CLASE3: INTERRUPCIONES Y TIMER

1. DESCRIPCION Y MANIPULACION DEL VECTOR DE INTERRUPCION PARA MICROCONTROLADORES PIC

   1.1. DEFINICION

   Una interrupción es un evento interno del microcontrolador que cuando se presenta el caso, el PIC deja de ejecutar el programa principal y atenderá dicha interrupción y al terminar de ejecutar la interrupción continuará la ejecución del programa principal donde lo había dejado.

   1.2. MANIPULACION DE DIRECTIVAS FUNCIONES PARA HABILITAR INTERRUPCIONES

   En CCS C compiler podemos hacer uso de las interrupciones con la directiva #INT_XXX, donde XXX indican la función de interrupción que se usará.
   Los PICs 16F877A de 40 pines tienen 14 directivas de interrupción, las cuales se detallan en el siguiente cuadro:

Interrupción	Función
#INT_RTCC	Desborde de Timer 0
#INT_RB	Cambio en RB<4:7>
#INT_EXT	Cambio en RB0
#INT_AD	Fin de conversión A/D
#INT_TBE	Fin de transmisión USART
#INT_RDA	Sato recibido en USART
#INT_TIMER1	Desborde Timer 1
#INT_TIMER2	Desborde Timer 2
#INT_CCP1	Captura/Comparación en CCP1
#INT_CCP2	Captura/Comparación en CCP2
#INT_SSP	Envió/Recepción de datos serie síncrono
#INT_PSP	Dato entrante en puerto esclavo paralelo
#INT_BUSCOL	Colisión de bus I2C
#INT_EEPROM	Fin de escritura EEPROM

   Para habilitar alguna interrupción se usa el comando **enable_interrupts(nivel)**, nivel es una constante definida en el 16F877.h y genera el código necesario para activar la interrupción. Además se debe activar una máscara para interrupción global el cual activa las interrupciones, se representa con el comando **enable_interrupts(GLOBAL).**

2. INTERRUPCION EXTERNA

   2.1. DEFINICION

   Interrupción que interrumpe el programa principal mediante un flanco de bajada o subida en el PIN_B0

2.2. CONFIGURACION DE LAS DIRECTIVAS Y FUNCIONES

2.2.1.DIRECTIVA

La directiva que se utiliza es **#INT_EXT**

2.2.2.FUNCIONES

- ext_int_edge(H_TO_L):
  Flanco de bajada
- ext_int_edge(L_TO_H):
  Flanco de subida

3. DESCRIPCION Y MANIPULACION DE DIRECTIVA Y FUNCIONES PARA TIMER0

3.1. DEFINICION

El Timer0 es un módulo temporizador/contador ascendente de 8 bits, cuando trabaja con el reloj del PIC se le suele llamar temporizador y cundo los pulsos los recibe de una fuente externa a través de la patilla RA4/TOCKI se le llama contador. Al tener un registro de 8 bits la cuenta máxima que realiza será de 255.

Antes de explicar las directivas y funciones para el funcionamiento del timer0 en ccs c compiler primero definamos los siguientes conceptos:

- Frecuencia de oscilación (Fosc):
  Frecuencia de trabajo externo del PIC generado por un cristal de cuarzo, un resonador, un circuito RC, etc.
- Frecuencia interna de instrucciones (Fint):
  Frecuencia de reloj interno de instrucciones generada a partir de la frecuencia de oscilación externa. Para los microcontroladores PIC esta frecuencia no coincide con Fosc así que para mantener la compatibilidad dividirla por cuatro:

$$Fins = FOSC/4$$

$$Tins = 1/Fins$$

- Prescalador:
  El presacalador es un divisor de frecuencia usado para hallar el tiempo de desbordamiento. Los valores que toma el presacalador son 1, 2, 4, 8, 16, 32, 64, 128 o 256.
- Tiempo de desbordamiento
  Es el tiempo que le toma al microcontrolador en realizar la cuenta de 0 a 255, se representa por la siguiente ecuación:

$$T = 4/(FOSC * PRESCALER * (256 - VALOR\ A\ CARGAR\ EN\ EL\ TIMER))$$

3.2. CONFIGURACION DE LAS DIRECTIVAS Y FUNCIONES

La función para configurar el timer0 es:

