Microcontroladores

Semana 14

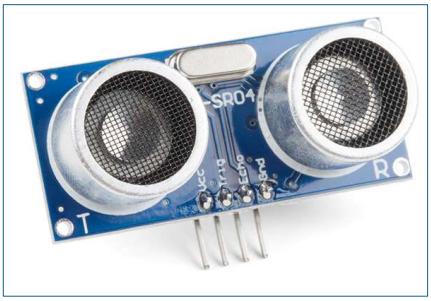
Por Kalun Lau

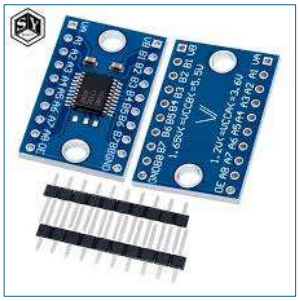
Agenda

- Sensor ultrasónico HC-SR04
- Módulo WS2812 (Neopixel)

El sensor HC-SR04

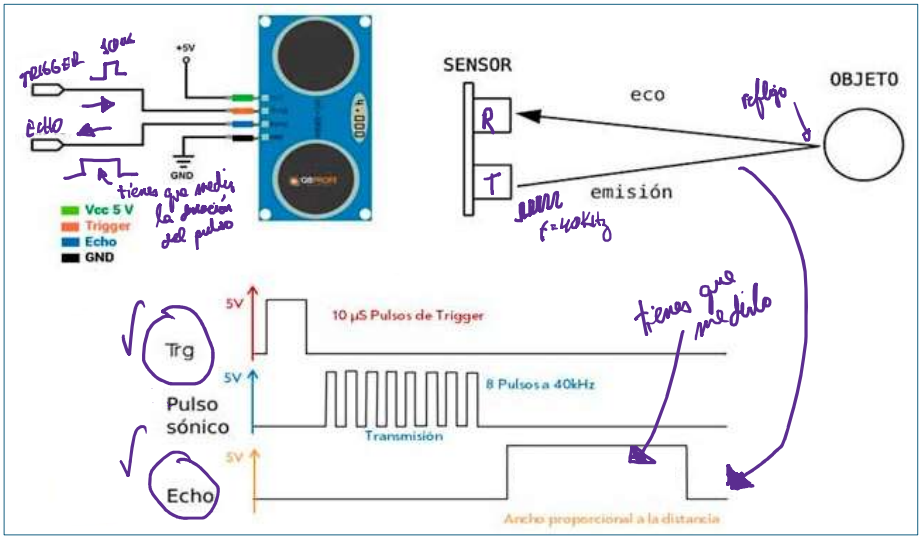
- El sensor HC-SR04 es un módulo que permite medir la distancia entre éste y un objeto entre un rango de 3cm y 300cm.
- Trabaja con alimentación de 5V por lo que se necesitará de un conversor de niveles lógicos 3.3V-5V, se recomienda emplear el TXS0108 en caso de que el microcontrolador trabaje en LVTTL
- Los transductores trabajan en 40KHz de frecuencia de sonido (rango ultrasónico – no es audible)





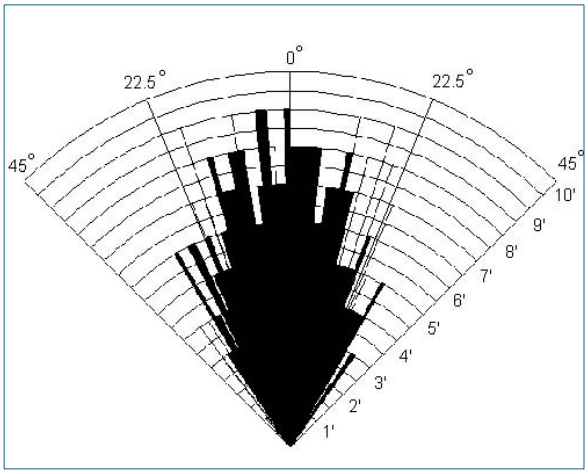
El sensor HC-SR04

- El sensor envía una ráfaga de 8 pulsos a 40KHz.
- El sonido viaja a una distancia doble con respecto al objeto donde se reflejó el sonido
- El sonido viaja a 340m/s a nivel del mar y a 20°C

