UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO FACULTAD DE INGENIERÍA
DIVISIÓN DE INGENIERÍA ELÉCTRICA
DEPARTAMENTO DE INGENIERÍA EN COMPUTACIÓN

M.I. Rubén Anaya García

LABORATORIO DE MICROCOMPUTADORAS Gpo 7

Práctica #3

Jose Francisco Ugalde Vivo	Gpo teoría 5
Hector Montoya Perez	Gpo teoría 5
Lara Sala Kevin Arturo	Gpo teoría 5

2021-2

Objetivo.

Desarrollar la habilidad de interpretación de esquemáticos.
Conocer el diagrama del sistema mínimo del microcontrolador, el software de comunicación.
Realizar aplicaciones con puertos paralelos en la modalidad de salida y la ejecución de un programa en tiempo real.

Introducción.

Para ejecutar un programa en el procesador, se debe alambrar el sistema mínimo, siguiendo el siguiente diagrama.

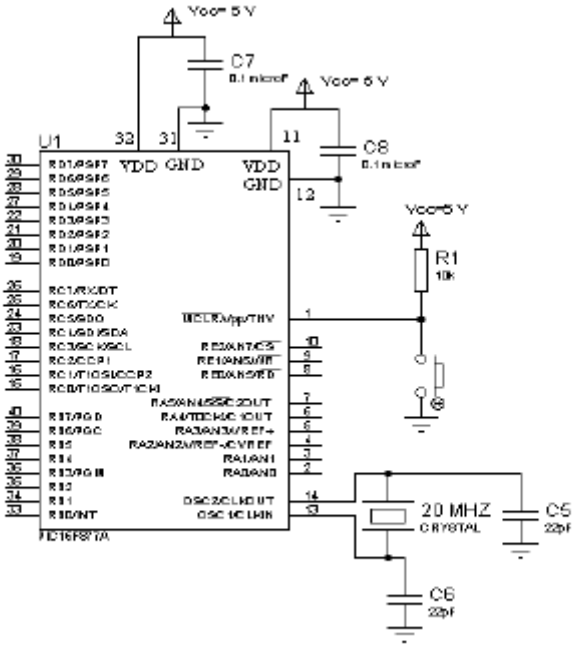


Figura 3.1 Sistema mínimo del microcontrolador PIC16F877(A)

Como se puede apreciar, el sistema requiere de tres módulos imprescindibles.

- Reloj.
  Formado por un cristal de cuarzo de 20[MHz] y dos capacitores de 22[pF], cuyo objetivo es la generación de la frecuencia de operación extrema.

- Circuito de reset.
  Formado por una resistencia y un push button; cuya finalidad es la generación del pulso para producir un reset en el sistema.

- Alimentación al sistema.
  Vdd = 5[V] y GND = 0[V].

Con la finalidad de no depender de la existencia del programador externo y tener la ventaja de tener un programador en circuito se debe agregar:

A. Circuito que permita la comunicación serie asíncrona.

B. Programar con antelación el bootloader al procesador.

El circuito queda de la siguiente manera:

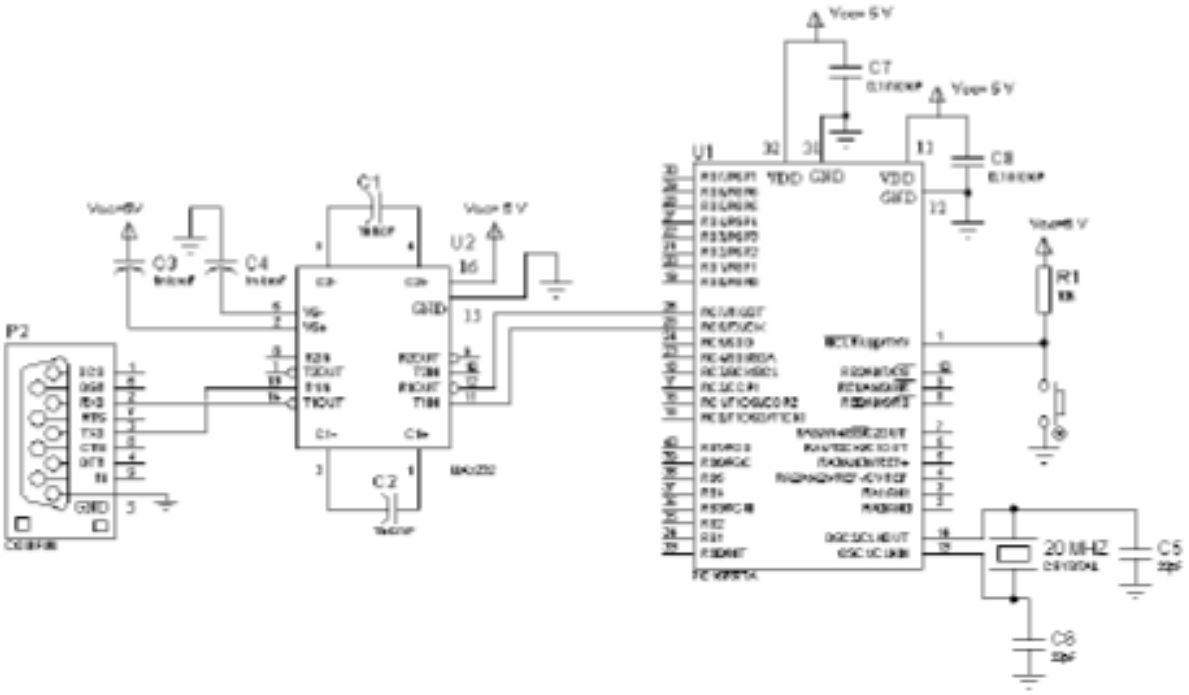


Figura 3.2 Sistema mínimo con comunicación serie.

Es importante mencionar que será necesario en caso de no disponer de un puerto serie asíncrono en su computadora, el uso del convertidor serie a USB.

Figura 3.3 Cable convertidor USB-Serie

Puertos Paralelos

El microcontrolador PIC tiene 5 puertos paralelos, denominados A, B, C, D y E, todos ellos se pueden configurar para operar como puerto de salida o entrada.