Setup_timer_0(modo)

Modo se define en el siguiente cuadro

MODO
RTCC_INTERNAL
RTCC_EXT_L_TO_H
RTCC_EXT_H_TO_L
RTCC_DIV_1
RTCC_ DIV_2
RTCC_ DIV_4
RTCC_ DIV_8
RTCC_ DIV_16
RTCC_ DIV_32
RTCC_ DIV_64
RTCC_ DIV_128
RTCC_ DIV_256

Los distintos modos se pueden agrupar mediante el empleo del símbolo |

Ejemplo:

setup_timer_0(RTCC_INTERNAL|RTCC_DIV_2);

Función para escribir en el timer0 es:

Set_trimer0(VALOR)

Donde:

valor = entero de 8 bits

Función para leer el valor actual del registro:

Valor = get_timer0()

Para entender mejor veremos un ejemplo:

- Generar una señal cuadrada de un periodo de 2ms usando la interrupción timer0.

$$T = \left(\frac{4}{FOSC}\right) * PRESCALER * (256 - VALOR\ CARGA\ AL\ TIMER)$$

El valor de T es el semiperiodo de la señal por tanto valdrá 1ms, luego elegimos el prescalador DIV_8 y usaremos un cristal de cuarzo de 4Mhz.

En la ecuación:

$$1 * 10^{-3} = \left(\frac{4}{4 * 10^{6}}\right) * 8 * (256 - VALOR\ CARGA\ TIMER)$$

$$125 = 256 - VALOR\ CARGA\ TIMER$$

$$VALOR\ CARGA\ TIMER = 131$$

El valor que cargaremos en la función set_timer0 (valor) será 131.

```
#INCLUDE <16F877A.H>
#FUSES XT
#USE DELAY( CLOCK = 4M)
#DEFINE CARGA_TIMER0 131

#INT_RTCC
void RTCC_isr( ){
   output_toggle(PIN_B0);
   set_timer0( CARGA_TIMER0 );
}
void main ( ){

   setup_timer_0( RTCC_DIV_8 );
   set_timer0( CARGA_TIMER0 );
   enable_interrupts( INT_RTCC );
   enable_interrupts(GLOBAL);

   for(;;){

   }
}
```

4. DESCRIPCION Y MANIPULACION DE DIRECTIVA Y FUNCIONES PARA TIMER1

4.1. DEFINICION

El Timer1 es un módulo temporizador/contador de 16 bits, lo que significa que tiene dos registros (TMR1L y TMR1H). Al tener un registro de 16 bits la cuenta máxima que realiza será de 65536.

El timer1 tiene las mismas funciones que el timer0, la diferencia en el tamaño del registro, prescalador y la ecuación del tiempo de desbordamiento.

- Prescalador

  Los valores que toma el presacalador del timer1 son 1, 2, 4 y 8.

- Tiempo de desbordamiento

  Es el tiempo que le toma al microcontrolador en realizar la cuenta de 0 a 65535, se representa por la siguiente ecuación:

  $$T = (\frac{4}{FOSC}) * PRESCALER * (65536 - VALOR\ CARGA\ AL\ TIMER0)$$

4.2. CONFIGURACION DE LAS DIRECTIVAS Y FUNCIONES

La función para configurar el timer1 es:

Setup_timer_1(modo)

Modo se define en el siguiente cuadro

MODO
T1_DISABLED
T1_INTERNAL
T1_EXTERNAL
T1_EXTERNAL_SYNC
T1_CLK_OUT
T1_DIV_BY_1
T1_DIV_BY_2
T1_DIV_BY_4
T1_DIV_BY_8

Los distintos modos se pueden agrupar mediante el empleo del símbolo |
Ejemplo:

setup_timer_1 (T1_INTERNAL|T1_DIV_BY_2);