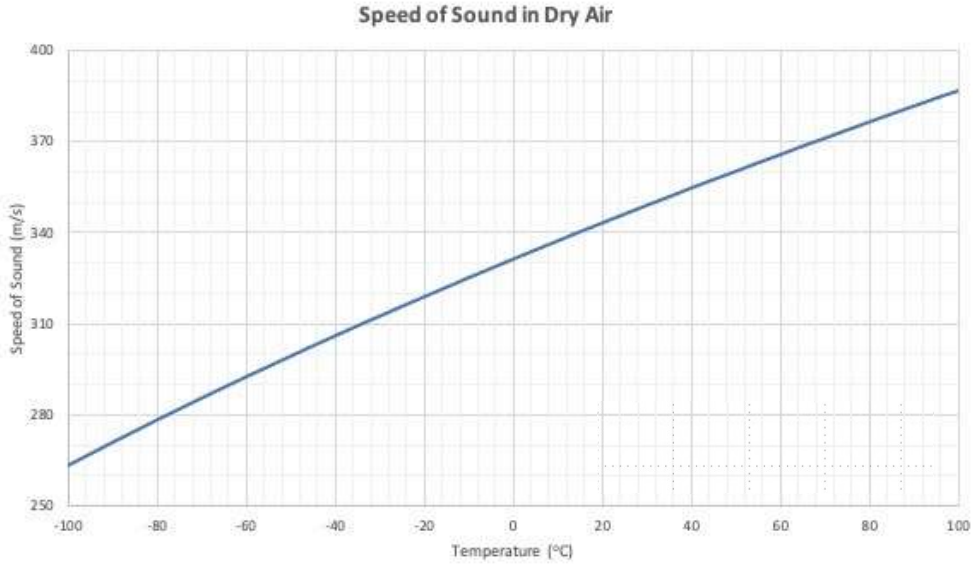


El sensor HC-SR04

- Tener en cuenta que la emisión ultrasónica no es muy directiva, a mayor distancia que recorre el sonido emitivo mayor será la dispersión.
- Mínima distancia a medir: 2cm
- Máxima distancia a medir: 4m (sin interferencias)

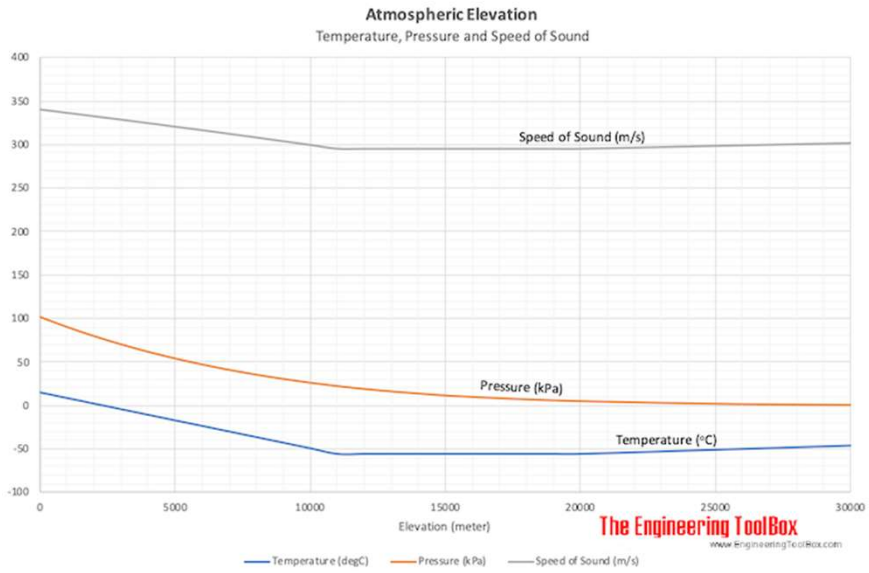


Velocidad del sonido con respecto a la temperatura



Velocidad del sonido con respecto a la temperatura y la presión atmosférica

Referencia:
https://www.engineeringtoolbox.com/elevation-speed-sound-air-d_1534.html



El sensor HC-SR04

- Cálculo de la distancia entre el sensor y el obstáculo:

1m = 1000cm

34300 cm/s

1s = 1×10^6 µs

0.0343000

Velocidad del sonido: 343 m/s (a nivel del mar y a 20°C)

↳ 0.0343 cm/µs ← convertido según resolución del Timer1 y las distancias a medir (de 2cm a 400cm)

4 metros en condiciones ideales

$$\text{Distancia} = \frac{(\text{velocidad del sonido}) \times \text{duración}}{2}$$

Timer1

$$\text{Distancia} = \frac{(0.0343)\ \text{cm/µs} \times \text{duración en microsegundos}}{2 \leftarrow \text{ida y vuelta}} = 0.01715 \times \text{duración} \Rightarrow \frac{\text{duración}}{58.31}$$

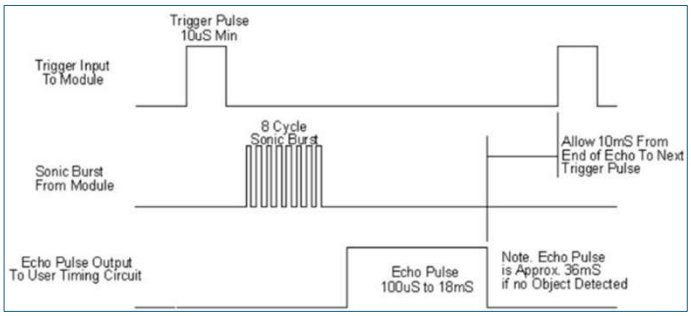
lo pasamos a cociente

1 µs x ct

$$\text{Distancia (cm)} = \frac{\text{Tiempo (µs)}}{58.31}$$

El sensor HC-SR04

- Posee dos puertos:
  - Trigger: El microcontrolador host le envía al puerto “trigger” del HC-SR04 un pulso activo en alto de 10us para que este último envíe ocho pulsos de sonido de 40KHz.
  - Echo: Luego del envío de los ocho pulsos de sonido de 40KHz el HC-SR04 enviará un pulso positivo al microcontrolador host con determinado ancho, este ancho representará el tiempo de recorrido del sonido (ida y vuelta) y mediante un cálculo matemático se obtendrá la distancia entre el módulo y el objeto.



¿Cómo hago para que el Timer1 funcione a 1us por cuenta?

Recordar configuración realizada para el relol de la semana 11:

- Configurar fuente de reloj a FOSC/4 teniendo en cuenta que se estará usando HFINTOSC a 32MHz
- Configurar prescaler a 1:8
- Inicialmente no habilitar el Timer1, recien empezará a contar cuando se reciba la señal de 1 por parte del ECHO del HCSR04 y detendrá el conteo cuando la señal pase a 0 en ECHO del HCSR04

```
//conf del TMR1
T1CLK = 0x01;       //Timer1 con FOSC/4
T1CON = 0x32;       //Timer1 prescaler 1:8 Timer1 OFF
```

Función para obtener la distancia en cm desde el HCSR04:

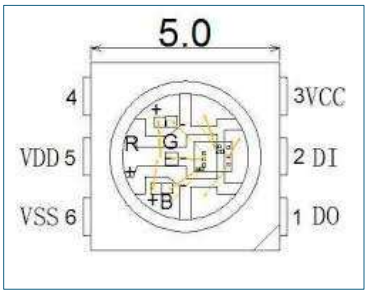
```
52  unsigned int distancia_HCSR04(void){
53      unsigned int distancia;
54      float calculo;
55      TMR1 = 0;               //cuenta inicial cero del Timer1
56      //Envio del pulso positivo de 10us a pin TRIG del HCSR04
57      LATCbits.LATC0 = 1;
58      __delay_us(10);
59      LATCbits.LATC0 = 0;
60      //Espera a que RC1 (pin ECHO del HCSR04) sea uno
61      while(PORTCbits.RC1 == 0);
62      T1CONbits.TMR1ON = 1;     //enciendes el Timer1
63      //Espera a que RC1 (pin ECHO del HCSR04) sea cero
64      while(PORTCbits.RC1 == 1);
65      T1CONbits.TMR1ON = 0;     //detienes el Timer1
66      calculo = TMR1 / 58.31;
67      distancia = calculo;
68      return distancia;
69  }
```

