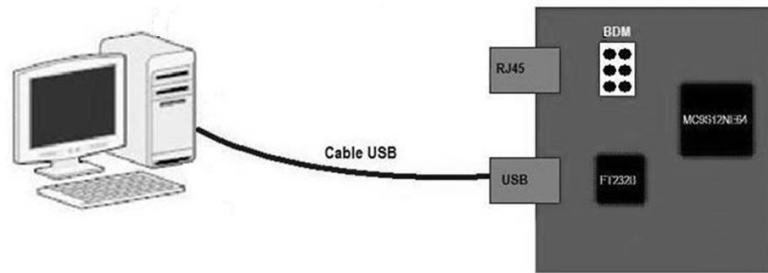


Figura 6.1 Conexión del cable USB

### 6.1.2 Conexión del programador

Para programar el microcontrolador se emplea un programador como el que se muestra en la Figura 6.2, el cual tiene un conector BDM en un extremo para conectarse al puerto BDM de la tarjeta de desarrollo (descrito en el capítulo 3) y en el otro extremo tiene un conector USB tipo B que le permite conectarse al puerto USB de una computadora mediante un cable USB con conectores tipo A y tipo B en sus extremos.



Figura 6.2 Programador para el microcontrolador

En la Figura 6.3 se muestra la conexión entre el programador, la computadora y el sistema de desarrollo.

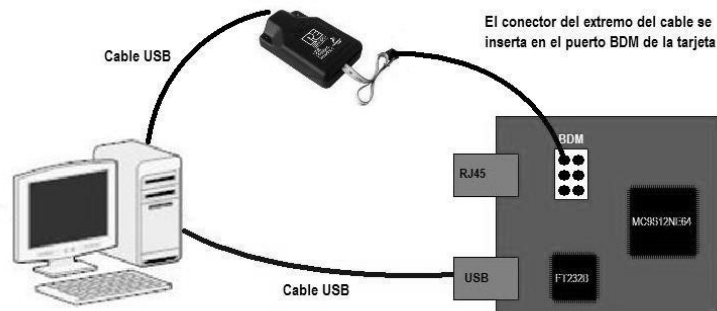
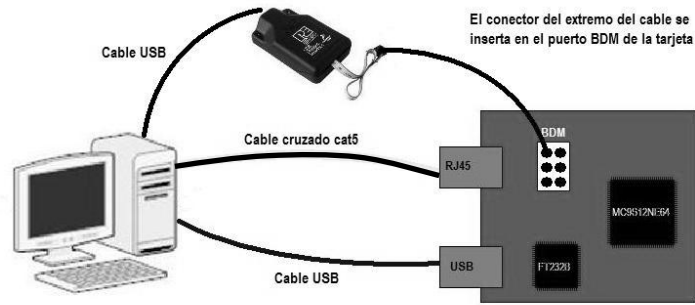


Figura 6.3 Conexión del programador entre la computadora y la tarjeta

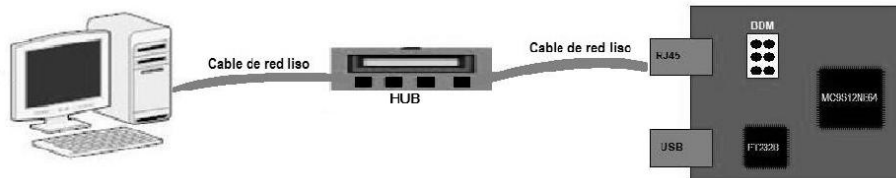
### 6.1.3 Conexión a la red local

Para formar una red local entre una computadora personal y la tarjeta con microcontrolador se requiere un cable de red cruzado categoría 5, con este cable la conexión es directa, ya que al estar cruzado, la señal Tx que envía un dispositivo entra en la terminal Rx del dispositivo con el que se comunica y viceversa. La Figura 6.4 ilustra la conexión.



**Figura 6.4** Conexión entre la tarjeta de desarrollo y una computadora a través de un cable de red cruzado

Si se emplea un dispositivo intermedio entre la computadora y la tarjeta de desarrollo, como un switch o un hub, la conexión se realiza con un cable uno a uno (no cruzado), de la computadora al hub o switch y de éste a la tarjeta con otro cable uno a uno. El dispositivo de red realiza la conversión entre las señales de transmisión y recepción de los dispositivos que comunica. En la Figura 6.5 se ilustra la conexión.



**Figura 6.5** Conexión entre la tarjeta de desarrollo y una computadora a través de un hub

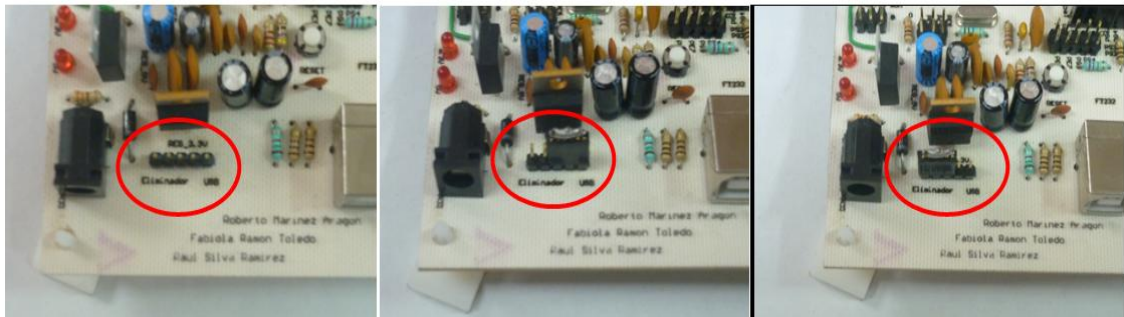
La conexión empleada para realizar las pruebas es la que se ilustra en la Figura 6.4. Con las conexiones mostradas la configuración del sistema queda lista para realizar las pruebas.

### 6.1.4 Alimentación

Para alimentar al sistema se tienen dos posibilidades:

1. Conectar un eliminador
2. Energizar la tarjeta directamente del puerto USB

Para seleccionar una de las dos posibilidades para la fuente de alimentación se coloca en la posición adecuada un selector que conecta tres terminales del conector de alimentación de cinco terminales, ya sea del lado del eliminador o del lado del conector USB. En la Figura 6.6 se muestra la selección de la fuente de alimentación.



Conector de alimentación

Selección del puerto USB como fuente de alimentación

Selección del eliminador como fuente de alimentación

**Figura 6.6** Selección de la fuente de alimentación

## 6.2 Programación del microcontrolador mediante Ambiente de Desarrollo Integrado (IDE)

Una vez realizadas las conexiones entre el sistema de desarrollo y la computadora trabajaremos los tres modos de programación del microcontrolador utilizando el Ambiente de Desarrollo Integrado (IDE) *CodeWarrior*. Estos modos de programación son el programador y depurador *P&E Multilink Cyclone Pro*, el *HC12 Serial Monitor*, con los que el microcontrolador se programa en el momento que se presiona el botón **DEBUG** del Ambiente de Desarrollo y, por último, la opción *Full Chip Simulation* que permite realizar una simulación del programa escrito sin tener el microcontrolador conectado a la computadora. Esta opción es útil para evaluar el comportamiento de los programas y detectar fallas o errores de tiempo de ejecución sin necesidad de tener el microcontrolador conectado.

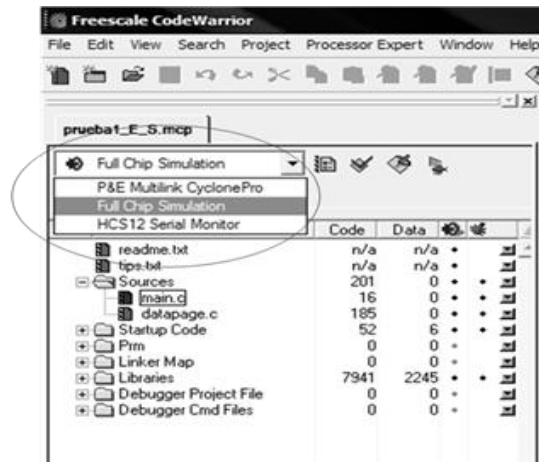


Figura 6.7 Selección de modo de programación y depuración de código

A continuación se describen dos pruebas que demuestran el uso de los métodos de programación del microcontrolador con PYDBDM y mediante el programa *Monitor Serie* usando un cable USB.

### 6.2.1 Programación mediante BDM

**Prueba 1:** Programación del microcontrolador por módulo BDM.

**Objetivo:** Demostrar que a través del módulo BDM de la tarjeta de desarrollo se puede programar la memoria del microcontrolador (revisar apéndice B *Monitor Serie*).

**Descripción:** Para llevar a cabo la prueba necesitamos conectar la tarjeta de desarrollo a la computadora a través del PYDBDM (Programador y Depurador DBM) y alimentarla con un eliminador de 9 V.

#### Desarrollo

1. En el ambiente CodeWarrior abrimos el proyecto “Serial Monitor” que previamente fue descargado de [www.freescale.com](http://www.freescale.com) y al cual se le han realizado las modificaciones que se especifican en el apéndice B.

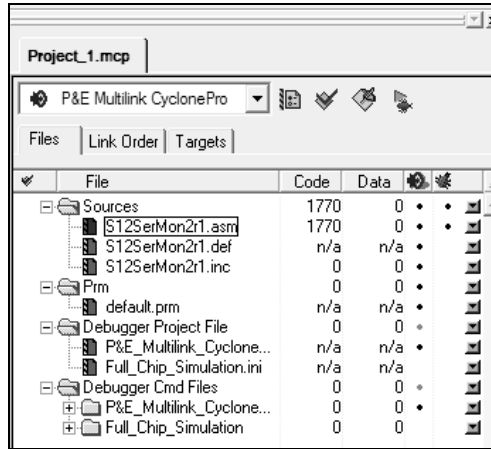


Figura 6.8 Contenido del proyecto Serial Monitor

2. Revisar que la opción seleccionada para la descarga del programa sea *P&E Multilink Cyclone Pro*, lo cual permite programar el microcontrolador a través del módulo BDM de la tarjeta de desarrollo. Al elegir la opción *P&E Multilink Cyclone Pro* el IDE CodeWarrior utiliza el software que controla el PYDDBM para realizar la comunicación con este dispositivo.

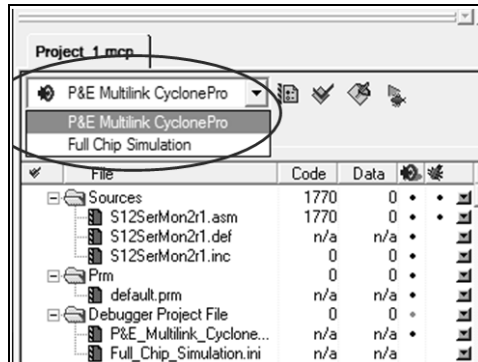


Figura 6.9 Selección de la opción P&E Multilink Cyclone Pro

3. Seleccionar la opción *Project* → *Make* para revisar el código y construir el archivo a descargar al microcontrolador, esperando no recibir una ventana con errores, en cuyo caso se deben atender dichos errores.

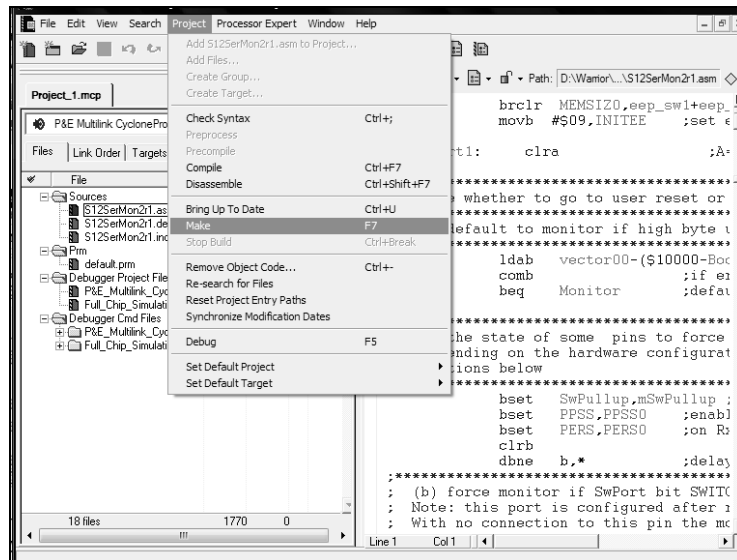


Figura 6.10 Selección de opción Make

4. Una vez que no se obtienen errores en el código al elegir la opción anterior, se selecciona la opción *Project* → *Debug*, con lo cual el programa se descarga al microcontrolador a través del PYDBDM conectado a uno de los puertos USB de la computadora. Obtenemos una ventana como la que se muestra en la Figura 6.11.

