Universidad Nacional Autónoma de México

Facultad de Ingeniería

Semestre 2023-2

Laboratorio de Microcomputadoras

Proyecto 6:

Uso del protocolo I2C y la función PWM.

Profesor:

M.I. Ruben Anaya García

Alumnos:	Número de cuenta:
Barreiro Valdez Alejandro	317520888
Gil Márquez Arath Emiliano	317083875
Herrera Carrillo Cristhian	317094662
Zepeda Baeza Jessica	317520747

Grupo Teoría: 1

Fecha de realización: 12 de junio de 2023

Fecha de entrega: 25 de junio de 2023

Introducción.

El microcontrolador PIC16F877A es uno de los dispositivos más populares de la familia PIC de Microchip Technology debido a su versatilidad y amplias capacidades. Este microcontrolador cuenta con varios periféricos integrados, como el Inter-Integrated Circuit (I2C) y el Pulse Width Modulation (PWM), que permiten la comunicación con otros dispositivos y la generación de señales de salida con diferentes niveles de voltaje.

El protocolo I2C es un estándar de comunicación serial síncrona utilizado para interconectar microcontroladores con otros dispositivos, como sensores, memorias, pantallas LCD y otros microcontroladores. Con el protocolo I2C, es posible establecer comunicación entre un maestro y uno o varios esclavos, donde el maestro controla la transferencia de datos. El PIC16F877A cuenta con hardware dedicado para el soporte de I2C, lo que facilita su implementación. Para utilizar el I2C es necesario configurar los registros adecuados para establecer la velocidad de comunicación, seleccionar el modo maestro o esclavo, y manejar las interrupciones y los buffers de datos, una vez configurado, el microcontrolador puede enviar y recibir datos a través del bus I2C utilizando las funciones y procedimientos proporcionados por el compilador o el software de desarrollo.

Por otro lado, el PWM es una técnica utilizada para generar señales analógicas mediante la modulación del ancho de pulso de una señal digital. El microcontrolador cuenta con módulos PWM que permiten generar señales con diferentes frecuencias y ciclos de trabajo. Estas señales PWM son ampliamente utilizadas para controlar la velocidad de motores, regular la intensidad luminosa de LEDs o controlar la posición de servomotores.

Para utilizar el PWM, se deben configurar los registros correspondientes para establecer la frecuencia de la señal PWM, el ciclo de trabajo deseado y el modo de operación. Además, es necesario definir los pines de salida que se utilizarán para generar la señal PWM. Una vez configurado, el microcontrolador generará la señal PWM según los parámetros establecidos, lo que permite controlar dispositivos y sistemas que requieren señales analógicas.

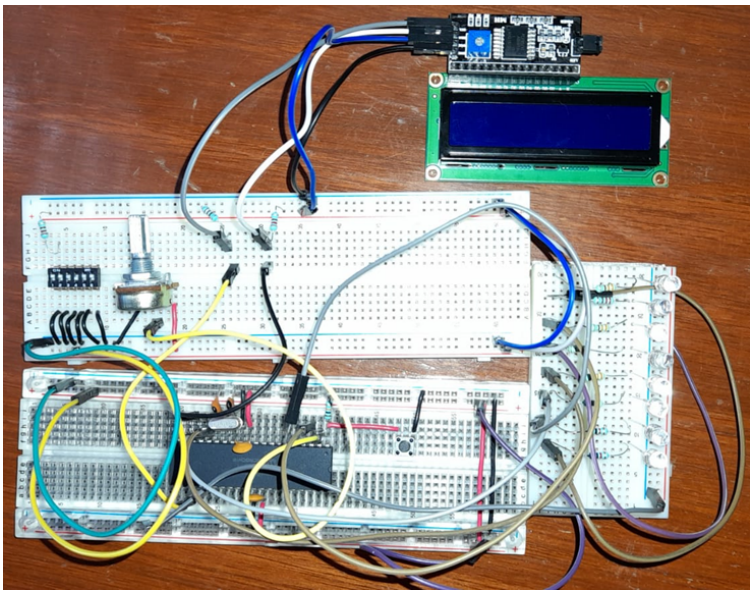
Desarrollo

El microcontrolador PIC16F877A es utilizado mediante el sistema mínimo. Gracias a sus puertos de entrada y salida es posible utilizarlo para interactuar con diferentes componentes, en este caso el LCD y los LEDs.