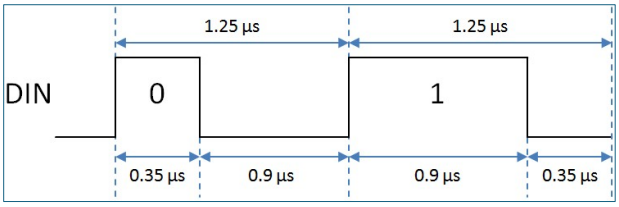
El WS2812 (LED Neopixel)

- Es un LED RGB en formato 5050 que integra un controlador.
- A través de una comunicación serial y protocolo propio es posible controlar la intensidad luminosa, así como la tonalidad resultante de la combinación de los colores rojo, verde y azul.

El WS2812 (LED Neopixel)

- Cada color corresponde a 8 bits por lo que el tital de datos que debe de recibir es 24 bits.
- Al colocar en cascada varios WS2812 se deberá de enviar para cada uno 24 bits.
- Se debe de tener en cuenta una temporización precisa en la comunicación serial de datos.



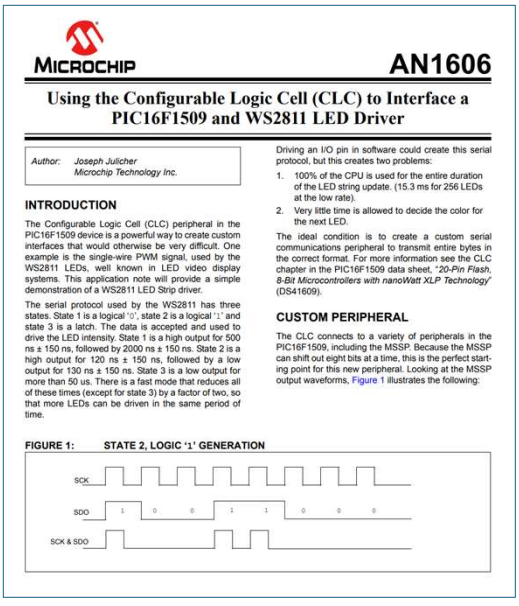


El WS2812 (LED Neopixel)

- La librería a utilizar (lib_ws2812.h lib_ws2812.c) fue elaborado por Kalun Lau teniendo como referencia el documento AN1606 elaborado por Josseph Julicher - Microchip.
- La formación de la trama de datos para el WS2812 se basó en una implementacion usando SPI, Timer2, CCP en PWM y CLB

```
/*Función de inicialización de
 comunicación para WS2812*/
void WS2812_INIT(void);

/*Función para enviar datos de
los colores del WS2812*/
void WS2812_DATA_SEND(unsigned char green, unsigned char red, unsigned char blue);
```



MICROCHIP **AN1606**

Using the Configurable Logic Cell (CLC) to Interface a PIC16F1509 and WS2811 LED Driver

Author: Joseph Julicher
Microchip Technology Inc.

INTRODUCTION

The Configurable Logic Cell (CLC) peripheral in the PIC16F1509 device is a powerful way to create custom interfaces that would otherwise be very difficult. One example is the single-wire PWM signal, used by the WS2811 LEDs, well known in LED video display systems. This application note will provide a simple demonstration of a WS2811 LED Strip driver.

The serial protocol used by the WS2811 has three states. State 1 is a logical '0', state 2 is a logical '1' and state 3 is a latch. The data is accepted and used to drive the LED intensity. State 1 is a high output for 500 ns ± 150 ns, followed by 2000 ns ± 150 ns. State 2 is a high output for 120 ns ± 150 ns, followed by a low output for 130 ns ± 150 ns. State 3 is a low output for more than 50 us. There is a fast mode that reduces all of these times (except for state 3) by a factor of two, so that more LEDs can be driven in the same period of time.

Driving an I/O pin in software could create this serial protocol, but this creates two problems:

1. 100% of the CPU is used for the entire duration of the LED string update. (15.3 ms for 256 LEDs at the low rate).
2. Very little time is allowed to decide the color for the next LED.