Función para escribir en el timer1 es:

Set_trimer1(VALOR)

Dónde: valor = entero de 16 bits

Función para leer el valor actual del registro:

Valor = get_timer1 ()

Para entender mejor veremos un ejemplo:

- Generar una señal cuadrada de un periodo de 2ms usando la interrupción timer1.

$$T = \left(\frac{4}{FOSC}\right) * PRESCALER * (65536 - VALOR\ CARGA\ AL\ TIMER1)$$

El valor de T es el semiperiodo de la señal por tanto valdrá 1ms, luego elegimos el prescalador DIV_8 y usaremos un cristal de cuarzo de 4Mhz.

En la ecuación:

$$1 * 10^{-3} = \left(\frac{4}{4 * 10^{6}}\right) * 8 * (256\ - VALOR\ CARGA\ TIMER1)$$

$$125 = 65536 - VALOR\ CARGA\ TIMER1$$

$$VALOR\ CARGA\ TIMER1 = 65411$$

```
#include <16F877a.h>
#FUSES XT
#use delay(clock = 4M)
#DEFINE CARGA_TIMER1 65411
#INT_TIMER1
void  TIMER1_isr(){
   set_timer1(CARGA_TIMER1);
}

void main() {
 setup_timer_1(T1_INTERNAL|T1_DIV_BY_8);
 enable_interrupts(INT_TIMER1);
 enable_interrupts(GLOBAL);
 set_timer1(CARGA_TIMER1);
 for(;;){
 }
}
```

5. DESCRIPCION Y MANIPULACION DE DIRECTIVA Y FUNCIONES PARA TIMER2

5.1. DEFINICION

El Timer2 es un módulo temporizador/contador de 8 bits, se puede emplear como base de tiempos para la modulación en ancho de pulso (PWM) mediante la utilización del módulo CCP.

- Prescalador
  Los valores que toma el presacalador del timer2 son 1, 4 y 16.
- Poscalador
  El postescalador es de 4bits teniendo los siguientes valores 0, 1,2…15
- Registro PR2
  Es utilizado para almacenar el final del valor de conteo, el Timer2 incrementa de 00h hasta alcanzar PR2 luego resetea a 00h.
- Tiempo de desbordamiento
  Es el tiempo que le toma al microcontrolador en realizar la cuenta de 0 a 255, se representa por la siguiente ecuación:

$$T = (\frac{4}{FOSC}) * PRESCALER * (VALOR\ CARGA\ PR2 + 1) * POSTSCALER$$

5.2. CONFIGURACION DE LAS DIRECTIVAS Y FUNCIONES

La función para configurar el timer2 es:

Setup_timer_2(modo, periodo, poscalador)

Modo se define en el siguiente cuadro

MODO
T2_DISABLED
T2_DIV_BY_1
T2_DIV_BY_4
T2_DIV_BY_16

Ejemplo:

Setup_timer_2 (T2_DIV_BY_4, 124,1)

Función para escribir en el timer0 es:

Set_trimer2 (VALOR)

Dónde: valor = entero de 8 bits

Función para leer el valor actual del registro:

Valor = get_timer2 ()

Para entender mejor veremos un ejemplo:

- Generar una señal cuadrada de un periodo de 2ms usando la interrupción timer1.

$$T = (\frac{4}{FOSC}) * PRESCALER * (VALOR\ CARGA\ PR2 + 1) * POSTSCALER$$

El valor de T es el semiperiodo de la señal por tanto valdrá 1ms, luego elegimos el prescalador DIV_8 y usaremos un cristal de cuarzo de 4Mhz.