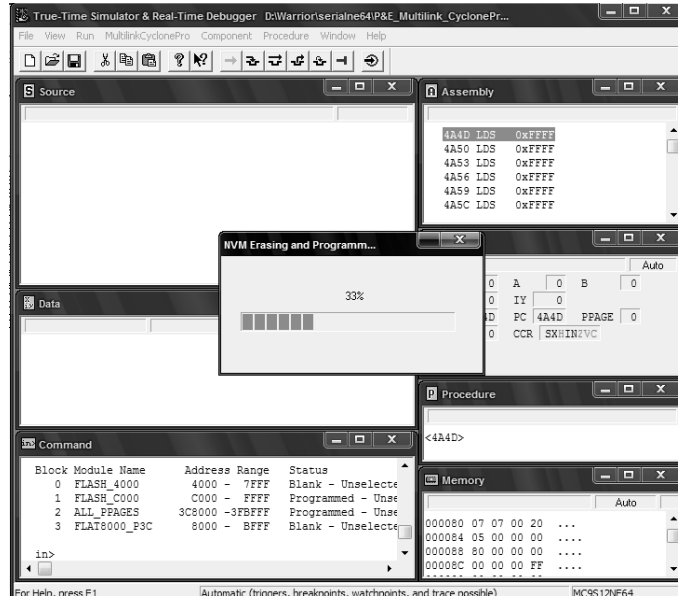


Figura 6.11 Proceso de descarga del programa al microcontrolador

Cuando se completa la descarga del programa al microcontrolador podemos observar una ventana como la que se muestra en la Figura 6.12.

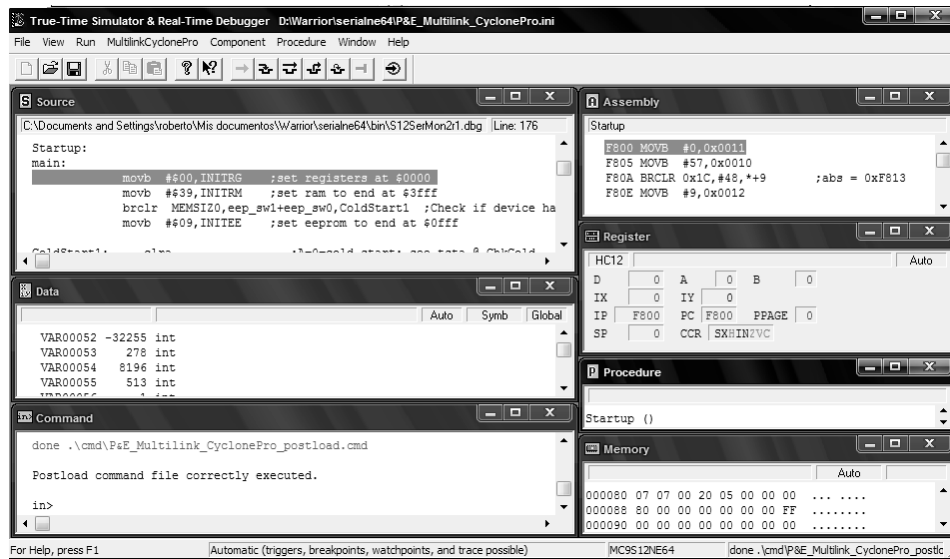


Figura 6.12 Ventana del Simulador y Depurador en Tiempo Real

5. Para comprobar que la memoria del microcontrolador fue programada correctamente revisamos desde *CodeWarrior* las direcciones de memoria del mismo, para tal efecto se tienen dos opciones:
  1. Maximizar la ventana que se titula *Memory* de la ventana de la Figura 6.12.
  2. Seleccionar en el menú superior *Component* → *Open* y elegir el icono denominado *Memory*.

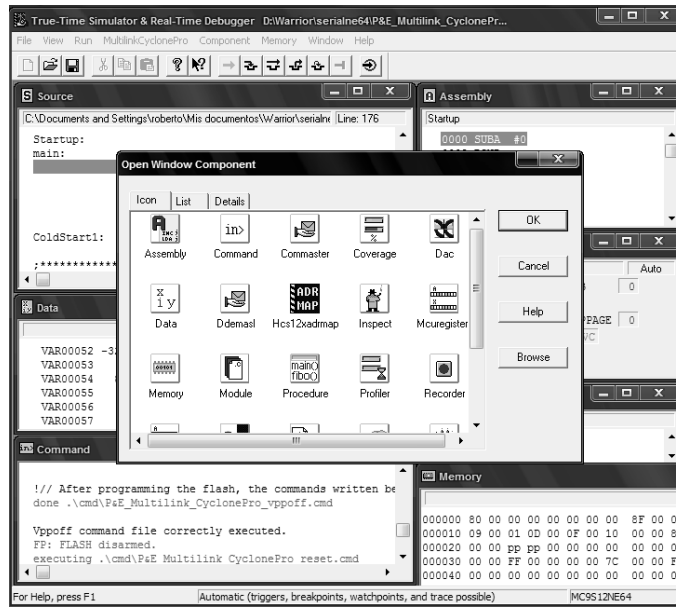


Figura 6.13 Ventana de apertura de componentes virtuales del microcontrolador

En la Figura 6.14 y Figura 6.15 se muestran las localidades de memoria del microcontrolador. Recordando que el *Monitor Serie* se almacena a partir de la dirección de memoria \$F800 (revisar apéndice B *Monitor Serie*), buscamos dicha dirección para comprobar que a partir de ahí se tiene almacenado el programa *Monitor Serie*.

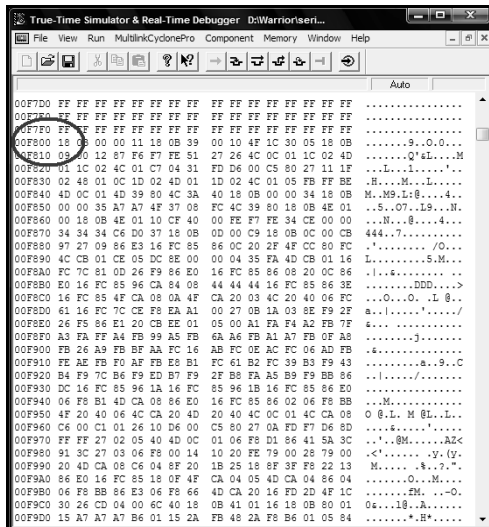


Figura 6.14 Inicio en memoria del programa *Monitor Serie*

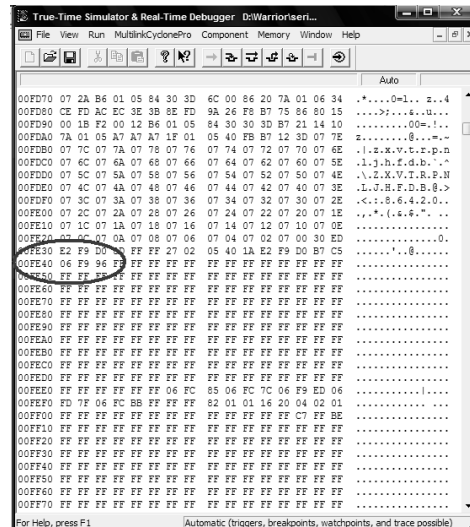


Figura 6.15 Fin en memoria del programa *Monitor Serie*

**Resultados:** Como se puede observar en la Figura 6.14, el programa *Monitor Serie* fue grabado satisfactoriamente en la memoria del microcontrolador a partir de la dirección \$F800 y ocupa un espacio aproximado de 2 Kb. En la Figura 6.15 se observa que la dirección de memoria donde termina el programa es \$FE42. Con estas imágenes demostramos que mediante el módulo BDM se puede programar la memoria del microcontrolador.

Al descargar el programa hemos preparado a la tarjeta para las pruebas siguientes, en las que el microcontrolador se programa mediante el *Monitor Serie* en lugar de la interfaz BDM, ya que no siempre se cuenta con este aditamento para llevar a cabo la programación del microcontrolador en la tarjeta de desarrollo.



## 6.2.2 Programación mediante el *Monitor Serie*

**Prueba 2:** Programación del microcontrolador usando el programa *Monitor Serie*.

**Objetivo:** Comprobar que es posible programar la memoria del microcontrolador sin el uso del módulo BDM, utilizando el programa *Monitor Serie* previamente cargado en el microcontrolador de la tarjeta de desarrollo.

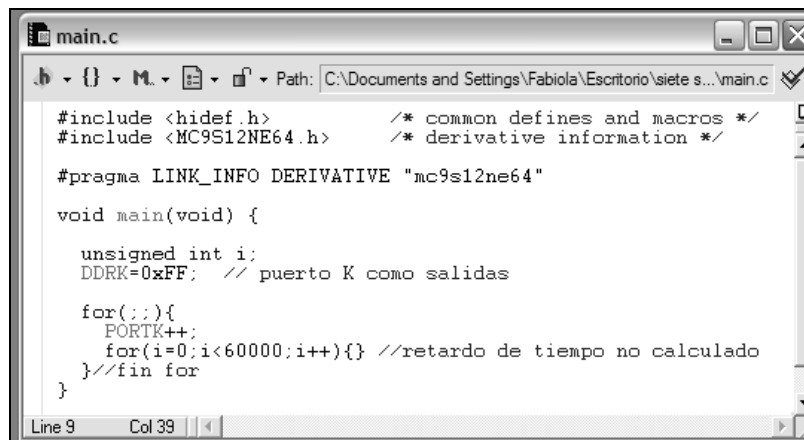
**Descripción:** En la prueba 1 descargamos al microcontrolador el programa *Monitor Serie*, el cual permite programar el microcontrolador sin utilizar el módulo BDM. Lo anterior representa una gran ventaja, ya que el usuario no necesitara adquirir un dispositivo PYDBDM, como el *USB Multilink Interface*, sólo se requiere un cable USB con conector tipo A y B para programar el microcontrolador.

Para realizar esta prueba debemos conectar el cable USB entre la computadora y la tarjeta de desarrollo y seleccionar el modo de alimentación, ya sea el puerto USB o el eliminador.

El código que se programa en el microcontrolador para esta prueba a través del *Monitor Serie* prende los leds alojados en el puerto K, representando el valor binario del puerto, el cual se incrementa recursivamente.

**Desarrollo:**

1. En el IDE *CodeWarrior* se crear un nuevo proyecto y se elige la opción *HC12 Serial Monitor* como medio de programación del microcontrolador.
2. Se escribe el programa de la Figura 6.16 en el archivo *main.c* del proyecto



```

main.c
Path: C:\Documents and Settings\Fabiola\Escritorio\siete s...main.c
#include <hieff.h> /* common defines and macros */
#include <MC9S12NE64.h> /* derivative information */

#pragma LINK_INFO DERIVATIVE "mc9s12ne64"

void main(void) {
    unsigned int i;
    DDRK=0xFF; // puerto K como salidas

    for(;;){
        PORTK++;
        for(i=0;i<60000;i++){ //retardo de tiempo no calculado
        } //fin for
    }
}
Line 9 Col 39

```

**Figura 6.16** Programa que prende leds del puerto K

3. El código escrito se comprueba seleccionando la opción *Project* → *Make*. Una vez que no se muestran errores se selecciona *Project* → *Debug*, con lo cual se inicia la descarga del programa al microcontrolador. Cuando la programación de la memoria flash del microcontrolador se completa, se presiona el botón *RESET* de la tarjeta de desarrollo y se comprueba que los leds encienden de forma secuencial, mostrando valores binarios incrementales.