Para implementar la comunicación I2C, el circuito utiliza dos líneas: SDA y SCL, ya que estas líneas permiten la transferencia de datos entre el microcontrolador y el dispositivo I2C conectado, en este caso el LCD. El LCD es utilizado para mostrar información, se conecta al microcontrolador a través del bus I2C, esto significa que se utiliza el módulo I2C en el microcontrolador y se programa para enviar los comandos y datos al LCD.

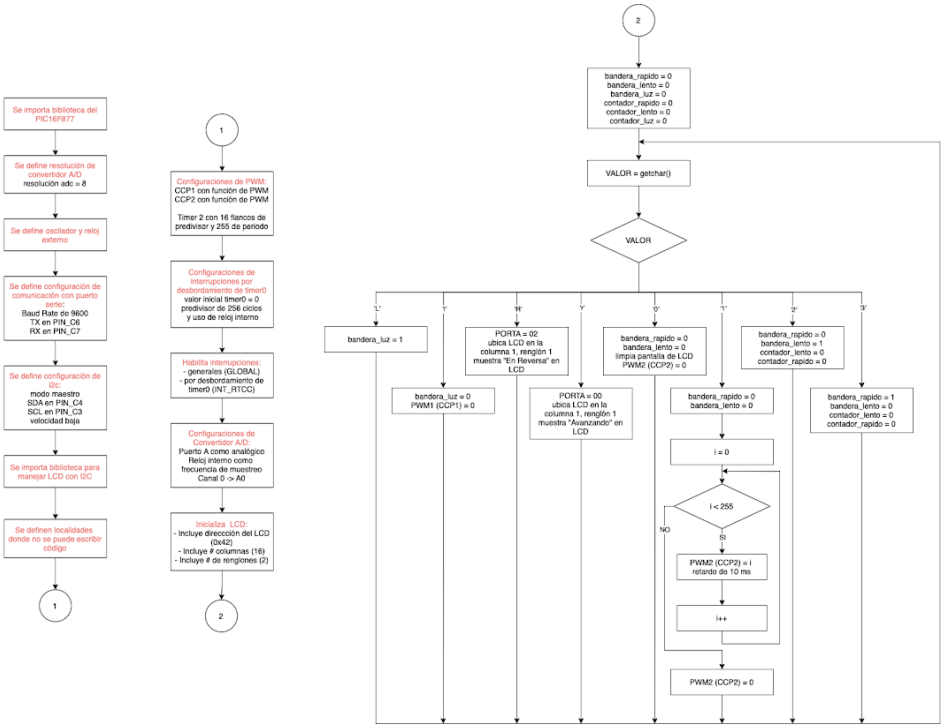
Para generar señales PWM y controlar los LEDs, el PIC16F877A tiene registros relacionados con el funcionamiento del PWM, estos registros permiten generar señales de ancho de pulso variable para controlar la intensidad de los LEDs. Para conectar los LEDs al microcontrolador se utilizaron pines de salida, cada pin de salida se conecta a un LED y la señal de PWM generada se activa por medio de la transmisión de un carácter por teclado, pudiendo apreciar la intensidad luminosa de los LEDs de diferente forma.

En la programación del microcontrolador, se utiliza un lenguaje de programación C para configurar los registros del microcontrolador, establecer la comunicación I2C con el LCD y generar las señales PWM para controlar los LEDs.



Algoritmos

Para este proyecto se realizó la siguiente carta ASM:



Configuraciones

En la primeras dos columnas se incluyen las bibliotecas, directivas y configuraciones iniciales que utiliza el programa como: las bibliotecas del PIC16F877 y la del control del LCD por medio de I2C, también se define una resolución del convertidor A/D de 8 bits junto con el reloj externo y el oscilador del microcontrolador.

Se configura la comunicación asíncrona por el puerto serie usando un Baud Rate de 9600 con TX en el pin C6 y RX en C7. Las configuraciones del I2C incluyen la definición del microcontrolador como maestro, SLC en el pin C3, SDA en el pin C4 y velocidad baja. Para el LCD se inicializa incluyendo su dirección, sus renglones y sus columnas.