En la ecuación:

$$1 * 10^{-3} = \left(\frac{4}{4 * 10^{6}}\right) * 4 * (VALOR\ CARGA\ PR2 + 1) * 2$$

$$125 = VALOR\ CARGA\ PR2 + 1$$

$$VALOR\ CARGA\ PR2 = 124$$

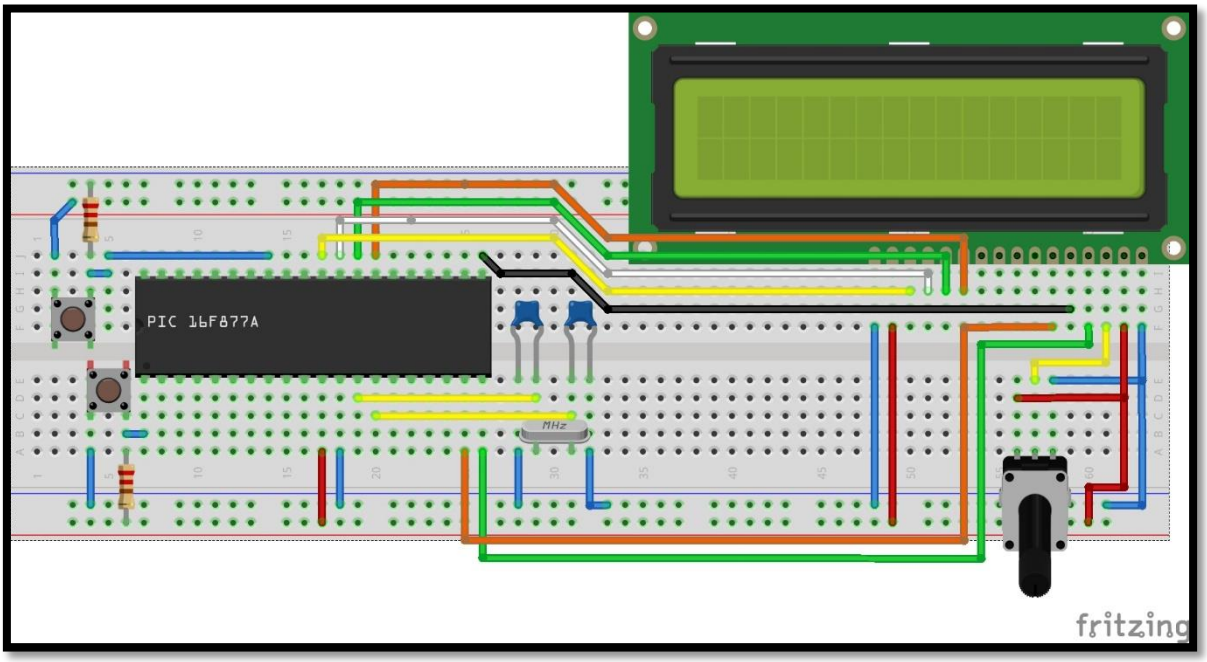
```
#INCLUDE <16F877A.H>
#FUSES XT
#USE DELAY( CLOCK = 4M)
#DEFINE PR2           124
#DEFINE POSTSCALER    2

#INT_TIMER2
void TMR2_isr( ){

}
void main ( ){

   setup_timer_2( T2_DIV_BY_4 , PR2 , POSTSCALER );
   enable_interrupts( INT_TIMER2 );
   enable_interrupts(GLOBAL);

   while (TRUE){
   }
}
```

6. EJEMPLOS DE LA CLASE3

6.1. Ejemplo1:

Conectar un pulsador en Modo pull-down en el PIN_B0 y realizar una interrupción externa, la interrupción se muestra en el lcd.