Puerto	Tamaño	Función	TRISX	PORTX
A	6	E/S	85H	05H
B	8	E/S	86H	06H
C	8	E/S	87H	07H
D	8	E/S	88H	08H
E	3	E/S	89H	09H

Al emplear un puerto paralelo, lo primeo que se debe de hacer es configurar su función, esto se realiza en las posiciones de memoria RAM denominados **TRISX** los cuales están ubicados en el banco número 1. Una vez ubicado en este banco se realiza la configuración, bajo la siguiente convención.

´0´	Configura el bit del puerto como salida
´1´	Configura el bit del puerto como entrada

Después que se ha configura todo el puerto, regresar al banco cero para enviar o recibir información a través de los registros de datos **PORTX**; a continuación se presenta las instrucciones que realizan lo anterior:

```
        processor 16f877          ; Indica la versión de procesador
        include <p16f877.inc>     ; Incluye la librería de la versión del procesador

        org 0H                    ; Carga al vector de RESET la dirección de inicio
        goto inicio

        org 05H                   ; Dirección de inicio del programa del usuario
inicio: BSF STATUS,RP0            ; Cambia la banco 1
        BCF STATUS,RP1
        MOVLW B'00000000'         ; Configura al puerto B como salida (8 bits)
        MOVWF TRISB
        BCF STATUS,RP0            ; Regresa al banco cero
        ……
        ……
        end                       ; Directiva de fin de programa
```

Desarrollo práctica 3.

Para cada uno de los siguientes ejercicios, realizar los programas solicitados y comprobar el funcionamiento de ellos.

El sistema utilizado para esta práctica está diseñado de acuerdo al siguiente diagrama.

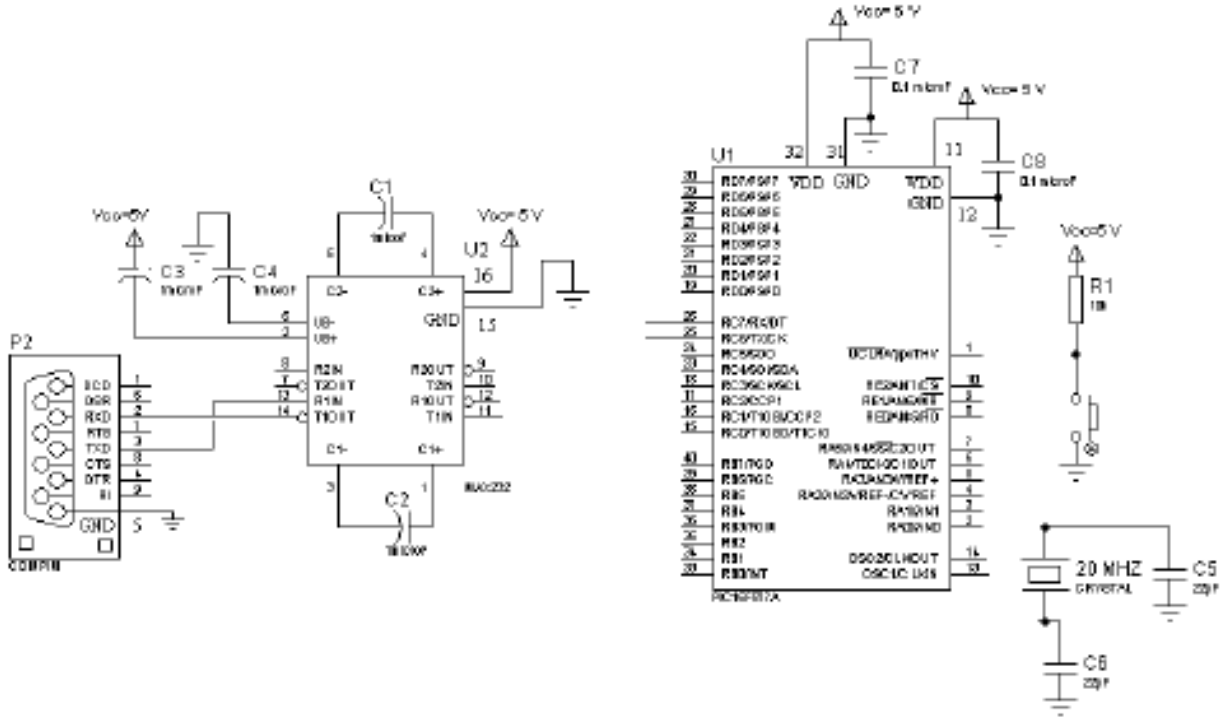


Figura 3.5 Circuito de prueba

1. **Revisar a detalle y en concordancia con el circuito 3.2, identificar las conexiones faltantes, discutir con sus compañeros y con su profesor el impacto y función de los mismos.**

Se revisó el diagrama, y más específicamente, las conexiones de alimentación del procesador, la cual, polariza todo el Circuito Integrado para su correcto funcionamiento.

De igual manera, se revisó con el profesor, la parte del reloj del sistema, que nos da una frecuencia de 20 [MHz], el cual nos va a dar los ciclos para llevar a cabo las funciones que se implementen en el microcontrolador.