**Resultados:** En la Figura 6.17 se observa el resultado de la ejecución del programa. Con esto comprobamos que es posible programar el microcontrolador con el uso de un cable USB a través del programa *Monitor Serie*. Se comprobó también que el módulo USB puede proveer la alimentación necesaria a la tarjeta de desarrollo.

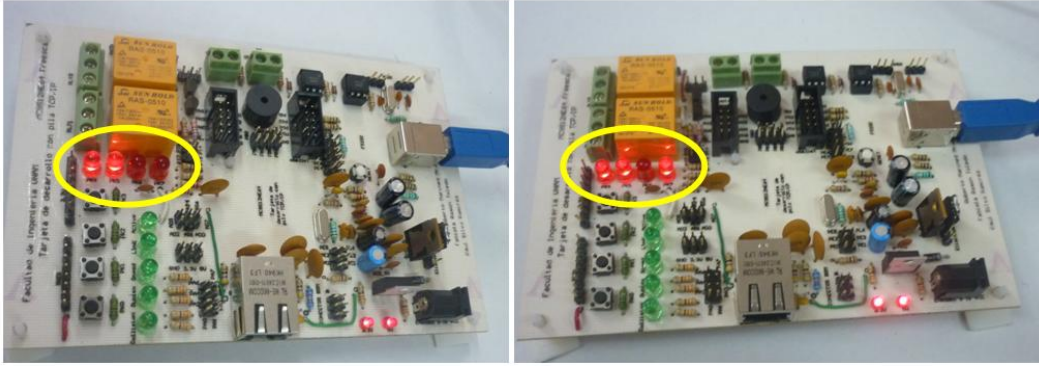


Figura 6.17 Resultado de la programación del microcontrolador a través del programa Monitor

## 6.3 Velocidad de operación del software

Las pruebas realizadas al sistema en esta sección demuestran la velocidad de operación del microcontrolador a través de la configuración del módulo PLL y el uso de variables tipo temporizador, las cuales se basan en la velocidad de reloj fijada por el PLL, por lo tanto, si la configuración de la velocidad del sistema a través del PLL funciona de manera adecuada, las variables empleadas para almacenar tiempos tendrán un desempeño satisfactorio.

El modo de verificar que el sistema puede ser configurado para que opere en diferentes velocidades consiste en realizar la configuración del módulo PLL, lo cual se realiza en la prueba 3, posteriormente, en la prueba 4, se demuestra el uso de variables tipo temporizador y con esto se comprueba que el sistema opera a la velocidad fijada por el módulo PLL.

### 6.3.1 Frecuencia de la señal de reloj

**Prueba 3:** Frecuencias de operación de la tarjeta de desarrollo.

**Objetivo:** Medir la frecuencia del cristal de entrada del microcontrolador y realizar una función que configure y habilite el módulo PLL para generar frecuencias de operación diferentes de la frecuencia base.

**Descripción:** El microcontrolador funciona con una fuente de reloj generada por un cristal externo de 25 MHz. Puede ser configurado para operar a la frecuencia determinada por el cristal externo o puede utilizar dicha frecuencia como base para generar una frecuencia de operación distinta. El módulo que permite la selección de la base de tiempo es el PLL (consultar capítulo 3 para detalles sobre el funcionamiento y la configuración del PLL). En esta prueba se implementa una función que permite configurar los registros del PLL para seleccionar una velocidad de operación diferente a la del oscilador del sistema.

Para llevar a cabo esta prueba, se conecta la tarjeta de desarrollo a la computadora a través del cable USB.

**Desarrollo:**

1. Medimos con el osciloscopio la terminal 48 del microcontrolador que corresponde a la entrada de reloj externa (EXTAL), en este caso del cristal de 25 MHz, y se obtiene la gráfica siguiente:

La señal que observamos en el osciloscopio tiene una frecuencia de 25 MHz y  $V_{pp}$  de 664.0 mV, esto corrobora la función del oscilador de convertir la energía de corriente continua en corriente alterna a una determinada frecuencia, además con esta prueba verificamos que la entrada al sistema generador de reloj y reset es de 25 MHz.

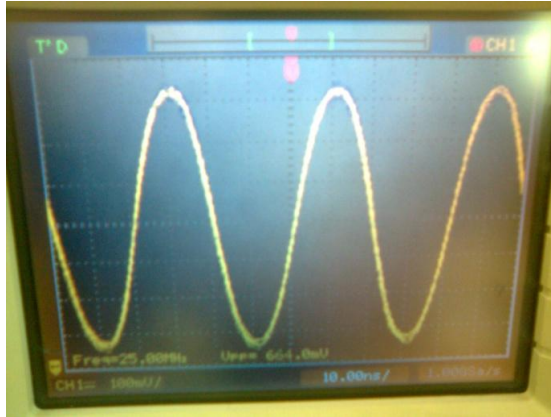


Figura 6.18 Señal de 25 MHz para generar alimentación de reloj para el microcontrolador

2. A continuación se crea un nuevo proyecto en el IDE *CodeWarrior* y se agregan dos archivos nuevos a la carpeta *Source* del proyecto, uno llamado “crg.h”, donde se define la función *setPLL* y el otro llamado “crg.c”, donde se implementa la función.



Figura 6.19 Contenido de la carpeta Source del proyecto

3. En el archivo “crg.h” se define la función *setPLL()* como:

```
void setPLL(unsigned char,mul,unsigned char);
```

4. Dentro del archivo “crg.c” implementamos la función que se muestra en la Figura 6.20, la cual configura la frecuencia del PLL según se describió en el capítulo 3.

```

// *** Selecciona y ajusta el PLL como fuente de reloj ***//
void setPLL(unsigned char mul, unsigned char div) {

    CLKSEL=0;
    CLKSEL_PLLSEL = 0; // Desactivar la selección de PLLCLK
    PLLCTL_PLLON = 0; // y apagar PLL para hacer configuraciones

    if(mul && div) // si los 2 valores son diferentes de cero
    {
        SYNRR = mul-1; // configurar valores de multiplicador y divisor
        REFDV = div-1;
        PLLCTL = PLLCTL_PLLON_MASK; // prender módulo PLL
        while(!CRGFLG_LOCK); // esperar hasta que sea seguro seleccionar PLLCLK
        CLKSEL = CLKSEL_PLLSEL_MASK; // seleccionar PLLCLK como fuente de reloj
    }
}

```

Figura 6.20 Función para seleccionar y ajustar el valor del PLL como señal de reloj del sistema

En la Figura 6.21 se muestra la configuración de la frecuencia del PLL a 50 MHz a partir de la frecuencia del oscilador de 25 MHz (es decir se ajusta la velocidad al doble), lo que genera una frecuencia de bus de 25 MHz. La siguiente fórmula expresa la frecuencia generada por el módulo PLL.

$$PLLCLK = 2 OSCCLK \frac{[SYNR + 1]}{[REFDV + 1]}$$

La función *setPLL()* recibe como parámetros los valores del numerador y denominador de la expresión anterior y almacena en los registros *SYNR* y *REFDV* los parámetros recibidos decrementados en una unidad, ya que al evaluar la expresión volverán a ser incrementados. Es decir, si se desea que la frecuencia del PLL sea el doble de la frecuencia del oscilador, el numerador y denominador de la expresión deben contener una cantidad igual (por ejemplo 1 y 1), esta cantidad se le pasa a la función cuando se invoca.

En el archivo *main.c* escribimos el programa que muestra la Figura 6.21 en la que la configuración del PLL, como se ha dicho, corresponde a una frecuencia de núcleo de 50 MHz y una frecuencia de bus de 25 MHz, sin perder de vista que de no activarse el módulo PLL la frecuencia de núcleo es de 25 MHz y la frecuencia de bus de 12.5 MHz, es decir, la mitad de la frecuencia de núcleo.

```
#include <hidef.h>          /* common defines and macros */
#include <MC9S12NE64.h>     /* derivative information */
#pragma LINK_INFO DERIVATIVE "mc9s12ne64"

#include "crg.h"

void main(void) {
    /* put your own code here */
    EnableInterrupts;
    SetPLL(1,1);
    for(;;) {} /* wait forever */
}
```

**Figura 6.21** Programa de configuración de la frecuencia PLL al doble de la frecuencia del oscilador

5. Para finalizar la prueba, se selecciona la opción *Project* → *Make*, con lo cual se revisa el código y se comprueba que el programa es correcto.

**Resultados:** Al término de esta prueba se ha verificado que la frecuencia de la señal de entrada al sistema generador de reloj y reset es de 25 MHz y se ha programado una función que permite activar el módulo PLL del microcontrolador para modificar la señal de reloj del sistema, provocando que se generen nuevas velocidades de operación.

## 6.3.2 Temporizadores

**Prueba 4:** Manejo de temporizadores

**Objetivo:** Utilizar la función implementada en la prueba 3 y las funciones explicadas en el tema 5.2 para demostrar el uso de temporizadores mediante la generación de una señal cuadrada.

**Descripción:** Con apoyo de las funciones descritas en el tema 5.2 sobre configuración y manejo de variables tipo temporizador y la función *setPLL()* (descrita en la prueba anterior), creamos una variable que se utiliza para contar intervalos de 100 ms, en los cuales se escriben valores de unos o ceros al puerto L. Estas variables se incrementan o decrementan en una interrupción que ocurre cada milisegundo aproximadamente, calculada en base a la frecuencia de bus de 25 MHz. Lo anterior genera una señal cuadrada en cada uno de los bits del puerto, de periodo igual a 200 ms, correspondiente a una frecuencia de 5 Hz. Este tipo de variables son muy utilizadas en el código de la pila TCP/IP, por tal motivo es importante comprobar su correcto funcionamiento.

**Desarrollo:**

1. En el IDE *CodeWarrior* abrimos el proyecto desarrollado para la prueba anterior y nos ubicamos en el archivo *main.c*.
2. Dentro del archivo *main.c* se escribe el código mostrado en la Figura 6.22

Como se muestra en el código, luego de las declaraciones de variables y configuraciones de puertos, se inicializa el PLL, el módulo de Interrupciones en Tiempo Real (RTI) y las variables de temporizador. Posteriormente se genera la señal cuadrada de frecuencia de 5 Hz a partir de la definición de 100 ms en alto y 100 ms en bajo, creando un periodo de 200ms. Para tales efectos se utilizan las funciones *create\_timer()*, *reset\_timer\_value()* y *read\_timer\_ms()*, descritas en el capítulo 5, tema 5.2.

```

main.c
Path: C:\Documents and Settings\roberto\Mis documentos\Warrior\leds\Sources\main.c

#include <hidef.h> /* common defines and macros */
#include <HC9S12NE64.h> /* derivative information */
#pragma LINK_INFO DERIVATIVE "mc9s12ne64"

#include "timers.h"
#include "crg.h"

void main(void) {
    unsigned char clock;

    /* put your own code here */
    EnableInterrupts;

    DDRL = 0xFF; // declarar el puerto L como salidas

    setPLL(1,1); // ajustar la frecuencia de bus a 25 MHz
    RTI_init(); // inicializar Interrupción de Tiempo Real (ajusta frecuencia a 1ms)
    timers_init(); // inicializar las variables temporizador

    clock = create_timer(DOWN,0,SEC); // crear un contador descendente inicializado en 0 segundos

    for(;;) {
        PTL = 0xFF; // escribir unos a todos los bits del puerto L
        reset_timer_value(clock,100,MS); // ajustar el valor del contador en 100ms
        while(read_timer_ms(clock)); // esperar ahí hasta que el contador valga cero
        PTL = 0; // escribir ceros a todos los bits del puerto L
        reset_timer_value(clock,100,MS); // ajustar el valor del contador en 100ms
        while(read_timer_ms(clock)); // esperar ahí hasta que el contador valga cero
    }
}
Line 34 Col 2

```

Figura 6.22 Ejemplo de manejo de temporizadores

- Una vez que el código está escrito, en el IDE *Code Warrior* seleccionamos las opciones *Project* → *Make* y posteriormente *Project* → *Debug* para descargar el programa al microcontrolador.
- Podemos entonces corroborar que los leds prenden y apagan y con ayuda del osciloscopio se lee la señal generada en una de las terminales del puerto L, obteniendo como resultado la señal mostrada en la Figura 6.23.