6.1.1.CONEXIÓN

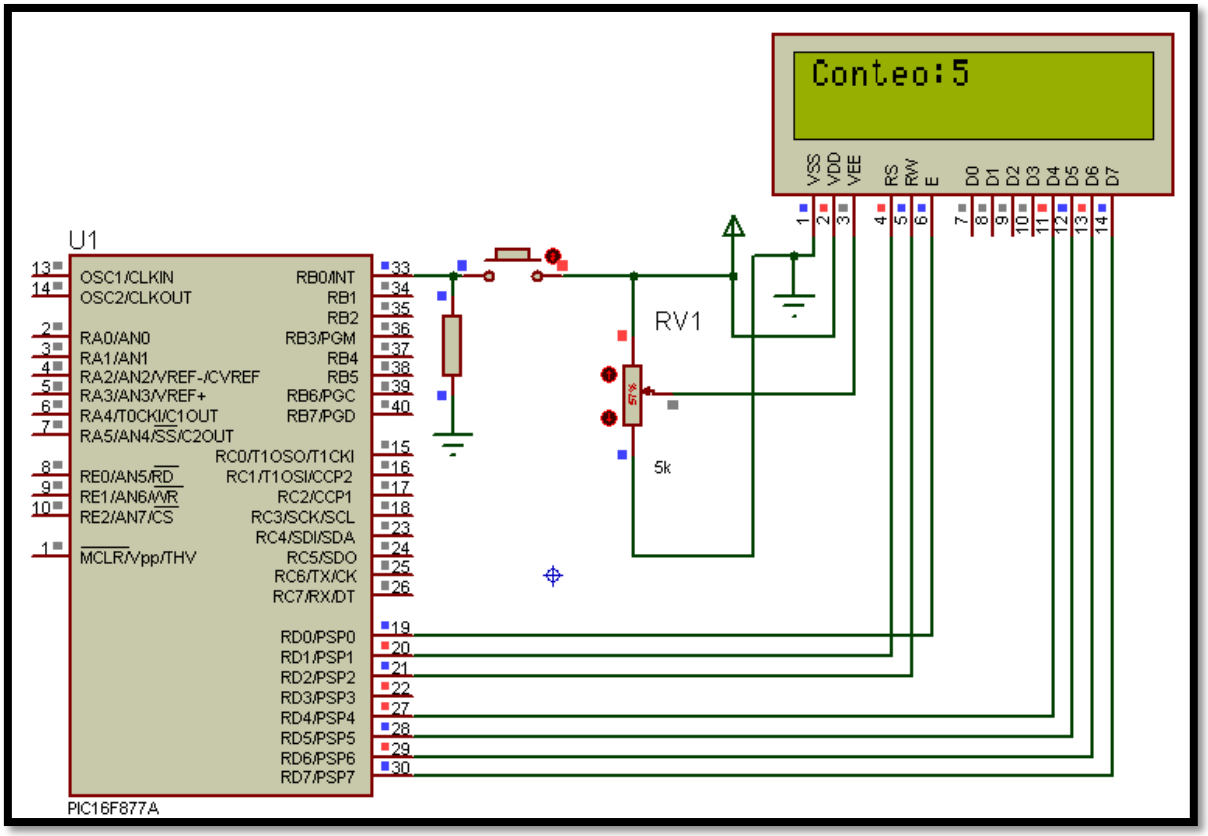


6.1.2.PROGRAMA

Ejemplo1.c

```
#include <16f877a.h>
#fuses    XT
#use      delay(clock=4M)
#include <lcd.c>
int a = 0;
#INT_EXT
void EXT_isr(){
   a = 1;
}
void main(){
   unsigned int16  i = 0;
   lcd_init();
   ext_int_edge(L_TO_H);
   enable_interrupts(INT_EXT);
   enable_interrupts(GLOBAL);
   for(;;){
      for(i = 0; i <= 100; i++){
         printf(lcd_putc,"\fConteo:%Lu",i);
         delay_ms(500);
         if(a == 1){
            printf(lcd_putc,"\fInterrupcion");
            delay_ms(100);
            a = 0;
         }
      }
   }
}
```