2. **Completar las conexiones faltantes, utilizando jumpers; cerciorar el alambrado correcto.**

3. **Una vez resueltas las actividades anteriores, identificar la terminal PB0 del puerto B, realizar la conexión con la salida de una resistencia y un led.**

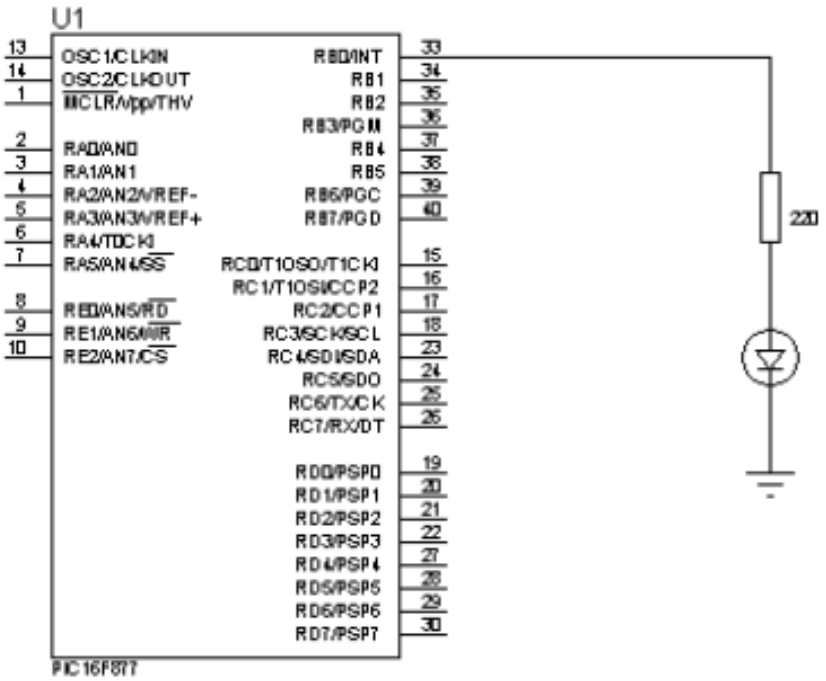


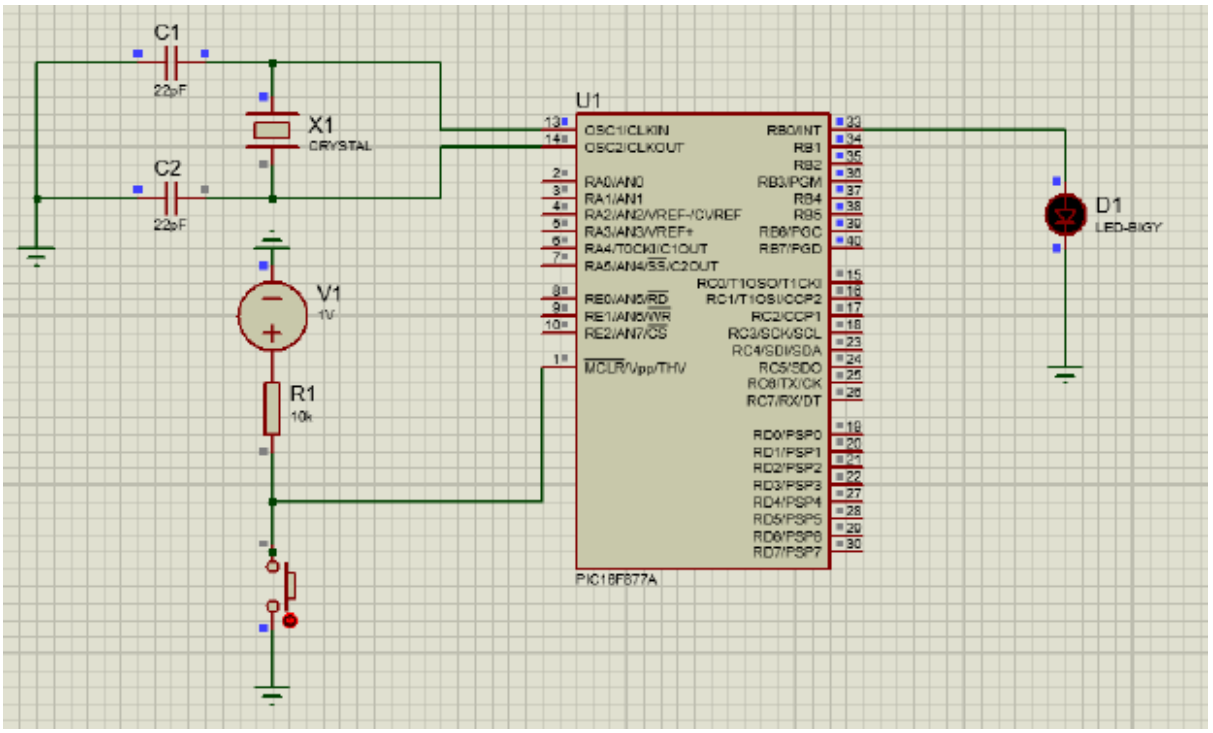
Figura 3.6 Circuito PB0

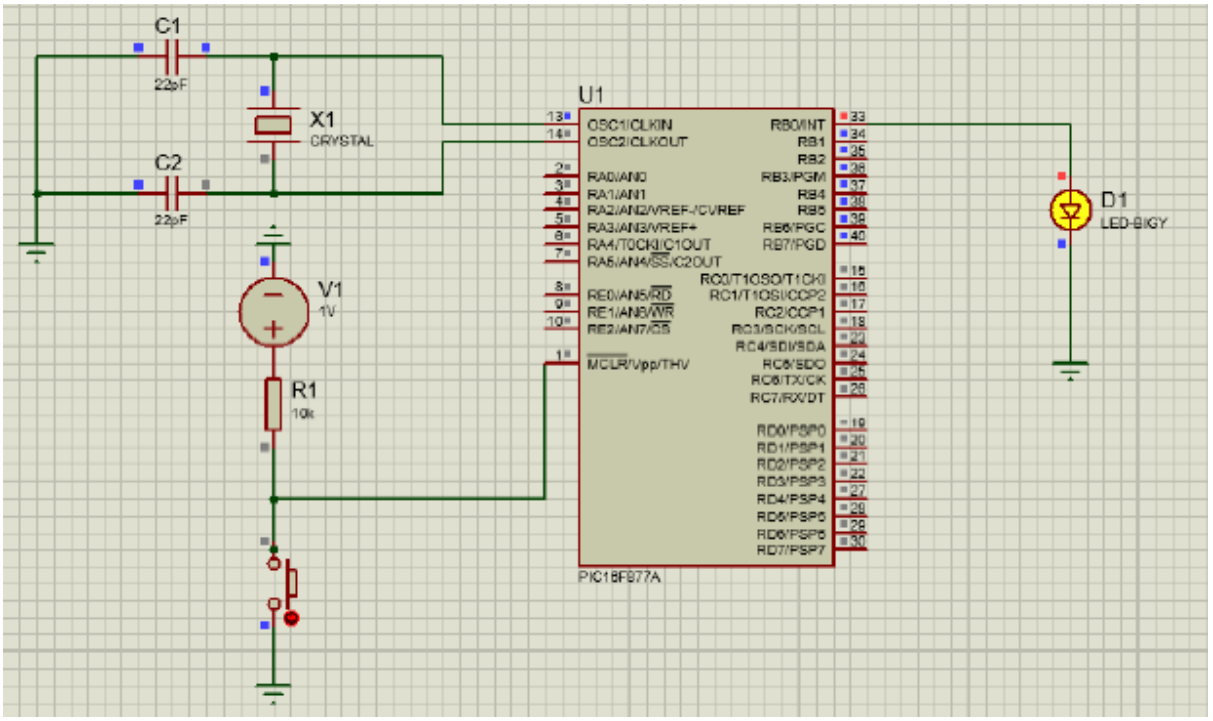
4. Escribir, comentar e indicar qué hace el siguiente programa.