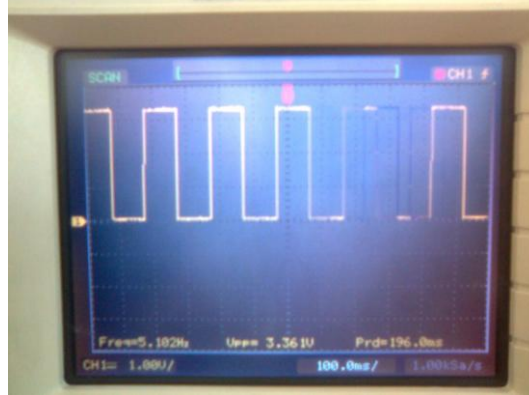


Figura 6.23 Señal generada con temporizadores

**Resultados:** Con esta prueba hemos demostrado que mediante el uso de variables tipo temporizador, del módulo de Interrupciones en Tiempo Real (RTI) y del módulo PLL, podemos generar distintos contadores de tiempo, mismos que pueden servir en la generación de nuevas señales, como es el caso de la señal cuadrada generada en el puerto L con una frecuencia de 5 Hz,  $V_{pp}$  de 3.3 V y periodo de 200ms. También podemos considerar correcto el funcionamiento de las funciones descritas en el tema 5.2.

## 6.4 Pruebas de dispositivos de Entrada/Salida de datos

En el capítulo 3 se describen los dispositivos con los que cuenta la tarjeta de desarrollo, unos corresponden a dispositivos de entrada y salida de datos y otros a dispositivos de monitoreo y control. Entre los primeros podemos mencionar a los botones, leds y *buzzer*. Conectados externamente a la tarjeta se tiene el teclado y el display. A continuación se realizan dos pruebas en las que se demuestra el funcionamiento de los dispositivos de entrada y salida.

## 6.4.1 Botones, leds y buzzer

**Prueba 5:** Manipulación de botones, leds y *buzzer* de la tarjeta de desarrollo.

**Objetivo:** Demostrar el funcionamiento de los dispositivos de entrada/salida y la interacción entre ellos.

**Descripción:** En esta prueba se desarrolla un programa que enciende un led al presionar un botón y suena el *buzzer* con un tono distinto para cada botón.

Los botones se alojan en la parte baja del puerto K, mientras que los leds están conectados en la parte alta del mismo puerto. El *buzzer* se encuentra conectado al bit 1 del puerto J.

Para realizar esta prueba se conecta la tarjeta de desarrollo mediante el cable USB a la computadora, se configura como fuente de alimentación el voltaje del puerto USB y se utiliza este mismo puerto para programar el microcontrolador (a través del *Monitor Serie*).

**Desarrollo:**

1. En el IDE *CodeWarrior* se crea un nuevo proyecto, seleccionando *HC12 Serial Monitor* como opción de programación del microcontrolador.
2. En el archivo *main.c* del proyecto se escribe el siguiente código:

```

main.c
Path: C:\Documents and Settings\Fabiola\Escritorio\...\main.c

#include <hidef.h> /* common defines and macros */
#include <MC9S12NE64.h> /* derivative information */
#pragma LINK_INFO DERIVATIVE "mc9s12ne64"

unsigned char enable_buzzer;

void buzzer(unsigned char i){
    unsigned int j,k;
    if(enable_buzzer){
        j=i*10;
        PTJ_PTJ1=0;
        for(k=0;k<j;k++){
            PTJ_PTJ1=1;
        }
    }
}

void main(void) {
    unsigned int i=0;
    DDRK=0xF0; //puerto K salidas/entradas
    DDRJ |= 0x02; //bit 1 del puerto J como salida
    enable_buzzer=1;

    for(;;) {
        PORTK=0;
        if(PORTK_BIT0==1){
            PORTK_BIT4=1;
            buzzer(20);
        }
        if(PORTK_BIT1==1){
            PORTK_BIT5=1;
            buzzer(100);
        }
        if(PORTK_BIT2==1){
            PORTK_BIT6=1;
            buzzer(150);
        }
        if(PORTK_BIT3==1){
            PORTK_BIT7=1;
            buzzer(230);
        }
    }
}
Line 20 Col 32

```

**Figura 6.24** Programa de interacción Botón, Led y Buzzer

3. Una vez que el código escrito no presente errores, se depura y programa el microcontrolador (*Project* → *Debug*).
4. Como resultado del programa se observa que al presionar el botón 1 (bit K0) el led 1 (bit K4) enciende y el *buzzer* emite un sonido, al presionar botón 2 (bit K1) el led 2 (bit K5) enciende y el *buzzer* emite un sonido distinto al anterior, así ocurre sucesivamente al presionar los botones 3 y 4. Cabe mencionar que

el sonido emitido por el *buzzer* sólo ocurre mientras se mantiene presionado el botón, una vez que se libera el botón el sonido desaparece.

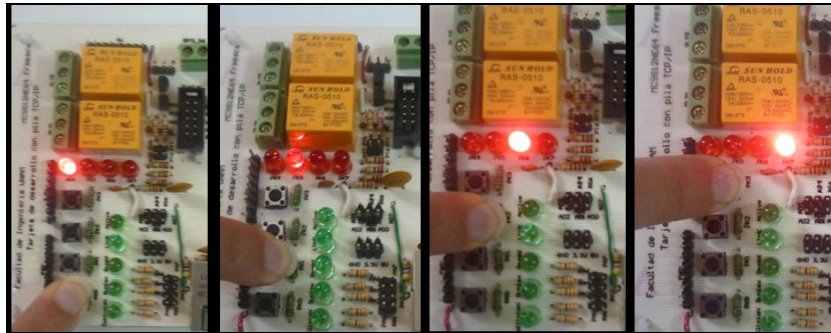


Figura 6.25 Prueba del programa Botones, Leds y Buzzer

**Resultados:** Con el programa desarrollado se comprueba la interacción que puede existir entre dispositivos de entrada y salida de la tarjeta de desarrollo, esto mediante la asignación de valores a los puertos configurados como salidas y la lectura de los puertos configurados como entradas. También se observa que al enviar un pulso al *buzzer* con un periodo distinto podemos provocar sonidos en diferentes tonos.

## 6.4.2 Teclado y display

**Prueba 6:** Uso del teclado matricial y display.

**Objetivo:** Demostrar el funcionamiento del teclado matricial y el display en la tarjeta de desarrollo mediante un programa.

**Descripción:** El programa que se desarrolla en esta prueba revisa qué tecla del teclado ha sido presionada e imprime en el display el caracter correspondiente.

En la realización de este programa se utilizan funciones de inicialización y manejo del teclado y display. Estas funciones fueron desarrolladas para facilitar la programación de dichos dispositivos y se encuentran documentadas en el apéndice A “Compendio de Funciones”.

Para llevar a cabo esta prueba se conecta la tarjeta de desarrollo a una computadora mediante el cable USB, se conecta el teclado matricial al Puerto G y el display al puerto H, tal como se describe en el capítulo 3.

**Desarrollo:**

1. Crear un nuevo proyecto en *CodeWarrior* seleccionando el *HC12 Serial Monitor* como opción de programación del microcontrolador.
2. Agregar al proyecto los archivos que contienen el manejo del LCD, el teclado y las variables de temporizadores (para inicializar el display) tal como se muestra en la Figura 6.26.

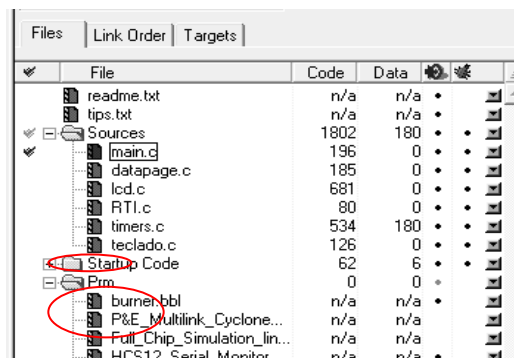


Figura 6.26 Archivos de control de temporizadores, teclado y display

3. Dentro del archivo *main.c* se escribe el programa siguiente:

```
#include <hidef.h> /* common defines and macros */
#include <MC9S12NE64.h> /* derivative information */
#pragma LINK_INFO DERIVATIVE "mc9s12ne64"

#include "general_functions.h"
#include "lcd.h"
#include "timers.h"
#include "teclado.h"

void main(void) {
    //inicializacion
    unsigned char temp,tecla=0,tvalue=0;
    EnableInterrupts;
    setPLL(1,1);
    RTI_init();
    timers_init();
    lcd_init();
    init_teclado();

    temp=create_timer(DOWN,1,SEC);//creacion de variable temporizador

    for(;;){
        lcd_comand(CLEAR); //limpiar display
        lcd_write_string("Tecla: ");
        tecla=leer_tecla(); //obtener numero de tecla presionada

        switch(tecla){
            case 1:tvalue='1';
            break;
            case 2:tvalue='2';
            break;
            case 3:tvalue='3';
            break;
            case 4:tvalue='A';
            break;
            case 5:tvalue='4';
            break;
            case 6:tvalue='5';
            break;
            case 7:tvalue='6';
            break;
            case 8:tvalue='B';
            break;
            case 9:tvalue='7';
            break;
            case 10:tvalue='8';
            break;
            case 11:tvalue='9';
            break;
            case 12:tvalue='C';
            break;
            case 13:tvalue='*';
            break;
            case 14:tvalue='0';
            break;
            case 15:tvalue='#';
            break;
            case 16:tvalue='D';
            break;
            default:tvalue='-';
            break;
        }

        lcd_data(tvalue);

        while(read_timer_ms(temp)!=0);
        reset_timer_value(temp,1,SEC);//renovar timer despues de 1 segundo

    }
}
```

**Figura 6.27** Programa para control de teclado y display

4. Una vez revisado el código, se programa el microcontrolador desde la opción *Project* → *Debug*. El resultado de cagar el código anterior se observa en la Figura 6.28, al presionar una tecla se observa en el display el caracter correspondiente.





Figura 6.28 Control de teclado–display

**Resultados:** En esta prueba se comprobó satisfactoriamente que a partir de las funciones desarrolladas para el manejo del display y teclado es sencillo crear programas que interactúen con estos dispositivos conjuntamente y que el funcionamiento de dichos dispositivos presenta un desempeño satisfactorio.

## 6.5 Pruebas de dispositivos de monitoreo y control

Entre los dispositivos con que cuenta la tarjeta de desarrollo, los dispositivos para monitoreo y control incluyen sensores, entradas optoacopladas y relevadores. Estos dispositivos le brindan al usuario la posibilidad de usar la tarjeta de desarrollo en una amplia gama de aplicaciones.

Con las siguientes pruebas se verifica el correcto funcionamiento de estos dispositivos.

### 6.5.1 Convertidor Analógico–Digital y relevador

**Prueba 7:** Control de relevador a partir de una entrada en el puerto Analógico–Digital con la señal de una fotorresistencia.

**Objetivo:** Comprobar el funcionamiento del convertidor Analógico–Digital y los relevadores conectados en la tarjeta de desarrollo.

**Descripción:** Teniendo conectada una fotorresistencia al bit 1 del puerto AD (puerto Analógico–Digital), se lee el valor del voltaje de la fotorresistencia con el módulo Analógico–Digital del microcontrolador, si el voltaje es mayor a 0.3 volts se cambia el estado del relevador y se enciende un led, de no ser así el estado del relevador no cambia y el led se mantiene apagado. Adicionalmente se muestra en el display el valor que detecta el módulo Analógico–Digital del microcontrolador.