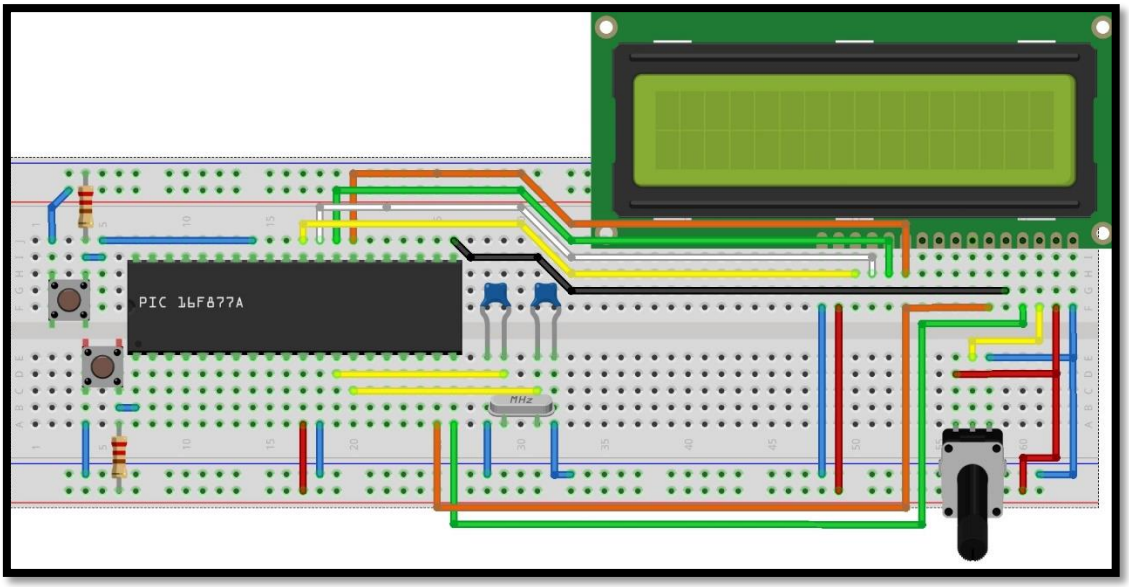
6.1.3.SIMULACION



6.2. Ejemplo2:

Realizar un reloj digital utilizando los temporizadores y mostrar la hora minuto y segundo en el LCD