D:\...\E1.asm*

```
processor 16f877
include <p16f877.inc>

valor1  EQU H'21'           ;VALOR1 = 21 hex
valor2  EQU H'22'           ;VALOR1 = 22 hex
valor3  EQU H'22'           ;VALOR1 = 23 hex

cte1    EQU 20h             ;cte1 = 20 Hex
cte2    EQU 20h             ;cte1 = 50 Hex
cte3    EQU 20h             ;cte1 = 60 Hex

        ORG 0
        GOTO INICIO

INICIO:
        BSF STATUS, RP0     ;RP0 = '1'.
        BCF STATUS, RP1     ;RP1 = '0'.
        MOVLW   H'0'        ;W = 00H.
        MOVWF   TRISB       ;TRISB = 00H.
        BCF STATUS, RP0     ;RP0 = '0'.
        CLRF    PORTB       ;Limpia el port B.

LOOP2:
        BSF PORTB, 0        ;Asigna un 0 al puerto B.
        CALL RETARDO        ;Llama a la subrutina de retardo.
        BCF PORTB, 0        ;Limpia el puerto B.
        CALL RETARDO        ;Lama a la función retardo.
        GOTO LOOP2          ;Llama a subrutina LOOP2.

RETARDO:
        MOVLW   cte1        ;Asigna W=20H.
        MOVWF   valor1      ;Mueve W a valor1;

TRES:
        MOVLW   cte2        ;Asigna 50H a W.
        MOVWF   valor2      ;Mueve W a valor2.
DOS:
        MOVLW   cte3        ;Asigna 60H a W.
        MOVWF   valor3      ;Mueve W a valor 3.
UNO:
        DECFSZ  valor3      ;Decrementa valor3 en 1.
        GOTO    UNO         ;Si el resultado es diferente de 0, ir a uno.
        DECFSZ  valor2      ;Decrementa valor2 en 1.
        GOTO    DOS         ;Si el resultado es diferente de 0, ir a dos.
        DECF    valor1      ;Decrementa valor1 en 1.
        GOTO    TRES        ;Si el resultado es diferente de 0, ir a tres.

        RETURN
        END
```

5. **Ensamblar y cargar el programa anterior en el microcontrolador; ¿qué es lo que se puede visualizar?**





6. **En el programa, modifique el valor de cte1 a 8H, ensamblar y programar, ¿qué sucede?**

Al cambiar el valor de cte1, el tiempo de encendido y apagado en el LED es más rápido. Esto debido a que afecta directamente a la subrutina y el número de comparaciones que debe hacer el programa.

7. **Modifique cte1 a 80H y ensamblar. ¿Sucedió algún cambio?**

Ahora, el tiempo de encendido y apagado aumentó considerablemente, ya que al igual que en la pregunta anterior, el número de comparaciones se modifica.

8. **Modificar el programa anterior para que ahora se actualice el contenido de todos los bits del puerto B, y se genera una rutina de retardo de un segundo. Este programa requiere 8 salidas conectadas al puerto B, tal como se muestra en la figura.**