La configuración del PWM involucró utilizar la función de PWM tanto del módulo CCP1 como del CCP2. Debido a que estos módulos utilizan el Timer 2 también se configura con un predivsor de 16 flancos y con 255 como valor del periodo. De manera que se definirá el tiempo de encendido con un valor de 1 a 255.

También se configuran interrupciones por desbordamiento de Timer 0 para provocar que cierto LED se prenda con PWM cada ciertos segundos. Por lo tanto se establece un valor inicial de 0 en Timer 0 y se configura con un predivisor de 256 ciclos utilizando el reloj interno (la interrupción sucederá cada 13.1 ms). Se habilitan las interrupciones globales y la particular de desbordamiento de Timer 0.

La última configuración es la del convertidor A/D donde se define al Puerto A como analógico, se utiliza el reloj interno como frecuencia de muestreo y se utiliza en canal 0 (A0).

Algoritmo

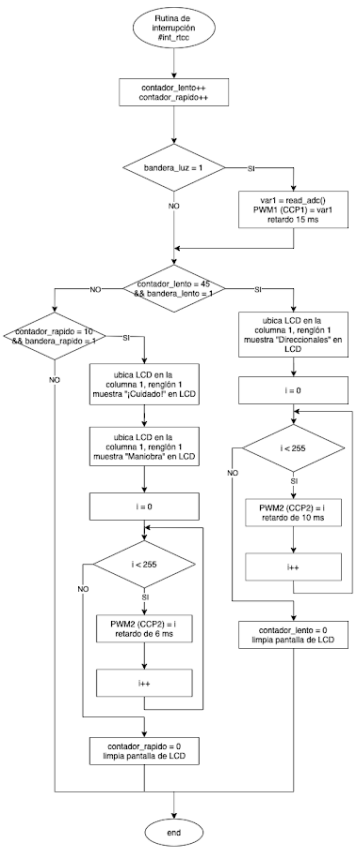
Como entrada se tiene un potenciómetro y como salidas se tienen 3 LEDS y un LCD.
El LED1 es el de Reversa y se enciende al ingresar una ‘R’ y se apaga al ingresar una ‘r’.
El LED2 es llamado Luz y se controla su intensidad por medio de PWM, el cual toma el valor obtenido del potenciómetro con el convertidor A/D.
El LED3 indica Direccionales o Intermitentes (dependiendo su velocidad) y se controla por medio de PWM donde se recorre un ciclo de trabajo desde 0% hasta 100%. Estos ciclos de encendido se realizan cada ciertos segundos utilizando interrupciones.
Se definieron 3 banderas y 2 contadores:

- Bandera_rapido: indica una velocidad alta para el LED3 (simula intermitentes).
- Bandera_lento: indica una velocidad baja para el LED3 (simula direccionales).
- Bandera_luz: indica si se debe obtener un valor analógico del potenciómetro.
- Contador_rapido: lleva la cuenta de las interrupciones para llegar a 0.13 segundos.
- Contador_lento: lleva la cuenta de las interrupciones para llegar a 0.6 segundos.

El programa inicializa todas las banderas y contadores es cero. Después tiene un loop infinito donde se recibe un caracter desde terminal, se guarda en una variable y se analiza en un switch para realizar una cierta acción:

- ‘L’: se prende bandera_luz.
- ‘l’: se apaga bandera_luz y se apaga el LED2, haciendo cero su ciclo de trabajo.
- ‘R’: se prende el LED1 y se muestra “En Reversa” en el LCD.
- ‘r’: se apaga el LED1 y se muestra “Avanzando” en el LCD.
- ‘0’ (Apagado): se apagan las banderas de rápido y lento, se limpia el LCD y se apaga el LED3 haciendo 0 su ciclo de trabajo.
- ‘1’ (Modo no periódico): se apagan las banderas de rápido y lento ya que sólo se prendera el LED3 una vez. Después con un ciclo *for* que va de 0 a 255, se modifica el ciclo de trabajo del LED3 en cada iteración junto con un retardo de 10 ms. De manera que se observa cómo se va haciendo más intensa la luz del LED3 por 2.5 segundos. Al final, se apaga el LED3 haciendo 0 su ciclo de trabajo.
- ‘2’ (Modo lento): Se apaga la bandera_rapido y se prende la bandera_lento. Además se inicializan ambos contadores en 0 para que a partir de ese momento empiecen a contar los segundos.
- ‘3’ (Modo rápido): Se apaga la bandera_lento y se prende la bandera_rapido. También se inicializan ambos contadores en 0.