**Desarrollo:**

1. Crear un nuevo proyecto en *CodeWarrior* y agregar los archivos correspondientes al manejo de temporizadores, display y convertidor Analógico–Digital, como se muestra en la Figura 6.29.

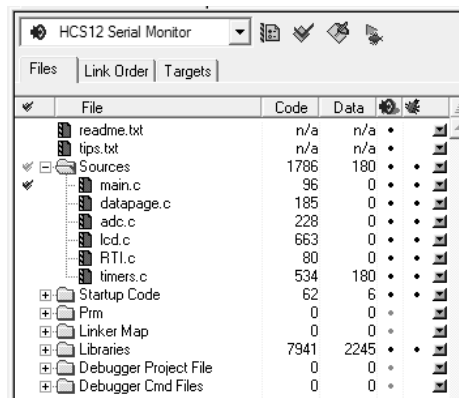


Figura 6.29 Proyecto fotorresistencia–relevador

2. En el archivo *main.c* se ingresa el siguiente código.

```
#include <hidef.h> /* common defines and macros */
#include <MC9S12NE64.h> /* derivative information */
#pragma LINK_INFO DERIVATIVE "mc9s12ne64"

#include "adc.h"
#include "lcd.h"
#include "timers.h"
#include "general_functions.h"

void main(void) {
    unsigned int lectura;
    unsigned char teap;
    DDRJ |= 0xC0; //los 2 bits mas significativos del puerto J como salidas
    DDRK=0xF0;
    RTI_init();
    setPLL(1.1);
    timers_init();
    lcd_init();
    teap=create_timer(DOWN,1,SEC);
    for(;;) {
        lcd_command(CLEAR);
        lcd_write_string("voltaje:");
        lcd_command(SECOND_LINE);
        lectura=adc_convert(1);
        if(lectura<100){
            PTJ_PTJ7=0;
            PORTK_BIT7=0;
        }
        else{
            PTJ_PTJ7=1;
            PORTK_BIT7=1;
        }
        adc_toint(lectura);
        lcd_data(' ');
        lcd_data('V');
        while(read_timer_ms(teap));
        reset_timer_value(teap,1,SEC);
    }
}
```

Figura 6.30 Código de prueba 7

3. Una vez que se revisa el código, se programa el microcontrolador mediante el *Monitor Serie* con la opción *Debug* del menú superior y se observa el funcionamiento del programa en ejecución. Si el nivel de luz que se detecta es bajo el relevador y el led no cambian de estado, el led se mantiene apagado. Cuando el sensor detecta un nivel de luz por arriba del valor fijado el relevador cambia su estado y el led se enciende. En el display se envía un mensaje que muestra el valor de voltaje correspondiente a la intensidad de luz detectada por el sensor, este voltaje está limitado al voltaje de entrada del ADC tomado de la fuente de 3.3 V del sistema.



Figura 6.31 Cambio en la intensidad de luz que detecta la fotorresistencia y salidas programadas

**Resultados:** Con esta prueba se verificó satisfactoriamente el uso de funciones de lectura de señales analógicas con el módulo ADC del microcontrolador y la manipulación de los relevadores. Adicionalmente se comprobó el manejo del sensor de luminosidad (fotorresistencia).

## 6.5.2 Entradas optoacopladas

**Prueba 8:** Manipulación de entradas optoacopladas de la tarjeta de desarrollo.

**Objetivo:** Desarrollar un programa que demuestre el funcionamiento de los circuitos optoacopladores con los que cuenta la tarjeta de desarrollo.

**Descripción:** Los dispositivos optoacopladores permiten aislar eléctricamente dos circuitos, cambiar niveles de voltaje y evitar que el ruido de uno de los circuitos se transfiera al otro (ver tema 3.5.3). Se utilizan las entradas optoacopladas con las que cuenta la tarjeta de desarrollo cuando las entradas provienen de dispositivos que operan a niveles de voltaje o corriente superiores a los manejados por el microcontrolador MC9S12NE64.

En esta prueba se desarrolla un programa sencillo que monitorea el bit 2 del puerto J en espera de una señal de entrada proveniente del optoacoplador 1 de la tarjeta de desarrollo. En caso de que exista dicha entrada se encienden los leds conectados al puerto K, en caso contrario los leds se mantienen apagados.

Para realizar esta prueba se conecta la tarjeta de desarrollo mediante el cable USB a la computadora, configurando como fuente de alimentación la salida de 5v del puerto USB de la computadora y se utiliza este mismo puerto para programar el microcontrolador (a través del *Monitor Serie*). Se conecta a la entrada optoacoplada 1 un voltaje aproximado de 9V y se utiliza un multímetro para revisar el voltaje a la entrada del circuito optoacoplador y a la salida del mismo.

### Desarrollo

1. Se crea un nuevo proyecto en el IDE *Code Warrior*, seleccionando *HC12 Serial Monitor* como opción de programación del microcontrolador.
2. En el archivo *main.c* del proyecto se escribe el siguiente código:

```
#include <hidef.h> /* common defines and macros */
#include <MC9S12NE64.h> /* derivative information */
#pragma LINK_INFO DERIVATIVE "mc9s12ne64"

void main(void) {
    EnableInterrupts;
    DDRK=0xFF; // puerto K como salidas
    DDRJ=0x00; //Puerto J como entradas

    /*Ciclo repetitivo*/
    for(;;) {

        if(PTJ_PTJ2==1)
            PORTK=0xFF;
        else
            PORTK=0x00;
    }
}
```

**Figura 6.32** Código del programa de prueba de entradas optoacopladas

3. Una vez escrito el código, se depura y programa el microcontrolador (*Project* → *Debug*).
4. Se conecta la entrada de voltaje externo al optoacoplador 1 y se observa el valor de este voltaje con el multímetro, como se muestra en la Figura 6.33.

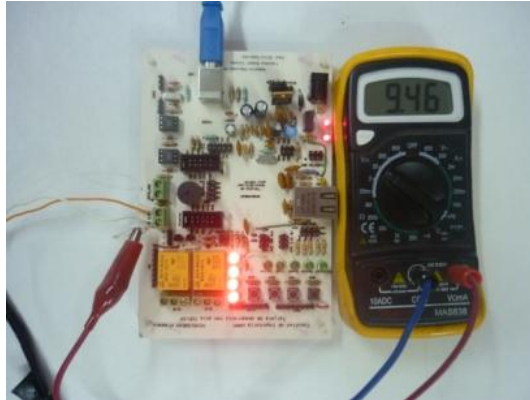


Figura 6.33 Entrada de voltaje alto que genera que se enciendan los leds

5. A continuación se mide el voltaje a la salida del optoacoplador para comprobar que con el nivel de voltaje de aproximadamente 3.3 V la regla implementada en el programa se cumple, es decir, existe un nivel alto en el bit 2 del puerto J, por lo tanto los leds del puerto K se encienden, tal como se muestra en la Figura 6.34.

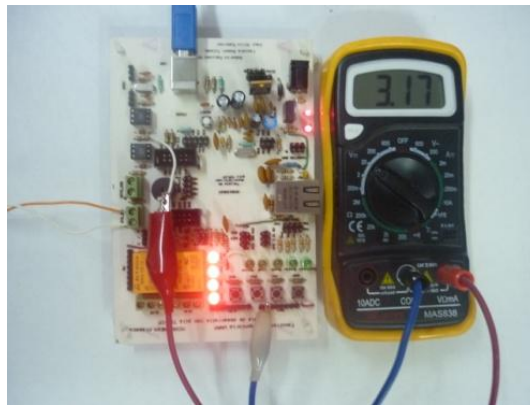


Figura 6.34 Voltaje de la entrada optoacoplada

6. Finalmente se interrumpe la entrada de voltaje externo de aproximadamente 9v y se mide nuevamente el voltaje a la salida del optoacoplador y como se observa en la Figura 6.35 el valor es cero, por lo tanto se lee un nivel bajo en el bit 2 del puerto J, por lo cual los leds del puerto K se apagan.

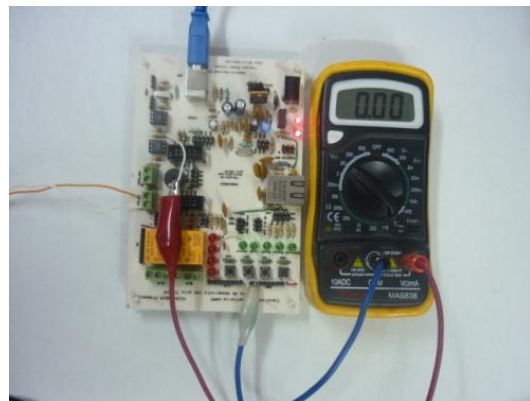


Figura 6.35 Entrada optoacoplada cuando se remueve la señal

**Resultados:** Con el programa desarrollado se comprueba el correcto funcionamiento de los circuitos optoacopladores con los que cuenta la tarjeta de desarrollo. Esta función abre la posibilidad de monitorear y

controlar procesos que manejen señales de magnitudes mayores a las que maneja el sistema de control, sin dañar el microcontrolador inmerso en la tarjeta de desarrollo.

## 6.6 Pruebas de Comunicación

Dentro de las interfaces de comunicación con las que cuenta la tarjeta de desarrollo podemos encontrar la interfaz serie a USB e interfaz Ethernet. Las siguientes pruebas describen el funcionamiento de estos medios de comunicación.

### 6.6.1 Comunicación serie

**Prueba 9:** Activación de comunicación serie.

**Objetivo:** Visualizar un mensaje enviado desde el microcontrolador en un programa terminal (en este caso se utiliza el programa *Hyper Terminal* de Windows).

**Descripción:** La importancia del manejo de la comunicación serie para esta tesis radica en la posibilidad de depuración de las funciones programadas que implementan los protocolos de la pila TCP/IP, esto se logra enviando mensajes a través del módulo SCIO (*Serial Communication Interface 0*) que se reciben en la computadora y se visualizan en un programa como el *Hyper Terminal* de Windows.

Para activar la comunicación serie del microcontrolador debemos configurar el módulo SCI de acuerdo a los siguientes pasos:

**Paso 1 de 5.** Seleccionar el *baud rate* (velocidad de transmisión) escribiendo el valor correspondiente en los registros del divisor de *baud rate* (SCIBDH/L) del microcontrolador.

Para calcular el *baud rate* de la comunicación serie se utiliza la siguiente expresión:

$$SCI\ baud\ rate = SCI\ module\ clock / (16 * SCIBR[12:0])$$

Donde *SCI baud rate* es la velocidad de comunicación deseada, *SCI module clock* es la velocidad de bus del microcontrolador y SCIBR[12:0] es el valor que se escribe en los bits del registro SCIBDH/L.

Para un baudaje de 9600:

$$\begin{aligned} SCIBR[12:0] &= SCI\ module\ clock / (16 * SCI\ baud\ rate) \\ SCIBR[12:0] &= 25\ 000\ 000 / (16 * 9600) = 162.7 \rightarrow 163 \\ SCIOBD &= 163; // para un baudaje de 9600 \end{aligned}$$

**Paso 2 de 5.** Configurar la longitud de palabra, paridad y otros bits de configuración escribiendo al registro de control 0 SCIOCR1.

$$SCIOCR1 = 0; // con cero se tiene modo normal, 8 bits sin paridad$$

**Paso 3 de 5.** Habilitar interrupciones, transmisor y receptor de la manera que se requiera escribiendo en el registro de control 2 SCIOCR2. Esta configuración selecciona automáticamente la función del puerto S asociada a la comunicación serie.

$$SCIOCR2 = SCIOCR2\_TE\_MASK | SCIOCR2\_RE\_MASK; // habilitar transmisor y receptor$$

Una vez configurado el módulo SCI, el procedimiento de transmisión es el siguiente:

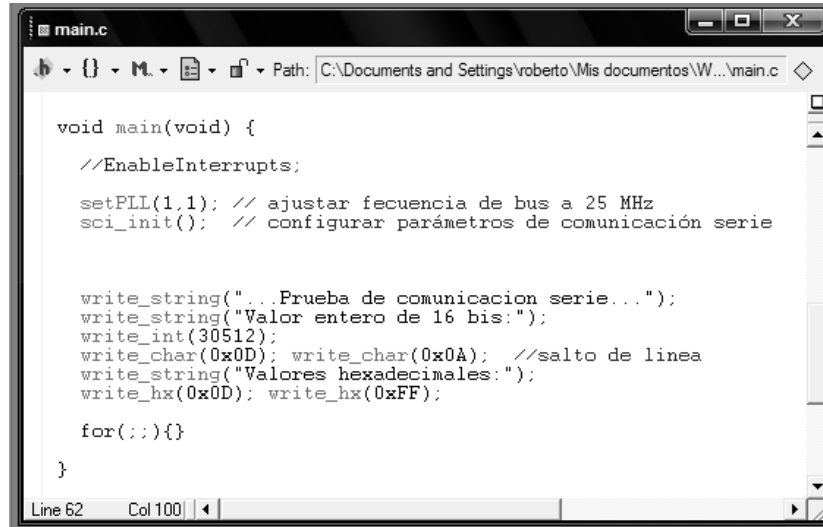
**Paso 4 de 5.** Leer el valor de la bandera TDRE (registro de transmisión vacío) del registro de estado 1 SCISR1.