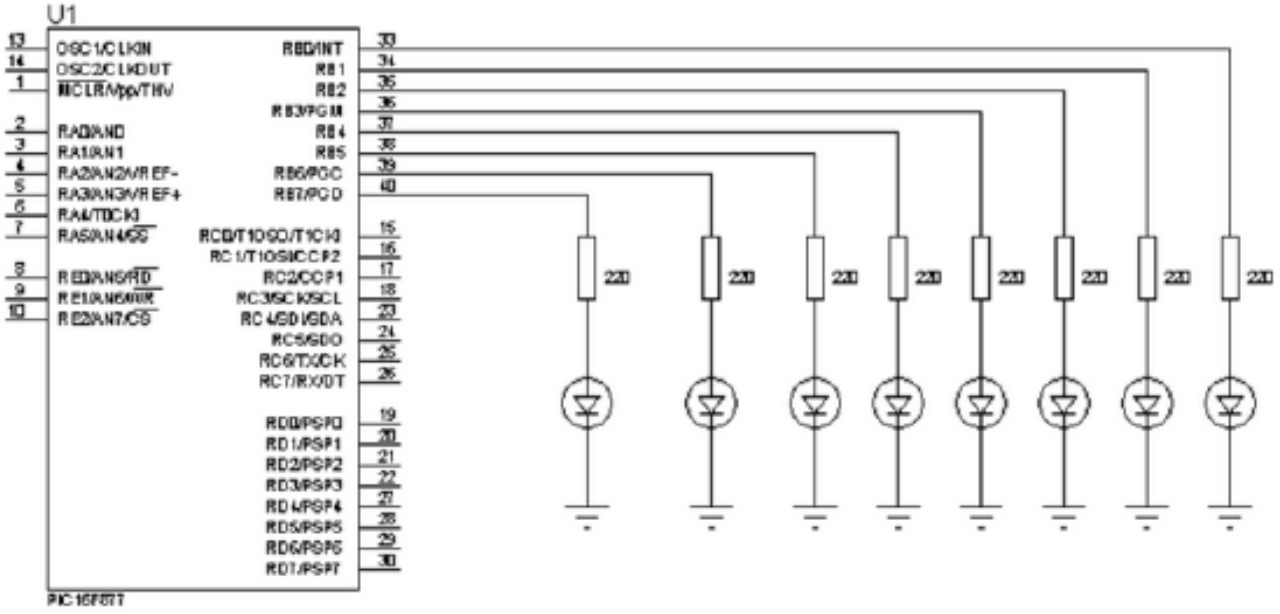


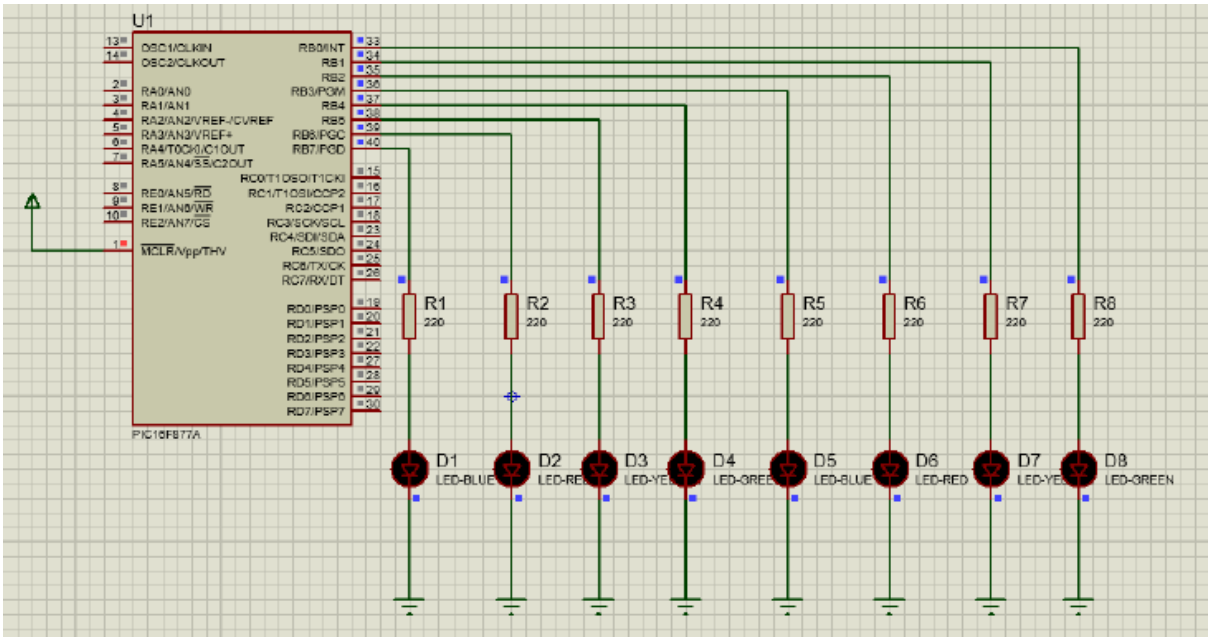
Figura 3.7 Conexión del sistema mínimo al módulo de 8 leds

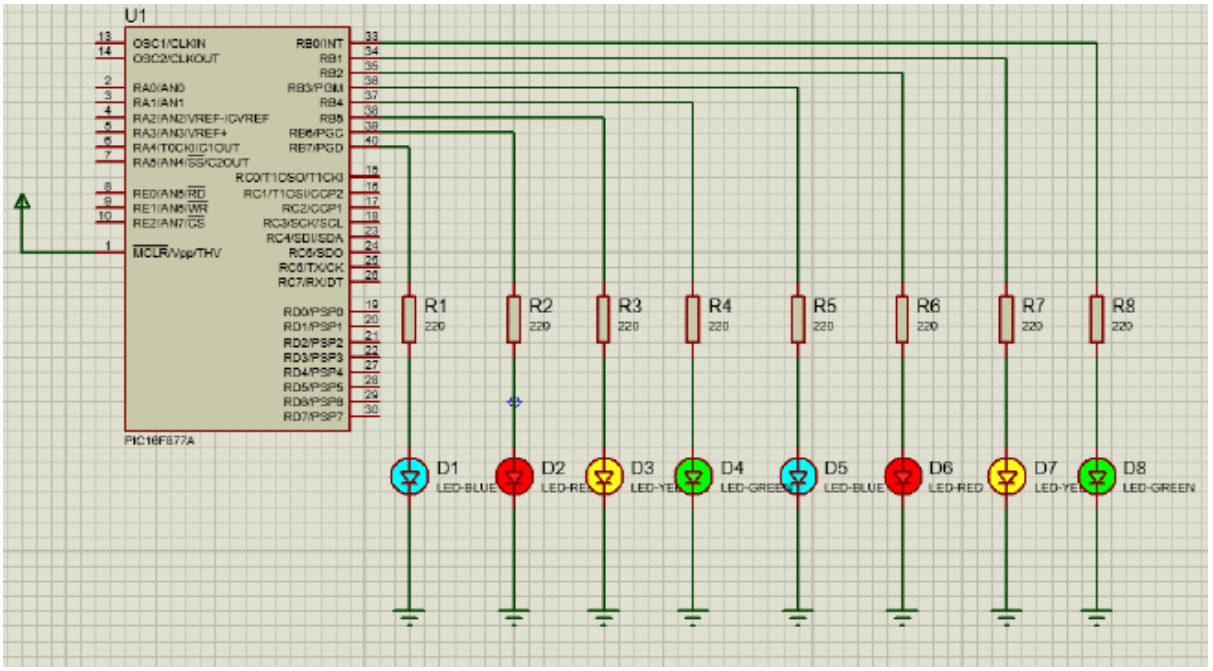
Código:

D:\...\E1.asm*

```
processor 16f877
include <p16f877.inc>

valor1  EQU H'21'           ;VALOR1 = 21 hex
valor2  EQU H'22'           ;VALOR1 = 22 hex
valor3  EQU H'22'           ;VALOR1 = 23 hex

ctel    EQU 20h             ;ctel = 20 Hex
cte2    EQU 20h             ;ctel = 50 Hex
cte3    EQU 20h             ;ctel = 60 Hex

        ORG 0
        GOTO INICIO
        ORG 5
INICIO:
        BSF STATUS, RP0     ;RP0 = '1'.
        BCF STATUS, RP1     ;RP1 = '0'.
        MOVLW   H'0'        ;W = 00H.
        MOVWF   TRISB       ;TRISB = 00H.
        BCF STATUS, RP0     ;RP0 = '0'.
        CLRF    PORTB       ;Limpia el port B.

LOOP2:
        MOVLW OxFF          ;Asigna FF al puerto B.
        MOVWF PORTB         ;W = PortB.
        CALL RETARDO        ;Llama a la subrutina de retardo.
        CLRF PORTB          ;Limpia el puerto B.
        CALL RETARDO        ;Llama a la subrutina.
        GOTO LOOP2.

RETARDO:
        MOVLW   ctel        ;Asigna W=20H.
        MOVWF   valor1      ;Mueve W a valor1;

TRES:
        MOVLW   cte2        ;Asigna 50H a W.
        MOVWF   valor2      ;Mueve W a valor2.
DOS:
        MOVLW   cte3        ;Asigna 60H a W.
        MOVWF   valor3      ;Mueve W a valor 3.
UNO:
        DECFSZ  valor3      ;Decrementa valor3 en 1.
        GOTO    UNO         ;Si el resultado es diferente de 0, ir a uno.
        DECFSZ  valor2      ;Decrementa valor2 en 1.
        GOTO    DOS         ;Si el resultado es diferente de 0, ir a dos.
        DECF    valor1      ;Decrementa valor1 en 1.
        GOTO    TRES        ;Si el resultado es diferente de 0, ir a tres.

        RETURN
        END
```

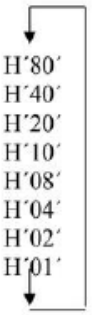
Simulación:





9. **Realizar un programa que muestre la siguiente secuencia en el puerto B, con retardos de ½ segundo.**

Secuencia:

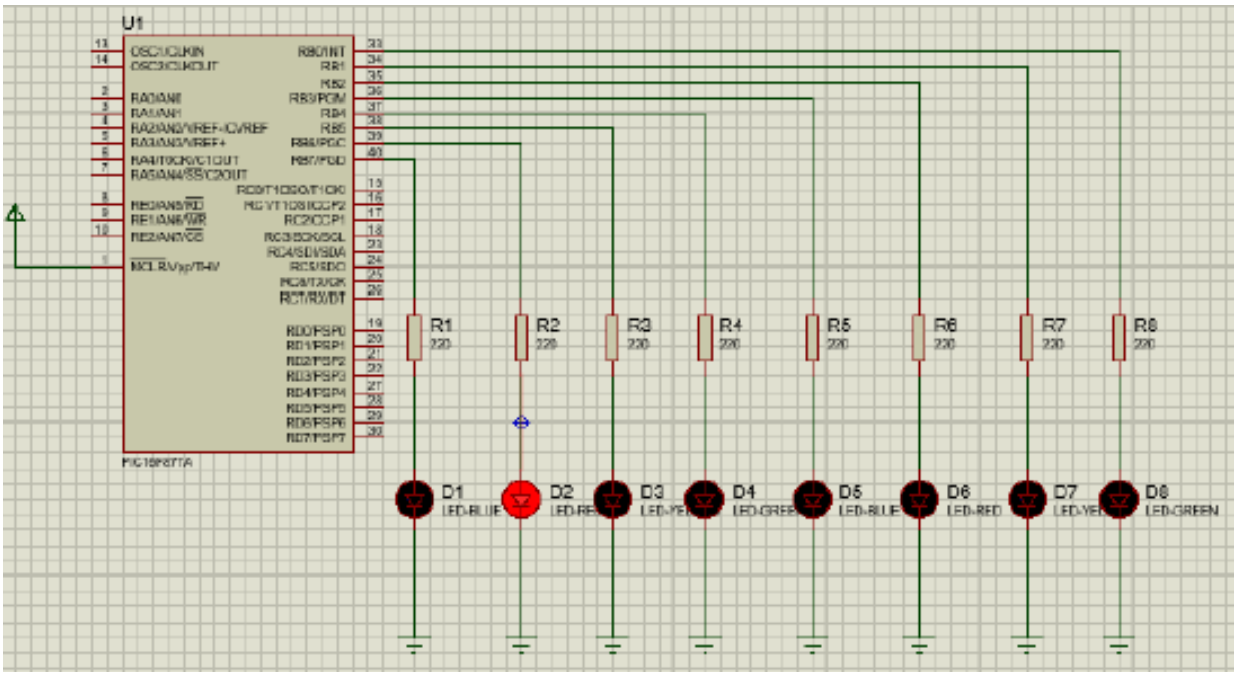


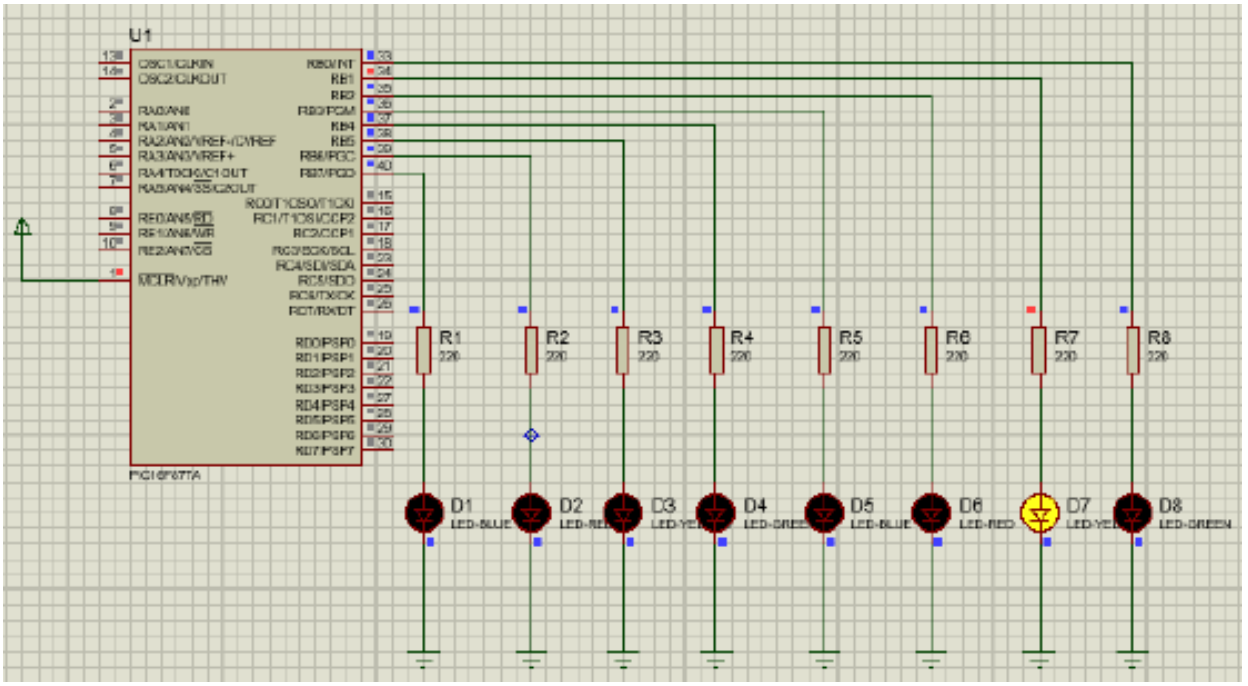
El circuito empleado es el mismo que en el ejercicio anterior.

D:\...\E1.asm*

```
processor 16f877
include <p16f877.inc>

valor1  EQU H'21'           ;VALOR1 = 21 hex
valor2  EQU H'22'           ;VALOR1 = 22 hex
valor3  EQU H'22'           ;VALOR1 = 23 hex

cte1    EQU 20h             ;cte1 = 20 Hex
cte2    EQU 20h             ;cte1 = 50 Hex
cte3    EQU 20h             ;cte1 = 60 Hex

        ORG 0
        GOTO INICIO
        ORG 5
INICIO:
        BSF     STATUS, RP0 ;RP0 = '1'.
        BCF     STATUS, RP1 ;RP1 = '0'.
        MOVLW   H'0'        ;W = 00H.
        MOVWF   TRISB       ;TRISB = 00H.
        BCF     STATUS, RP0 ;RP0 = '0'.
        CLRF    PORTB       ;Limpia el port B.

LOOP2:
        MOVLW   H'80'       ;W = 80 hex.
        BCF     STATUS, 0   ;Carry = '0'.
        MOVWF   PORTB       ;PORTB = 80 hex.
CICLO:
        CALL    RETARDO
        RRF     PORTB, F    ;Rota a la derecha PORTB.
        BTFSS   STATUS, 0   ;Skip si Carry = '1'.
        GOTO    CICLO.
        GOTO    LOOP2

RETARDO:
        MOVLW   cte1        ;Asigna W=20H.
        MOVWF   valor1      ;Mueve W a valor1;

TRES:
        MOVLW   cte2        ;Asigna 50H a W.
        MOVWF   valor2      ;Mueve W a valor2.
DOS:
        MOVLW   cte3        ;Asigna 60H a W.
        MOVWF   valor3      ;Mueve W a valor 3.
UNO:
        DECFSZ  valor3      ;Decrementa valor3 en 1.
        GOTO    UNO         ;Si el resultado es diferente de 0, ir a uno.
        DECFSZ  valor2      ;Decrementa valor2 en 1.
        GOTO    DOS         ;Si el resultado es diferente de 0, ir a dos.
        DECF    valor1      ;Decrementa valor1 en 1.
        GOTO    TRES        ;Si el resultado es diferente de 0, ir a tres.

        RETURN
        END
```

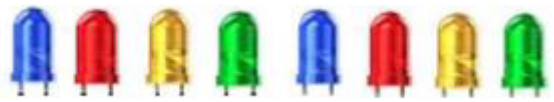
Simulación.


U1
R1 220
R2 220
R3 220
R4 220
R5 220
R6 220
R7 220
R8 220
D1 LED-BLUE
D2 LED-RED
D3 LED-YELLOW
D4 LED-GREEN
D5 LED-BLUE
D6 LED-RED
D7 LED-YELLOW
D8 LED-GREEN


U1
R1 220
R2 220
R3 220
R4 220
R5 220
R6 220
R7 220
R8 220
D1 LED-BLUE
D2 LED-RED
D3 LED-YELLOW
D4 LED-GREEN
D5 LED-BLUE
D6 LED-RED
D7 LED-YELLOW
D8 LED-GREEN