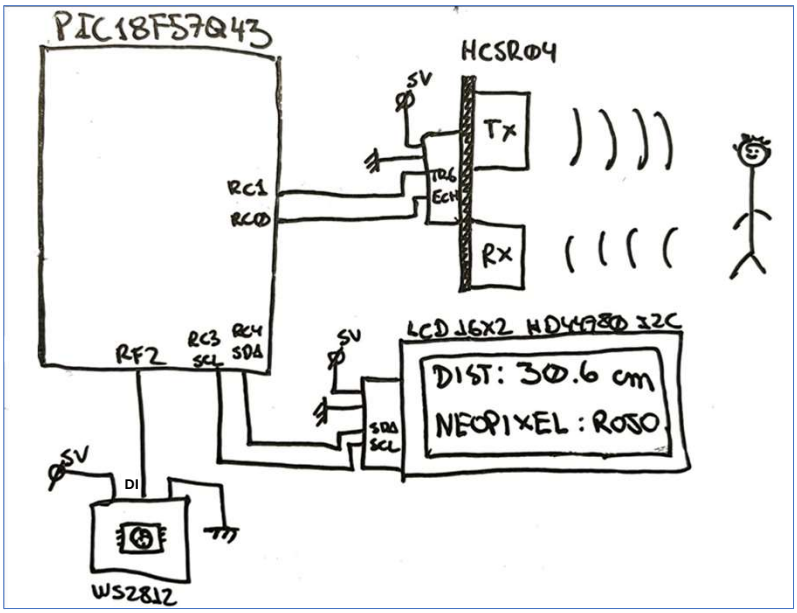
The ideal condition is to create a custom serial communications peripheral to transmit entire bytes in the correct format. For more information see the CLC chapter in the PIC16F1509 data sheet, "*20-Pin Flash, 8-Bit Microcontrollers with nanoWatt XLP Technology*" (DS41609).

CUSTOM PERIPHERAL

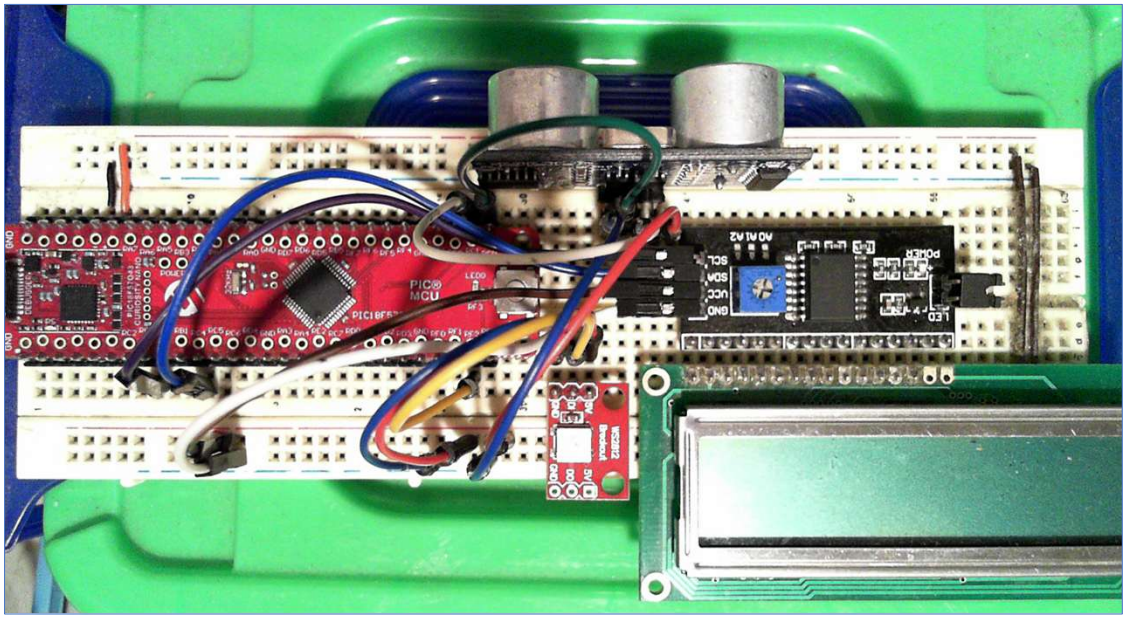
The CLC connects to a variety of peripherals in the PIC16F1509, including the MSSP. Because the MSSP can shift out eight bits at a time, this is the perfect starting point for this new peripheral. Looking at the MSSP output waveforms, Figure 1 illustrates the following:

FIGURE 1: STATE 2, LOGIC '1' GENERATION

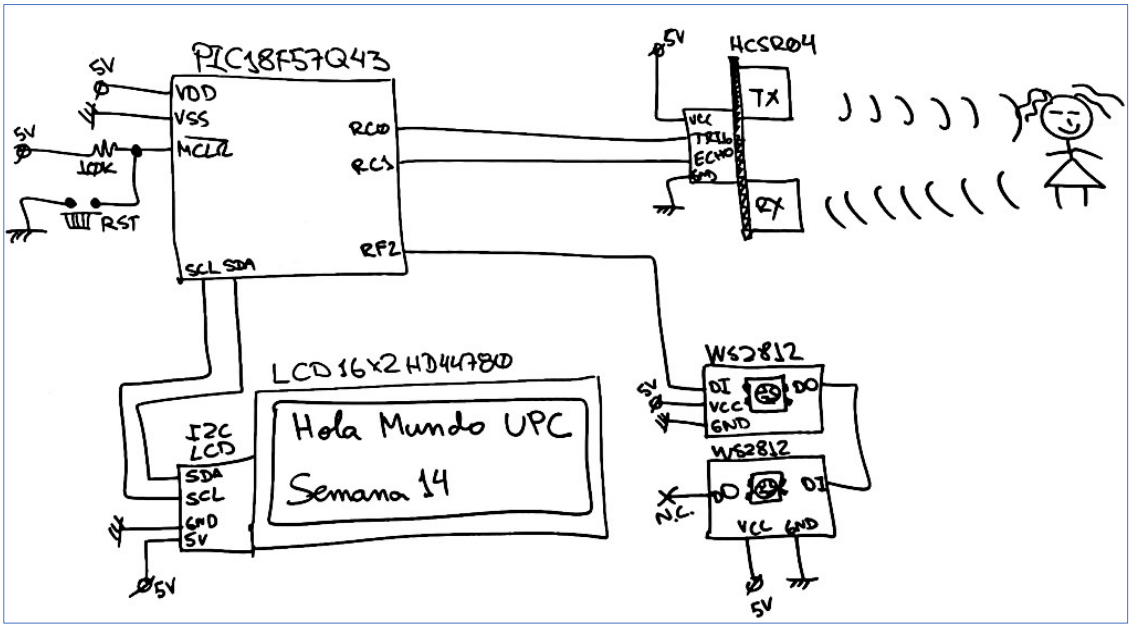
Implementar el siguiente circuito



Implementación



Implementar el siguiente circuito



Código ejemplo:

Ubica el error de funcionamiento en el WS2812

```
/*...6 lines */
#include <xc.h>
#include "cabecera.h"
#include "I2C_LCD.h"
#include "LIB_WS2812.h"
#define _XTAL_FREQ 32000000UL

void configuro(void){
    OSCCON1 = 0x60;
    OSCFRQ = 0x06;
    OSCEN = 0x40;
    //conf inicial de librerias
    I2C_LCD_INIT();
    WS2812_INIT();
    WS2812_DATA_SEND(0,0,0);
}

void splash_screen(void){
    I2C_POS_CURSOR(1,0);
    I2C_ESCRIBE_MENSAJE2("Hola mundo UPC");
    I2C_POS_CURSOR(2,0);
    I2C_ESCRIBE_MENSAJE2("Cargando");
    for(unsigned xvar=8;xvar<13;xvar++){
        I2C_POS_CURSOR(2,xvar);
        I2C_ESCRIBE_MENSAJE2("O");
        __delay_ms(100);
        I2C_POS_CURSOR(2,xvar);
        I2C_ESCRIBE_MENSAJE2("o");
        __delay_ms(100);
        I2C_POS_CURSOR(2,xvar);
        I2C_ESCRIBE_MENSAJE2(".");
        __delay_ms(100);
    }
    I2C_ESCRIBE_MENSAJE2("OK");
    __delay_ms(3000);
    I2C_BORRAR_LCD();
}

void main(void) {
    configuro();
    splash_screen();
    I2C_POS_CURSOR(1,0);
    I2C_ESCRIBE_MENSAJE2("WS2812 show");
    while(1){
        I2C_POS_CURSOR(2,0);
        I2C_ESCRIBE_MENSAJE2("Color: Rojo      ");
        WS2812_DATA_SEND(0,255,0);
        __delay_ms(1000);
        I2C_POS_CURSOR(2,0);
        I2C_ESCRIBE_MENSAJE2("Color: Verde     ");
        WS2812_DATA_SEND(255,0,0);
        __delay_ms(1000);
        I2C_POS_CURSOR(2,0);
        I2C_ESCRIBE_MENSAJE2("Color: Azul      ");
        WS2812_DATA_SEND(0,255,0);
        __delay_ms(1000);
    }
}
```