**Paso 5 de 5.** Si la bandera TDRE vale 1 (lo cual indica que el registro de datos del SCI está vacío y listo para recibir un dato), escribir el dato en el registro de datos SCIDRL. Esta acción pondrá a cero la bandera TDRE, si se requiere enviar más datos se tendrá que esperar a que la bandera valga 1.

```
void write_char(char c){  
    while ((SCIOSR1 & SCIOSR1_TDRE_MASK)==0);  
    SCIODRL = c; }  
}
```

### Desarrollo

1. Crear un nuevo proyecto en el IDE *CodeWarrior* y añadir los archivos “sci.h” y “sci.c”, correspondientes a la configuración del módulo SCI y las funciones de impresión de caracteres y números por puerto serie.
2. Utilizando las funciones desarrolladas para facilitar la programación del envío de mensajes (revisar apéndice A), se escribe el código mostrado en la Figura 6.36.



```
main.c  
Path: C:\Documents and Settings\roberto\Mis documentos\W...\main.c  
  
void main(void) {  
    //EnableInterrupts;  
  
    setPLL(1,1); // ajustar frecuencia de bus a 25 MHz  
    sci_init(); // configurar parámetros de comunicación serie  
  
    write_string("...Prueba de comunicacion serie...");  
    write_string("Valor entero de 16 bis:");  
    write_int(30512);  
    write_char(0x0D); write_char(0x0A); //salto de linea  
    write_string("Valores hexadecimales:");  
    write_hx(0x0D); write_hx(0xFF);  
  
    for(;;){}  
}
```

Figura 6.36 Programa de comunicación Serie

3. Luego de conectar el sistema de desarrollo a la computadora mediante un cable USB, se programa el microcontrolador utilizando el *Monitor Serie*.
4. Una vez que el microcontrolador cuenta con el programa correspondiente para la comunicación serie, se abre en Windows el programa *Hyper Terminal* ubicado en *Inicio* → *Todos los programas* → *Accesorios* → *Comunicaciones* → *Hyper Terminal*. Configuramos las propiedades de la comunicación tal como se configuraron para el programa del módulo SCI, la configuración se realiza como muestra en la Figura 6.37.

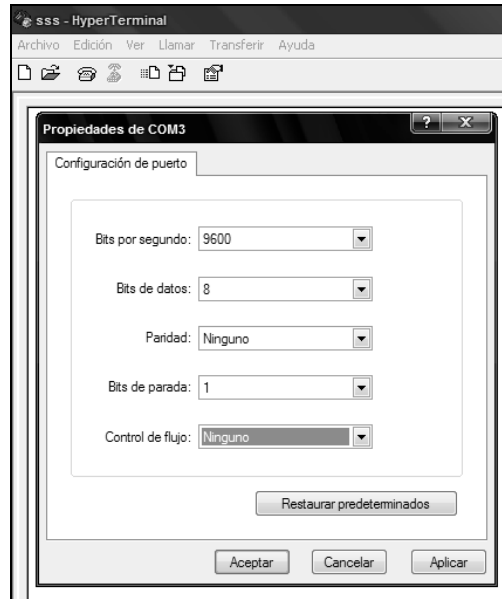


Figura 6.37 Configuración de hiperterminal de Windows

5. Iniciamos el programa previamente cargado al microcontrolador y en la ventana de la *Hyper Terminal* observamos un resultado como el que muestra la Figura 6.38.

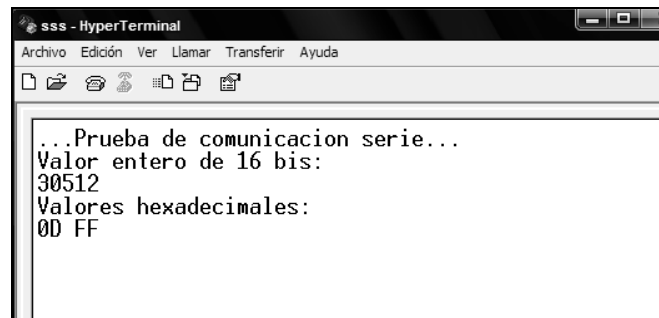


Figura 6.38 Resultados de la prueba de comunicación serie

**Resultados:** Con esta prueba se logró verificar el funcionamiento de la comunicación serie al enviar mensajes a través del módulo SCI del microcontrolador y con la ayuda del circuito convertidor FT232 integrado en la tarjeta de desarrollo se evitó el uso de puertos RS232, es decir, sólo se necesita un puerto USB, tanto en la computadora como en el sistema de desarrollo. El programa *Hyper Terminal*, que es un accesorio con el que cuentan la mayoría de los sistemas operativos Windows, permitió la visualización de los mensajes.

Es importante mencionar que mientras se utiliza el puerto USB para enviar mensajes serie no es posible utilizar el modo de depuración del *Monitor Serie* controlado desde el IDE *CodeWarrior*, ya que ambos hacen uso del mismo puerto.

### 6.6.2 Comunicación de protocolos de red

Las pruebas de los protocolos de comunicación se realizan de forma modular comenzando por los niveles más bajos y concluyendo con los protocolos de más alto nivel.

Para realizar las pruebas correspondientes a los protocolos de red se crea una red local entre la tarjeta de desarrollo y una computadora, asignando direcciones de red a cada uno que pertenezcan a la misma subred.

En la computadora con sistema operativo Windows XP estos valores se ajustan entrando en *Panel de Control* → *Conexiones de Red* → *Conexión de Área Local* → *General* → *Protocolo Internet (TCP/IP)* → *Propiedades*. En la Figura 6.39 se muestra la configuración asignada a la computadora.

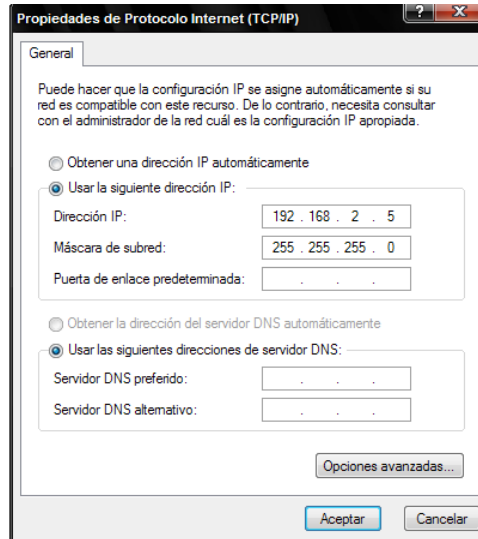


Figura 6.39 Configuración de red asignada a la computadora

El valor de la dirección de red, la dirección física y la máscara de subred del microcontrolador se ajusta en el archivo *address.c* del software del controlador de red proporcionado por Freescale. En la Figura 6.40 se muestran los valores asignados a la configuración del microcontrolador.

```

address.c
Path: C:\Documents and Settings\robeto\Mis documentos\Warrior\pilaTCP_FAB\pruebas...\address.c
/*****
*
* (c) Freescale Inc. 2004 All rights reserved
*
* File Name      : address.c
* Description    : This file contains definition of hardware and protocol (IP)
*                  addresses of the device
*
* Version       : 2.1
* Date         : 02/04/04
*
*****/
#include "MOTYPES.h"

const tU08 hard_addr [6] = { 0x01, 0x23, 0x45, 0x56, 0x78, 0x9a };

const tU08 prot_addr [4] = { 192, 168, 2, 3 };
const tU08 netw_mask [4] = { 255, 255, 255, 0 };
const tU08 defw_addr [4] = { 192, 168, 2, 1 };
const tU08 brcs_addr [4] = { 192, 168, 2, 255 };

```

Figura 6.40 Direcciones asignadas al microcontrolador

Una vez realizadas las configuraciones que permiten la comunicación en red local entre ambos dispositivos, conectamos mediante un cable de red cruzado (revisar capítulo 3) la tarjeta de desarrollo y la computadora utilizando como fuente de alimentación de la tarjeta de desarrollo el módulo USB o una fuente externa de 5 V a 12 V.

**Prueba 10:** Comunicación a nivel físico, de enlace (Ethernet) y de red (IP).

**Objetivo:** Con esta prueba se muestra la existencia de comunicación a nivel físico y de enlace entre la computadora y la tarjeta de desarrollo y se verifica el estado de la conexión entre ambos dispositivos a nivel de red.

**Descripción:** Para realizar esta prueba se carga al microcontrolador la pila de protocolos TCP/IP, cuya programación fue desarrollada en el capítulo 5 y a la cual previamente se le configuró la dirección de red, la dirección física y la máscara de subred al igual que a la computadora con la que se forma la red local.



La comprobación del uso de los protocolos ARP, IP e ICMP se lleva a cabo con el uso del comando denominado *Ping*, el cual comprueba el estado de la conexión de red entre equipos remotos por medio de paquetes de solicitud de *ECHO* y respuesta de *ECHO* (definidos en el protocolo de red ICMP) para determinar si una IP específica es accesible en una red.

El comando *Ping* corresponde al envío de un paquete de datos encapsulados en un datagrama IP. Este comando trabaja en el protocolo ICMP enviando un paquete denominado *ECHO*, el equipo que recibe este paquete contesta enviando una respuesta en la que incluye los datos que el emisor haya incluido en su solicitud. Si la solicitud se envía y recibe de manera correcta es señal de que el protocolo de red y los niveles inferiores funcionan correctamente.

## Desarrollo

1. Con la pila de protocolos programada en el microcontrolador de la tarjeta de desarrollo, se conecta el cable de red cruzado entre los dispositivos que componen la red local y, si la inicialización de los módulos EMAC y EPHY del microcontrolador es correcta, la computadora debe detectar la conexión física de manera inmediata, mostrando en la barra de tareas un icono que indica la presencia de una conexión y la velocidad de transmisión. En la ventana que se encuentra en *Panel de Control* → *Conexiones de Red* → *Conexión de Área Local* se encuentran detalles de la conexión física establecida. En la Figura 6.41 se muestra el resultado de una conexión establecida detectada por la computadora.