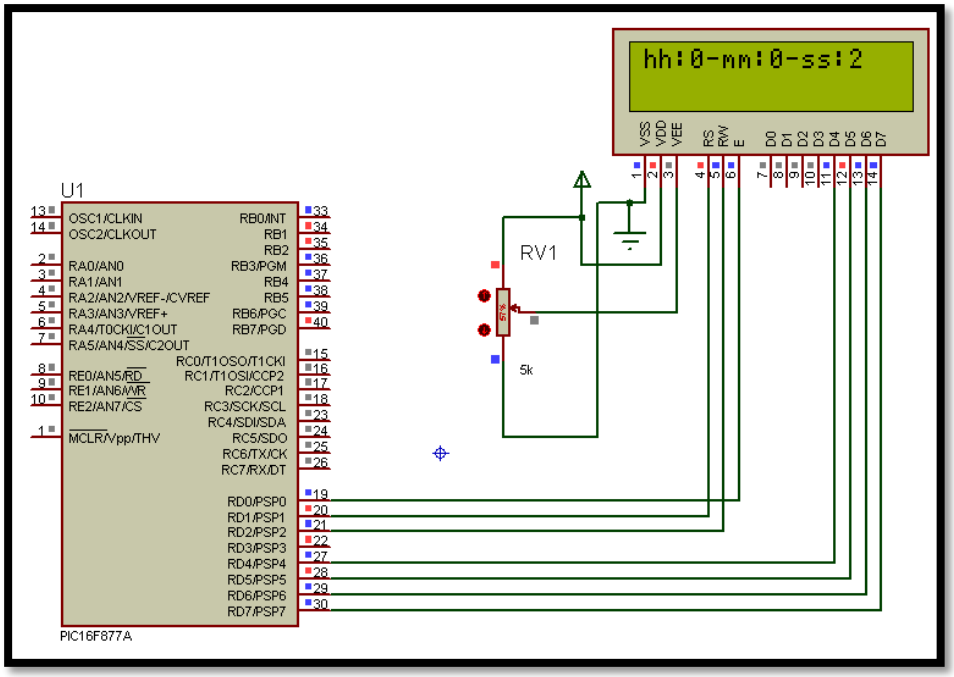
6.2.1.CONEXIÓN



6.2.2.PROGRAMA

```
#INT_TIMER1
void TMR1_isr( ){
   output_toggle(PIN_B0);
   i++;
   if( i>= 10){
      i = 0;
      ss++;
      if(ss>=60){        ss = 0;mm++;
         if(mm>=60){     mm = 0;hh++;
            if(hh>=24){ hh = 0;
            }
         }
      }
      lcd_putc('\f');
      printf(lcd_putc,"hh=%u-mm=%u-ss=%u",hh,mm,ss);
   }

   set_timer1( CARGA_TIMER1 );
}
void main ( ){

   setup_timer_1( T1_INTERNAL|T1_DIV_BY_8 );
   set_timer1( CARGA_TIMER1 );
   enable_interrupts( INT_TIMER1 );
   enable_interrupts(GLOBAL);
   set_tris_b(0);
   lcd_init();
   while (TRUE){
   }
}
```

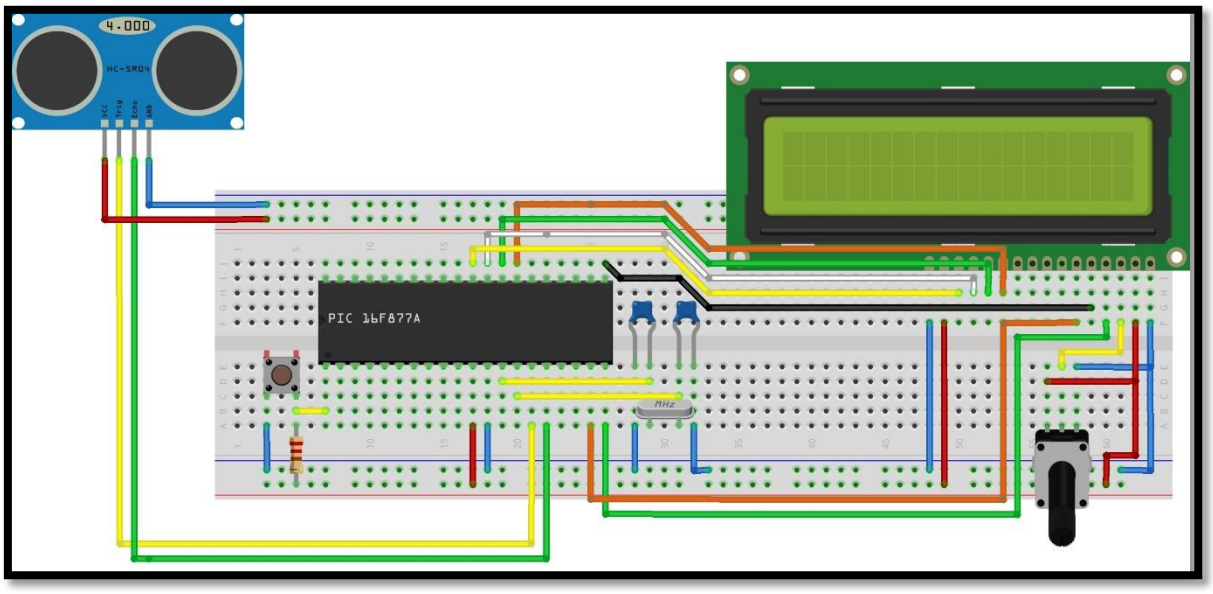
6.2.3.SIMULACION



6.3. Ejemplo3:

Realizar un medidor de distancia utilizando el sensor ultrasónico HC-SR04, el pin de TRIGGER va conectado al PIN_C0 y el pin de ECHO se conecta con el PIN_C1