Después de realizar las asignaciones, se vuelve a recibir un caracter de terminal.



Por último, se tiene la rutina de interrupción donde se hacen uso de todas las banderas antes mencionadas.

Una vez que se genera la interrupción se incrementan los contadores rápido y lento.

Después se verifica si la bandera_luz está prendida; si lo está se hace la conversión del potenciómetro y se le asigna el valor obtenido al ciclo de trabajo del LED2. Debido a que bandera_luz no tiene un contador asociado, se va a actualizar su valor cada 13 ms en caso de que su bandera siga prendida. Si no está prendida no se hace nada.

Después se verifica si contador_lento es 45 (ya pasaron 0.6 segundos) y si bandera_lento es 1. En ese caso, se muestra “Direccionales” en el LCD y se realiza un ciclo *for* para aumentar el ciclo de trabajo del LED3 en cada iteración junto con un retardo de 10 ms. Cuando termina el ciclo *for* se apaga el LED3 haciendo 0 su ciclo de trabajo y se reinicia con 0 contador_lento. De manera que una vez que acabe el LED3 de encenderse se volverá a encender después de 0.6 segundos siempre y cuando su bandera siga encendida.

Si la condición anterior no se cumple, se verifica si contador_rapido es 10 y bandera_rapido es 1. En ese caso se realiza lo mismo que el caso anterior con la diferencia de que el tiempo que se tendrá entre cada encendido será de 0.13 ms y el retardo entre cada iteración será de 6 ms, lo que se verá como un encendido más rápido. También se muestra “¡Cuidado! Maniobra” en el LCD. Cuando termina el ciclo *for* se apaga el LED3 haciendo 0 su ciclo de trabajo y se reinicia con 0 contador_lento. De manera que una vez que acabe el LED3 de encenderse se volverá a encender después de 0.13 segundos siempre y cuando su bandera siga encendida.