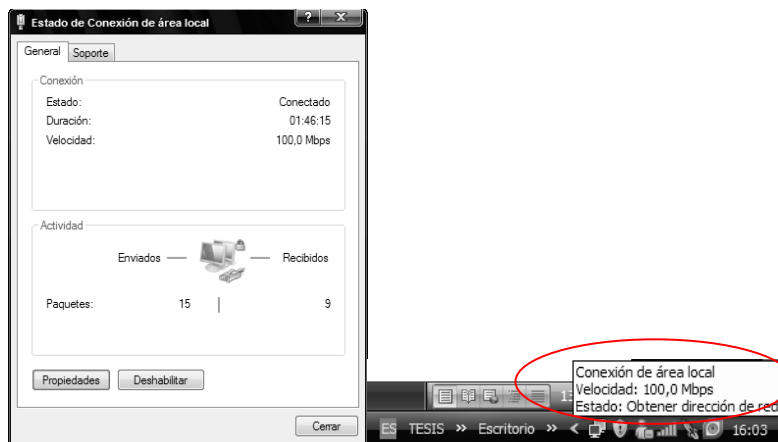


Figura 6.41 Establecimiento de la conexión física

Desde la computadora se envía la solicitud de *ECHO* indicando la dirección de red asignada al microcontrolador. Para tal efecto se abre en la computadora la ventana correspondiente al *símbolo del sistema* y se obtiene un resultado como el que presentado en la Figura 6.42, en la cual se muestra la cantidad de datos que se reciben en la respuesta y el tiempo que tarda en contestar el dispositivo destino. Esta información asegura que el dispositivo al que se le envía el mensaje *ECHO* está conectado a la red y provee una idea de lo congestionada que está la misma basado en el tiempo de respuesta.

```

C:\WINDOWS\system32\cmd.exe
Microsoft Windows XP [Versión 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\Documents and Settings\roberto>ping 192.168.2.3

Haciendo ping a 192.168.2.3 con 32 bytes de datos:

Respuesta desde 192.168.2.3: bytes=32 tiempo=21ms TTL=100
Respuesta desde 192.168.2.3: bytes=32 tiempo=21ms TTL=100
Respuesta desde 192.168.2.3: bytes=32 tiempo=21ms TTL=100
Respuesta desde 192.168.2.3: bytes=32 tiempo=21ms TTL=100

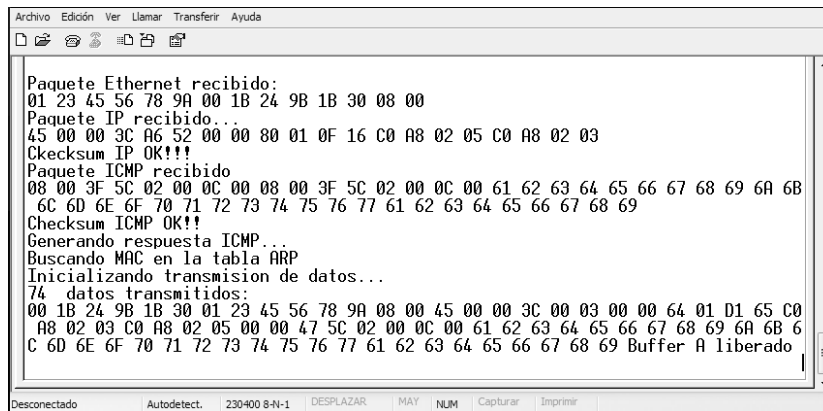
Estadísticas de ping para 192.168.2.3:
    Paquetes: enviados = 4, recibidos = 4, perdidos = 0
    (0% perdidos),
    Tiempos aproximados de ida y vuelta en milisegundos:
        Mínimo = 21ms, Máximo = 21ms, Media = 21ms

C:\Documents and Settings\roberto>

```

Figura 6.42 Resultado de la ejecución del comando PING desde la ventana del símbolo del sistema de Window

- Con la finalidad de corroborar el funcionamiento de la pila TCP/IP programada en el microcontrolador, se abre la aplicación *Hyper Terminal* de Windows y se establece conexión con la tarjeta de desarrollo, como se realizó en la prueba 9, donde podemos ver paso a paso las acciones realizadas por la pila en lo que a los protocolos ARP, ICMP e IP refiere.



**Figura 6.43** Acciones realizadas por el microcontrolador ante un mensaje ECHO

En la Tabla 6.1 se muestra el contenido del paquete recibido por el microcontrolador, el cual se muestra en la Figura 6.43.

**Tabla 6.1** Interpretación de un paquete de red recibido por el microcontrolador

Tipo de paquete	Campos	Contenido	Descripción
Ethernet	Dirección destino	01 23 45 56 78 9A	Dirección física asignada al microcontrolador, en este caso es el dispositivo remoto
	Dirección Origen	00 1B 24 9B 1B 30	Dirección física de la computadora, es el dispositivo que envía
	Tipo (protocolo)	08 00	Código del protocolo IP, es quien debe procesar el paquete que se ha recibido
IP	Versión/Hlen	45	Paquete IPv4 de longitud igual a 5 palabras (o 20 bytes)
	TOS	00	Tipo de servicio, su valor es cero
	Longitud total	00 3C	Longitud total del paquete IP, incluye encabezado IP y datos, el valor se muestra en hexadecimal (3C), corresponde a 60 bytes
	Número de identificación	A6 52	Valor en hexadecimal del número que identifica al paquete IP recibido
	Flags/ desplazamiento	00 00	Corresponde a un paquete no fragmentado
	TTL	80	Valor hexadecimal del tiempo de vida del paquete IP, se expresa en segundos y corresponde a un valor de cimal de 128
	Protocolo	01	Indica el código del protocolo superior que debe procesar el paquete IP, en este caso corresponde al protocolo ICMP
	Checksum	0F 16	Valor hexadecimal de la suma de verificación IP incluida en el paquete recibido
IP origen	C0 A8 02 05	Valor hexadecimal de la dirección IP de la computadora, 192.168.2.5	

	IP destino	C0 A8 02 03	Valor hexadecimal de la dirección IP del microcontrolador, 192.168.2.3
ICMP	Tipo	08	Solicitud de ECHO
	Código	00	cero
	Checksum	3F 5C	Valor hexadecimal de la suma de verificación ICMP incluida en el paquete recibido
	Identificador	02 00	Valor hexadecimal del número de identificación del paquete ICMP recibido
	Número de secuencia	0C 00	Valor hexadecimal del número de secuencia del paquete ICMP recibido
	Datos	61, 62, 63...	Datos enviados en la solicitud, corresponden a las letras del abecedario, a,b,c...

El paquete que envía el microcontrolador como respuesta a la solicitud de ECHO tiene el mismo formato que el paquete mostrado en la Tabla 6.1.

Los mensajes mostrados en la Figura 6.43 se generan en el microcontrolador, aquí podemos observar que cuando el microcontrolador recibe una solicitud de ECHO, el paquete es recibido por la capa Ethernet, en donde se especifica la dirección física del microcontrolador, seguida de la dirección física de la computadora (quien envía el mensaje ECHO) y el código del protocolo superior (en este caso IP = 0x0800). Al pasarle el paquete a IP, este protocolo lo valida, determina que es un paquete correcto y que contiene datos ICMP, quita los encabezados de IP y pasa el resto del paquete a ICMP. ICMP a su vez lo valida, determina que es un paquete que contiene una solicitud de ECHO y genera una respuesta. Para formar el paquete de respuesta se busca la dirección física de la computadora en la tabla ARP y se entrega el paquete correspondiente.

El comando *Ping*, como muestra la Figura 6.42, envía cuatro solicitudes y al final muestra estadísticas de la comunicación. En el primer paquete ICMP que se envía la computadora no conoce aún la dirección física del microcontrolador, por lo que antes de enviar el paquete ICMP envía un paquete ARP con una solicitud de la dirección física del microcontrolador, cuando éste le responde forma el paquete ICMP con solicitud de ECHO y lo envía. Por su parte el microcontrolador al recibir el paquete ARP de la computadora, toma sus datos y los almacena en su tabla ARP, de modo que cuando contesta a la solicitud de ECHO ya conoce su dirección. La Figura 6.44 muestra el procesamiento de la solicitud ARP en el microcontrolador.

```

sss - HyperTerminal
Archivo Edición Ver Llamar Transferir Ayuda

Paquete Ethernet recibido:
FF FF FF FF FF FF 00 1B 24 9B 1B 30 08 06
Paquete ARP recibido...
00 01 08 00 06 04 00 01 00 1B 24 9B 1B 30 C0 A8 02 05 00 00 00 00 00 C0 A8 02
 03 Solicitud ARP
Iniciando transmision de datos...

60 datos transmitidos:
00 1B 24 9B 1B 30 01 23 45 56 78 9A 08 06 00 01 08 00 06 04 00 02 01 23 45 56 78
F EB A5 A5 E9 AD 30 Buffer A liberado

```

Figura 6.44 Procedimiento de la solicitud ARP en el microcontrolador

**Resultados:** Esta prueba verifica el correcto funcionamiento de los niveles más bajos de la pila TCP/IP, correspondientes al nivel físico, de enlace y de red, mediante el uso del comando *Ping* y corrobora dicho funcionamiento con una serie de mensajes enviados a la computadora a través del módulo SCI del microcontrolador, mensajes donde se muestran las acciones que lleva a cabo la pila de protocolos TCP/IP programada en el microcontrolador.

**Prueba 11:** Comunicación a nivel de transporte y aplicación (TCP y HTTP).

**Objetivo:** Visualizar en un navegador de la computadora una página HTML almacenada en el microcontrolador.

**Descripción:** Las aplicaciones web se programan bajo el protocolo HTTP que a su vez hace uso del protocolo de transporte TCP. La forma de probar estos protocolos es haciendo una solicitud HTTP desde la computadora, abriendo un navegador web e indicando la dirección IP del microcontrolador y el nombre de un recurso almacenado en su memoria. Si la implementación de los protocolos es correcta, el microcontrolador responde enviando el recurso que el navegador despliega en su área para visualización del recurso. Si se presenta algún error en algún punto de la comunicación, la página no se despliega y el error se detecta con ayuda de la *Hyper Terminal*, mediante los mensajes que se envían del microcontrolador a la computadora. Por ejemplo, si el cálculo del *checksum* es incorrecto, se envía un mensaje notificando la situación, entonces se puede revisar esa parte del código de la pila y corregir el problema.

En esta prueba debemos contar con una página web desarrollada en lenguaje HTML, la cual se debe convertir a formato ASCII y almacenarse en la memoria del microcontrolador, de tal manera que cuando el microcontrolador detecte la solicitud de este recurso, el protocolo HTTP busca dicho recurso y lo envía a la computadora que realizó la solicitud.

## Desarrollo

1. Desarrollar una página web denominada “index.html” en lenguaje HTML.
2. Convertir a formato ASCII la página web previamente desarrollada, utilizando la herramienta proporcionada por *Freescade*, la cual al ser ejecutada proporciona dos archivos, denominados “index.c” e “index.h”, mismos que se añaden a la carpeta WEB de la pila TCP/IP.
3. En el archivo “server.c” de la pila TCP/IP se agregara el nuevo recurso del servidor como se muestra en la Figura 6.45.

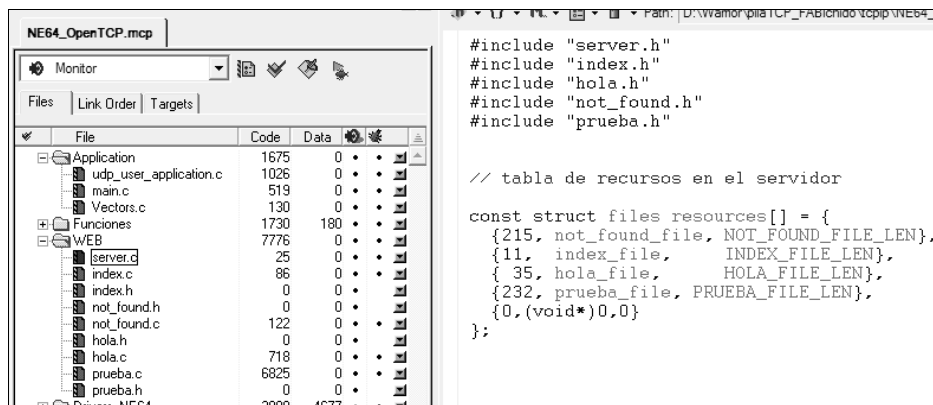


Figura 6.45 Estructura de carpetas de la pila TCP/IP

4. Una vez almacenada la página web en los archivos de la pila, se programa nuevamente el microcontrolador con la modificación hecha en la pila.
5. En un navegador web se solicita la página de inicio almacenada en el microcontrolador, la cual simplemente contiene un mensaje de bienvenida. En la Figura 6.46 se muestra el resultado de la prueba desde el navegador de la computadora.