6.3.1.CONEXIÓN



6.3.2.PROGRAMA

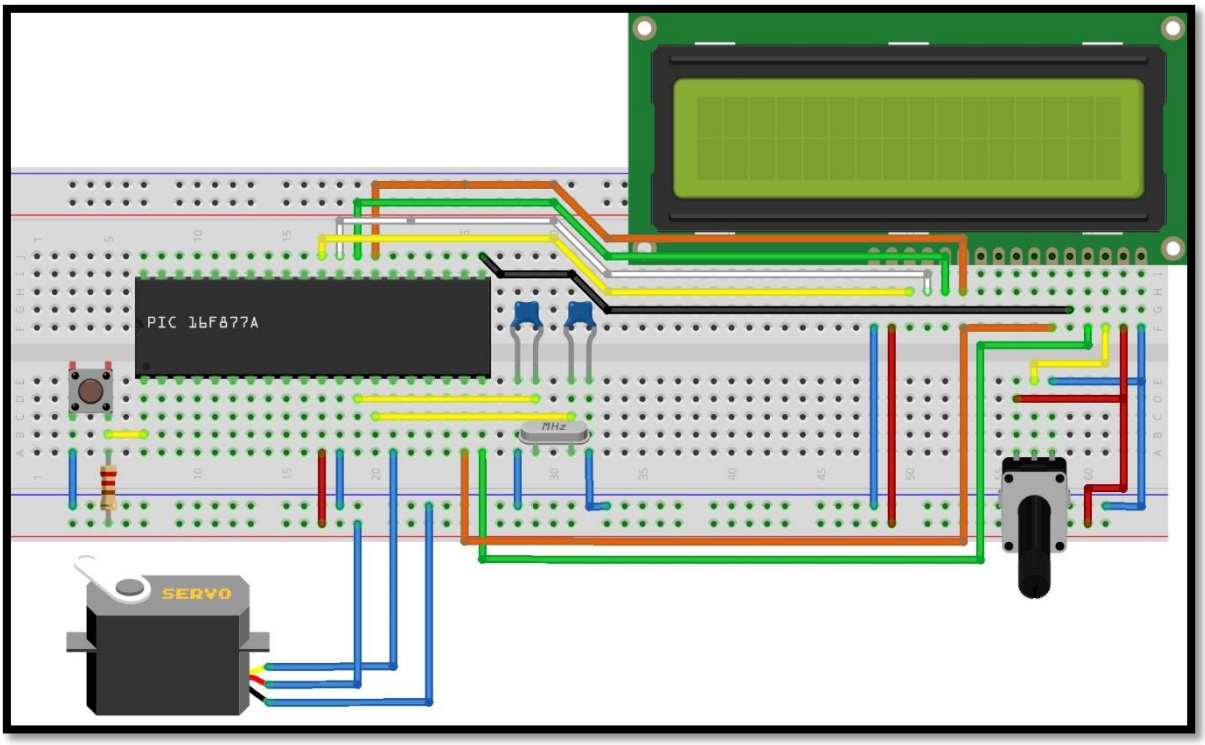
main.c

```
#include <16f877a.h>
#fuses    xt
#use      delay(clock=4M)
#use rs232(BAUD=9600,XMIT=PIN_C6,RCV=PIN_C7)
#include "ultrasonico.c"

void main(){

   unsigned int16 tiempo = 0;
   float distancia;

   init_ultrasonico();

   for(;;){
      tiempo = read_time_ultrasonico();
      distancia = calcular_distancia(tiempo);
      printf("tiempo = %Lu - distancia: %f\n",tiempo,distancia);
      delay_ms(300);
   }
}
```

6.4. Ejemplo4:

Controlar el giro en sentido horario y antihorario de un servomotor, el pin de control del servomotor va conectado al PIN_C0

6.4.1.CONEXIÓN



6.4.2.PROGRAMA

main.c

```
#include <16f877a.h>
#fuses    xt
#use      delay(clock=4M)
#include "servomotor.c"


void main ( ){
   init_servomotor();

   for(;;){
      for(i=0;i<=17;i++){
         delay_ms(500);
      }
   }
}
```