Programa comentado

```
#include <16f877.h>                          //Biblioteca para PIC16F877
#fuses HS,NOPROTECT,
#device ADC = 8                              //8 bits de lectura de entrada analógica
#use delay(clock=20000000)                   //Configurando el reloj
#use rs232(baud=9600, xmit=PIN_C6, rcv=PIN_C7)   //Configurando comunicación serie asíncrona
```

```
#use i2c(MASTER, SDA=PIN_C4, SCL=PIN_C3, SLOW, NOFORCE_SW)
/*
Directiva para protocolo I2C, se configura:
-Modo maestro
-SDA en Pin C4
-CSL en Pin C3
-Velocidad baja
*/
#include <i2c_LCD.c>                                          //Biblioteca para LCD con I2c
#org 0x1F00,0x1FFF void loader16F877(void){}

int var1,i;
/*
var1: valor leído de entrada analógica
i:    contador en fors
*/
char valor;                                                   //Valor de control
int bandera_rapido,bandera_lento,bandera_luz;
/*
bandera_rapido:   Bandera de activación intermitentes.
bandera_lento:    Bandera de activación direccionales.
bandera_luz:      Bandera de activación luces coche.
*/
int contador_lento,contador_rapido;
/*
contador_rapido:   Contador de tiempo para intermitentes.
contador_lento:    Contador de tiempo para direccionales.
*/

#int_rtcc                                                     //Desbordamiento del TIMER0
clock_isr(){
  contador_lento++;                                           //Incrementa contador direccionales
  contador_rapido++;                                          //Incrementa contador intermitentes

  if(bandera_luz == 1){                                       //Si se activa bandera de luz
    var1 = read_adc();                                        //Se lee el valor del convertidor
    set_pwm1_duty(var1);                                      //El PWM de CCP1 es el valor del convertidor
    delay_ms(15);                                             //Retardo de 15ms
  }

  if(contador_lento == 45 && bandera_lento == 1){             //Si se activa la bandera de direccionales y se llega al contador
    lcd_gotoxy(1,1);
    printf(lcd_putc,"Direccionales");                         //Indicar en LCD que son direccionales
    for(i  = 0;i < 255; i += 5){                              //De 0 a 255
      set_pwm2_duty(i);                                       //Cambiar el PWM de CCP2 de cinco en cinco
      delay_ms(10);                                           //Cada 10ms
     }
    contador_lento = 0;                                       //Reinicia el contador de direccionales
    lcd_clear();                                              //Se limpia LCD
  }

  if(contador_rapido == 10 && bandera_rapido == 1){           //Si se activa la bandera de intermitentes y se cumple el contador
    lcd_gotoxy(1,1);
    printf(lcd_putc,"¡Cuidado!");
    lcd_gotoxy(1,2);
    printf(lcd_putc,"Maniobra");                              //Indicar en LCD que se está en una maniobra
    for(i  = 0;i < 255; i += 5){                              //De 0 a 255
      set_pwm2_duty(i);                                       //Cambiar el PWM de CCP2 de cinco en cinco
```

```
      delay_ms(6);                                          //Cada 6ms
    }
    contador_rapido = 0;                                    //Reinicia el contador de intermitentes
    lcd_clear();                                            //Se limpia LCD
  }
}

void main(){
  //Configuración de pwm
  setup_ccp1(CCP_PWM);                                      //Habilita CCP1 como PWM
  setup_ccp2(CCP_PWM);                                      //Habilita CCP2 como PWM
  setup_timer_2(T2_DIV_BY_16, 255, 1);                      //Configura el TIMER2 con predivisor, periodo y postescalador

  //Configuración de interrupciones
  set_timer0(0);                                            //Inicia Timer en 0
  setup_counters(RTCC_INTERNAL,RTCC_DIV_256);               //Fuente de reloj y pre-divisor
  enable_interrupts(INT_RTCC);                              //Habilita interrupción de Timer0
  enable_interrupts(GLOBAL);                                //Habilita interrupciones globales

  //Configuracion de convertidor A/D
  setup_adc(ADC_CLOCK_INTERNAL);                            //Configura el reloj del convertidor
  setup_adc_ports(AN0);                                     //Se configura AN0 como analógico
  set_adc_channel(0);                                       //Se utiliza el primer canal como el convertidor
  delay_ms(100);                                            //Retardo de 100ms

  lcd_init(0x4E, 16, 2);                                     //Inicializa LCD en 4E

  //Inicializa todos las banderas y contadores en 0
  bandera_rapido = 0;
  bandera_lento = 0;
  bandera_luz = 0;
  contador_lento = 0;
  contador_rapido = 0;
  i = 0;

  while(1){
    valor = getchar();                                      //Leer valor de comunicación en serie asíncrona
    switch(valor){                                          //Switch con el valor introducido
      case 'L':                                             //L -> Prende los faros del coche
        bandera_luz = 1;                                    //Activa la bandera de luz
        break;
      case 'R':                                             //R -> Prende faros de reversa
        output_a(0x02);                                    //Salida de faros en A1
        lcd_gotoxy(1,1);
        printf(lcd_putc,"En Reversa");                      //Se indica en LCD que se está en reversa
        break;
      case 'l':                                             //l -> Apaga los faros del coche
        bandera_luz = 0;                                    //Desactiva la bandera de luz
        set_pwm1_duty(0);                                   //El valor de PWM de CCP1 es 0
        break;
      case 'r':                                             //r -> Apaga faros de reversa
        output_a(0x00);                                     //Apaga A1
        lcd_gotoxy(1,1);
        printf(lcd_putc,"Avanzando");                      //Se indica en LCD que se está avanzando
        break;
      case '0':                                             //0 -> Apagar luces de intermitentes y direccionales
        bandera_rapido = 0;                                 //Se apaga bandera intermitentes
        bandera_lento = 0;                                 //Se apaga bandera direccionales
```

```
        lcd_clear();                        //Se limpia el LCD
        set_pwm2_duty(0);                   //El valor de PWM de CPP2 es 0
        break;
      case '1':                             //1 -> Solo prender faros una vez
        bandera_rapido = 0;                 //Se apaga bandera de intermitentes
        bandera_lento = 0;                  //Se apaga bandera de direccionales
        for(i  = 0;i < 255; i += 5){        //De 0 a 255
          set_pwm2_duty(i);                 //Cambiar PWM de CCP2 de cinco en cinco
          delay_ms(10);                     //Cada 10ms
        }
        set_pwm2_duty(0);                   //El valor de PWM de CCP2 es 0
        break;
      case '2':                             //2 -> Prender direccionales
        bandera_rapido = 0;                 //Se apaga bandera intermitentes
        bandera_lento = 1;                  //Se prende bandera direccionales
        contador_lento = 0;                 //Reinicia el contador direccionales
        contador_rapido = 0;                //Reinicia el contador intermitentes
        break;
      case '3':                             //3 -> Prender intermitentes
        bandera_rapido = 1;                 //Se prende bandera intermitentes
        bandera_lento = 0;                  //Se apaga bandera direccionales
        contador_lento = 0;                 //Reinicia el contador direccionales
        contador_rapido = 0;                //Reinicia el contador intermitentes
        break;
      default:
        printf("\nVALOR NO VÁLIDO\n");     //Imprimir valor no válido
    };
  }
}
```