Codigo final:

```
#include <xc.h>
#include "cabecera.h"
#include "I2C_LCD.h"
#include "LIB_WS2812.h"
#define _XTAL_FREQ 32000000UL

void configuro(void){
    OSCCON1 = 0x60;
    OSCFRQ = 0x06;
    OSCEN = 0x40;
    //conf de las E/S
    TRISCbits.TRISC0 = 0;   //RC0 salida
    ANSELCbits.ANSELC0 = 0; //RC0 digital
    TRISCbits.TRISC1 = 1;   //RC1 entrada
    ANSELCbits.ANSELC1 = 0; //RC21 digital
    //conf del TMR1
    T1CLK = 0x01;       //Timer1 con FOSC/4
    T1CON = 0x32;       //Timer1 prescaler 1:8
    //conf inicial de librerias
    I2C_LCD_INIT();
    WS2812_INIT();
    WS2812_DATA_SEND(0,0,0);
}

void splash_screen(void){
    I2C_POS_CURSOR(1,0);
    I2C_ESCRIBE_MENSAJE2("Hola mundo UPC");
    I2C_POS_CURSOR(2,0);
    I2C_ESCRIBE_MENSAJE2("Cargando");
    for(unsigned xvar=8;xvar<13;xvar++){
        I2C_POS_CURSOR(2,xvar);
        I2C_ESCRIBE_MENSAJE2("O");
        __delay_ms(100);
        I2C_POS_CURSOR(2,xvar);
        I2C_ESCRIBE_MENSAJE2("o");
        __delay_ms(100);
        I2C_POS_CURSOR(2,xvar);
        I2C_ESCRIBE_MENSAJE2(".");
        __delay_ms(100);
    }
    I2C_ESCRIBE_MENSAJE2("OK");
    __delay_ms(3000);
    I2C_BORRAR_LCD();
}

unsigned int distancia_HCSR04(void){
    unsigned int distancia;
    float calculo;
    TMR1 = 0;               //cuenta inicial cero del Timer1
    //Envio del pulso positivo de 10us a pin TRIG del HCSR04
    LATCbits.LATC0 = 1;
    __delay_us(10);
    LATCbits.LATC0 = 0;
    //Espera a que RC1 (pin ECHO del HCSR04) sea uno
    while(PORTCbits.RC1 == 0);
    T1CONbits.TMR1ON = 1;    //enciendes el Timer1
    //Espera a que RC1 (pin ECHO del HCSR04) sea cero
    while(PORTCbits.RC1 == 1);
    T1CONbits.TMR1ON = 0;    //detie
    calculo = TMR1 / 58.31;
    distancia = calculo;
    return distancia;
}

void main(void) {
    configuro();
    splash_screen();
    I2C_POS_CURSOR(1,0);
    I2C_ESCRIBE_MENSAJE2("HCSR04 UPC");
    while(1){
        I2C_POS_CURSOR(2,0);
        I2C_ESCRIBE_MENSAJE2("Dist:");
        I2C_LCD_ESCRIBE_VAR_INT(distancia_HCSR04(), 3, 0);
        I2C_ESCRIBE_MENSAJE2(" cm   ");
        __delay_ms(500);
    }
}
```

Fin de la sesión