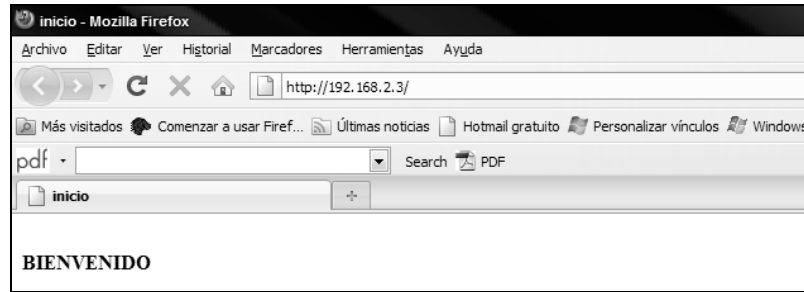


Figura 6.46 Petición de recurso web desde la computadora

6. Para corroborar el funcionamiento de la pila TCP/IP se abre una ventana de *Hyper Terminal*.

#### Resultados:

**Primera etapa:** establecimiento de la conexión TCP, se puede observar el intercambio de los tres paquetes TCP que establecen una conexión a nivel de transporte (Recepción de SYN, envío de SYN + ACK, recepción de ACK).

```

sss - HyperTerminal
Archivo Edición Ver Llamar Transferir Ayuda

Paquete Ethernet recibido:
01 23 45 56 78 9A 00 1B 24 9B 1B 30 08 00
Paquete IP recibido...
45 00 00 30 40 7C 40 00 80 06 34 F3 C0 A8 02 05 C0 A8 02 03
Ckecksum IP OK!!!
Paquete TCP recibido
0A 7B 00 50 6A 31 BE 57 00 00 00 70 02 FF FF CA 72 00 00 02 04 05 B4 01 01 04

02
Checksum TCP OK!!
paquete SYN recibido
enviar SYN + ACK...
Buscando MAC en la tabla ARP
Inicializando transmision de datos...
60 datos transmitidos:
A8 02 03 C0 A8 02 05 00 50 0A 7B 00 00 00 64 6A 31 BE 58 50 12 02 00 F4 C0 00 0

0 6F AD A5 ED AD 31 Buffer B liberado

Paquete Ethernet recibido:
01 23 45 56 78 9A 00 1B 24 9B 1B 30 08 00
Paquete IP recibido...
45 00 00 28 40 7F 40 00 80 06 34 F8 C0 A8 02 05 C0 A8 02 03
Ckecksum IP OK!!!
Paquete TCP recibido
0A 7B 00 50 6A 31 BE 58 00 00 00 65 50 10 FF FF F6 C1 00 00
Checksum TCP OK!!
Confirmacion de mi paquete SYN + ACK
... CONECTADO ...
Buffer A liberado

0:04:10 conectado Autodetect. 230400 8-N-1 DESPLAZAR MAY NUM Capturar Imprimir

```

Figura 6.47 Establecimiento de la conexión TCP

**Segunda etapa:** intercambio de información, solicitud del recurso por parte del cliente, entrega del recurso por parte del servidor (microcontrolador).

Recepción de la solicitud de recurso web y envío de la confirmación:

```

sss - HyperTerminal
Archivo Edición Ver Llamar Transferir Ayuda
Paquete Ethernet recibido:
01 23 45 56 78 9A 00 1B 24 9B
Paquete IP recibido...
45 00 01 BF 40 81 40 00 80 06 33 5F C0 A8 02 05 C0 A8 02 03
Checksum IP OK!!!
Paquete TCP recibido
0A 7B 00 50 6A 31 BE 58 00 00 00 65 50 18 FF FF A4 15 00 00 47 45 54 20 2F 20 48
54 54 50 2F 31 2E 31 0D 0A 48 6F 73 74 3A 20 31 39 32 2E 31 36 38 2E 32 2E 33 0
D 0A 55 73 65 72 2D 41 67 65 6E 74 3A 20 4D 6F 7A 69 6C 6C 61 2F 35 2E 30 20 28
57 69 6E 64 6F 77 73 3B 20 55 3B 20 57 69 6E 64 6F 77 73 20 4E 54 20 35 2E 31 3B
20 65 73 2D 45 53 3B 20 72 76 3A 31 2E 39 2E 31 2E 38 29 20 47 65 63 6
2 30 31 30 30 32 30 32 20 46 69 72 65 66 6F 78 2F 33 2E 35 2E 38 20 28 2E 4E 45
54 20 43 4C 52 20 33 2E 35 2E 33 30 37 32 39 29 0D 0A 41 63 63 65 70 74 3A 20 74
65 78 74 2F 68 74 6D 6C 2C 61 70 70 6C 69 63 61 74 69 6F 6E 2F 78 68 74 6D 6C 2
B 78 6D 6C 2C 61 70 70 6C 69 63 61 74 69 6F 6E 2F 78 6D 6C 3B 71 3D 30 2E 39 2C
2A 2F 2A 3B 71 3D 30 2E 38 0D 0A 41 63 63 65 70 74 2D 4C 61 6E 67 75 61 67 65 3A
20 65 73 2D 65 73 2C 65 73 3B 71 3D 30 2E 38 2C 65 6E 2D 75 73 3B 71 3D 30 2E 3
5 2C 65 6E 3B 71 3D 30 2E 33 0D 0A 41 63 63 65 70 74 2D 45 6E 63 6F 64 69 6E 67
3A 20 67 7A 69 70 2C 64 65 66 6C 61 74 65 0D 0A 41 63 63 65 70 74 2D 43 68 61 72
73 65 74 3A 20 49 53 4F 2D 38 38 35 39 2D 31 2C 75 74 66 2D 38 3B 71 3D 30 2E 3
7 2C 2A 3B 71 3D 30 2E 37 0D 0A 4B 65 65 70 2D 41 6C 69 76 65 3A 20 33 30 30 0D
0A 43 6F 6E 6E 65 63 74 69 6F 6E 3A 20 6B 65 65 70 2D 61 6C 69 76 65 0D 0A 0D 0A

Checksum TCP OK!!
Datos recibidos para la aplicacion
47 45 54 20 2F 20 Enviar confirmacion de datos
Buscando MAC en la tabla ARP
Iniciando transmision de datos...
60 datos transmitidos:
00 1B 24 9B 1B 30 01 23 45 56 78 9A 08 00 45 00 00 28 00 01 00 00 64 06 D1 76 C0
A8 02 03 C0 A8 02 05 00 50 0A 7B 00 00 65 6A 31 BF EF 50 10 02 00 F3 2A 00 0
0 6F AD A5 ED AD 31 Buffer A liberado
1:58:04 conectado Autodetect. 230400 8-N-1 DESPLAZAR MAY NUM Capturar Imprimir

```

Figura 6.48 Recepción de la solicitud de recurso y envío de la confirmación

Envío del recurso web y recepción de la confirmación:

```

sss - HyperTerminal
Archivo Edición Ver Llamar Transferir Ayuda
enviando datos del recurso
Comenzando a transferir el archivo
Buscando MAC en la tabla ARP
Iniciando transmision de datos...
158 datos transmitidos:
00 1B 24 9B 1B 30 01 23 45 56 78 9A 08 00 45 00 00 90 00 02 00 00 64 06 D
A8 02 03 C0 A8 02 05 00 50 0A 7B 00 00 65 6A 31 BF EF 50 18 02 00 1C 36 00 0
0 48 54 54 50 2F 31 2E 30 20 32 30 30 20 4F 4B 0D 0A 0D 0A 3C 68 74 6D 6C 3E 3C
68 65 61 64 3E 3C 74 69 74 6C 65 3E 20 69 6E 69 63 69 6F 20 3C 2F 74 69 74 6C 65
3E 3C 2F 68 65 61 64 3E 0D 0A 3C 62 6F 64 79 3E 3C 62 72 3E 3C 62 3E 42 49 45 4
E 56 45 4E 49 44 4F 3C 2F 62 3E 3C 2F 62 6F 64 79 3E 3C 2F 68 74 6D 6C 3E
Paquete Ethernet recibido:
01 23 45 56 78 9A 00 1B 24 9B 1B 30 08 00
Paquete IP recibido...
45 00 00 28 40 85 40 00 80 06 34 F2 C0 A8 02 05 C0 A8 02 03
Checksum IP OK!!!
Paquete TCP recibido
0A 7B 00 50 6A 31 BF EF 00 00 00 CD 50 10 FF 97 F5 2A 00 00
Checksum TCP OK!!
Llego confirmacion de datos enviados
0:15:49 conectado Autodetect. 230400 8-N-1 DESPLAZAR MAY NUM Capturar Imprimir

```

Figura 6.49 Envío del recurso y recepción de la confirmación

**Tercera etapa:** cierre de la conexión TCP, se determina que no hay más datos por enviar, se manda un paquete con bandera de FIN, se espera por el FIN del otro extremo de la conexión.

```

sss - HyperTerminal
Archivo Edición Ver Llamar Transferir Ayuda
El recurso se termino de transferir, cerrar conexion
no hay datos pendientes por confirmar, se cerrara la conexion
Buscando MAC en la tabla ARP
Inicializando transmision de datos. Envío de FIN
60 datos transmitidos:
00 1B 24 9B 1B 30 01 23 45 56 78 9A 08 00 45 00 00 28 00 03 00 00 64 06 D1 74 C0
A8 02 03 C0 A8 02 05 00 50 0A 7B 00 00 00 CD 6A 31 BF EF 50 11 02 00 F2 C1 00 0
0 48 54 54 50 2F 31
Paquete Ethernet recibido:
01 23 45 56 78 9A 00 1B 24 9B 1B 30 08 00
Paquete IP recibido...

Checksum IP OK!!!
Paquete TCP recibido
0A 7B 00 50 6A 31 BF EF 00 00 00 CE 50 10 FF 97 F5 29 00 00
Checksum TCP OK!!
Llego confirmacion de FIN Recepción de ACK de FIN
Buffer A liberado

Paquete Ethernet recibido:
01 23 45 56 78 9A 00 1B 24 9B 1B 30 08 00
Paquete IP recibido...
45 00 00 28 40 87 40 00 80 06 34 F0 C0 A8 02 05 C0 A8 02 03
Checksum IP OK!!!
Paquete TCP recibido
0A 7B 00 50 6A 31 BF EF 00 00 00 CE 50 11 FF 97 F5 28 00 00
Checksum TCP OK!!
Llego FIN, enviar ACK Recepción de FIN
Buscando MAC en la tabla ARP Envío de ACK
Inicializando transmision de datos...
60 datos transmitidos:
00 1B 24 9B 1B 30 01 23 45 56 78 9A 08 00 45 00 00 28 00 04 00 00 64 06 D1 73 C0
A8 02 03 C0 A8 02 05 00 50 0A 7B 00 00 00 CE 6A 31 BF F0 50 10 02 00 F2 C0 00 0
0 48 54 54 50 2F 31 Buffer B liberado
cerrar conexion

0:19:07 conectado Autodetect. 230400 8-N-1 DESPLAZAR MAY NUM Capturar Imprimir
    
```

Figura 6.50 Cierre de la conexión

Con la serie de pruebas realizadas a lo largo de este capítulo se demuestra el correcto funcionamiento del sistema de desarrollo diseñado e implementado a lo largo de esta tesis, primero comprobando el funcionamiento de los dispositivos periféricos de entrada-salida y de monitoreo y control que acompañan al microcontrolador MC9S12NE64 y posteriormente comprobando el funcionamiento de la comunicación con otros dispositivos a través del puerto USB o mediante el uso de la pila de protocolos TCP/IP desarrollada en el capítulo 5.

En este punto del trabajo de tesis se cuenta con la implementación de un sistema de desarrollo con pila TCP/IP basado en un microcontrolador de 16 bits, el cual tiene las características necesarias para ser utilizado en cualquier aplicación escolar e incluso aplicaciones sencillas de orden comercial e industrial.