Conclusiones y/o comentarios.

Barreiro Valdez Alejandro:

En este proyecto se utilizó el protocolo I2C y la función de PWM en los puertos de CCP1 y CCP2. I2C es un protocolo que permite la comunicación entre un dispositivo maestro y varios esclavos a partir de dos líneas de comunicación: SDA y SCL. Estas líneas permiten lo que se conoce como la comunicación en serie síncrona, ya que para este protocolo se utiliza una línea de reloj para la comunicación. En este proyecto se desarrolló el uso del protocolo I2C utilizando un circuito PCF8574. Este circuito facilita la comunicación utilizando I2C entre un maestro y un Display de Cristal Líquido. Para el proyecto se configuró como maestro a la PIC y esclavo a LCD, después, utilizando la dirección del esclavo se llevó a cabo el algoritmo que se utiliza en I2C para enviar datos. Además, se programó el PWM que representa el ciclo de trabajo que tiene una señal que genera la PIC utilizando el TIMER2. Esta señal fue utilizada para controlar la intensidad de LEDs que pudieran representar los

faros de un coche. Gracias a estos componentes se pudo entender dos componentes básicos y esenciales de las microcomputadoras.

Gil Márquez Arath Emiliano:

La interfaz de LCD basada en I2C simplifica la conexión y comunicación con el microcontrolador, lo que facilita la visualización de información y la interacción con el usuario. El uso del protocolo I2C también permite la conexión de múltiples dispositivos utilizando solo dos cables, lo que contribuye a una implementación más limpia y ordenada del circuito. Al igual que el uso del PWM nos permitió generar señales de salida de ancho de pulso modulado, que son ideales para controlar la intensidad luminosa de los LEDs. La combinación de estos dispositivos y su uso es una solución potente y flexible para proyectos electrónicos que requieren visualización y control de componentes

Herrera Carrillo Cristhian:

La implementación del circuito con el microcontrolador para controlar un LCD mediante la comunicación I2C y generar señales PWM para controlar LEDs proporciona una solución versátil y eficiente para la visualización de información y el control de la intensidad de luz. Esta combinación de funcionalidades brinda flexibilidad y amplias posibilidades de aplicación en diversos proyectos electrónicos.

Zepeda Baeza Jessica:

Para este proyecto se tuvo la idea de implementar PWM y la comunicación I2C de una manera que contribuyera al proyecto final. Por lo tanto se utilizaron LEDs que hacen uso de PWM como direccionales, intermitentes y como luces que cambian su intensidad según un valor analógico. La comunicación I2C se observa al hacer que el LCD muestre en palabras las distintas acciones que se realizan. Además para hacer funcionales y coherentes las luces fue necesario hacer uso de funciones del microcontrolador utilizadas anteriormente como interrupciones, uso de Timers, comunicación asíncrona por el puerto serie y más. Finalmente, se pudieron observar los registros, puertos y Timers involucrados tanto en la configuración del PWM como del convertidor A/D y el I2C.