10. Realizar un programa que controle el funcionamiento de dos semáforos. Cada estado tendrá una duración de 2 segundos.

Estado	Salida
1	V1, R2
2	A1, R2
3	R1, V1
4	R1, A2

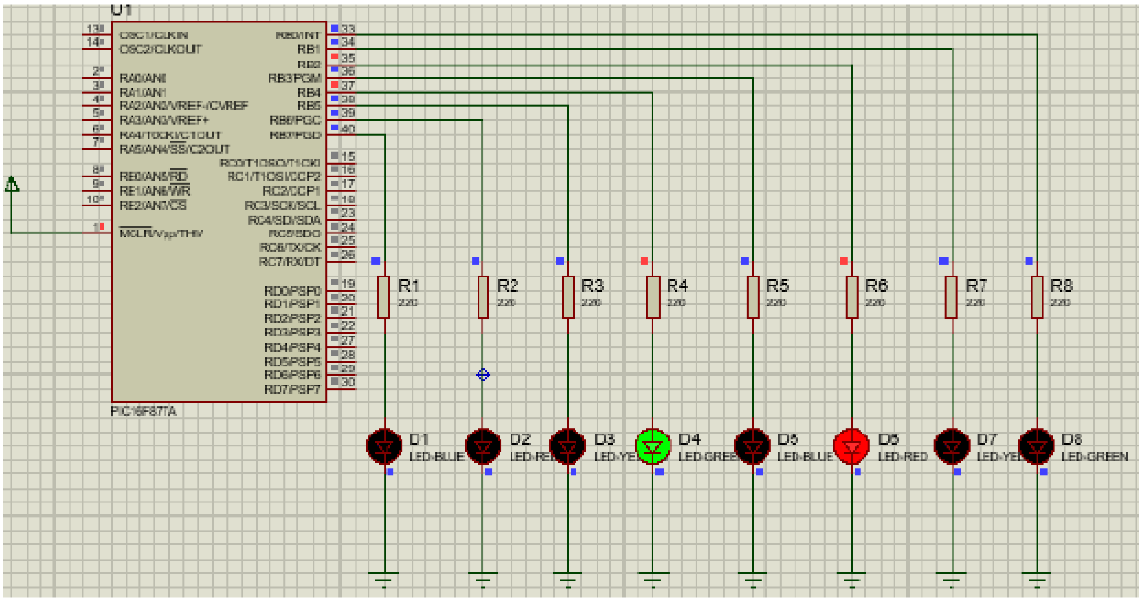


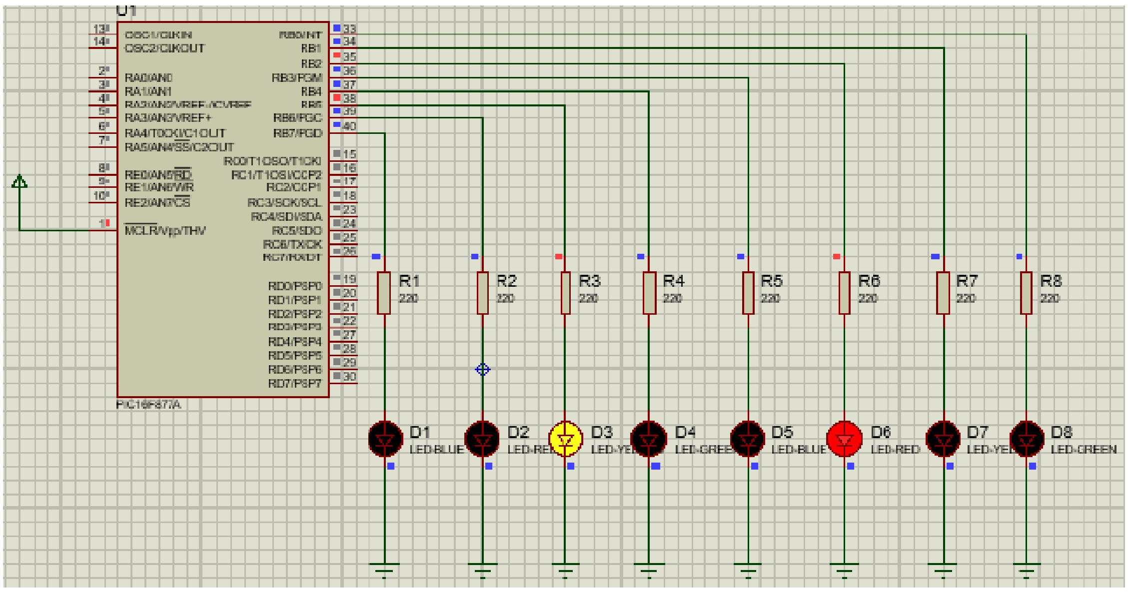
Código:

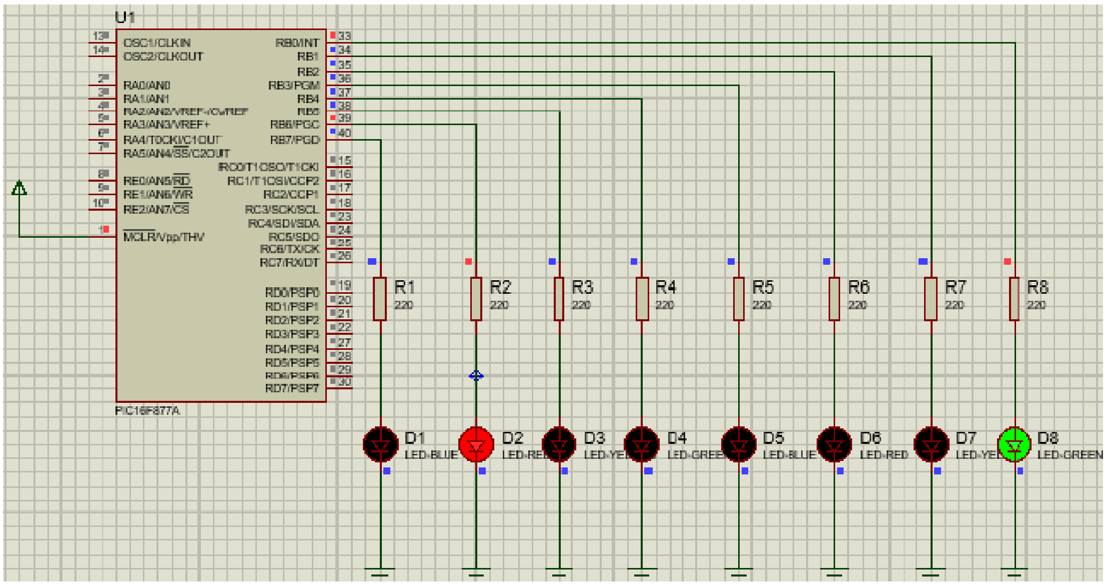
D:\...\E1.asm*

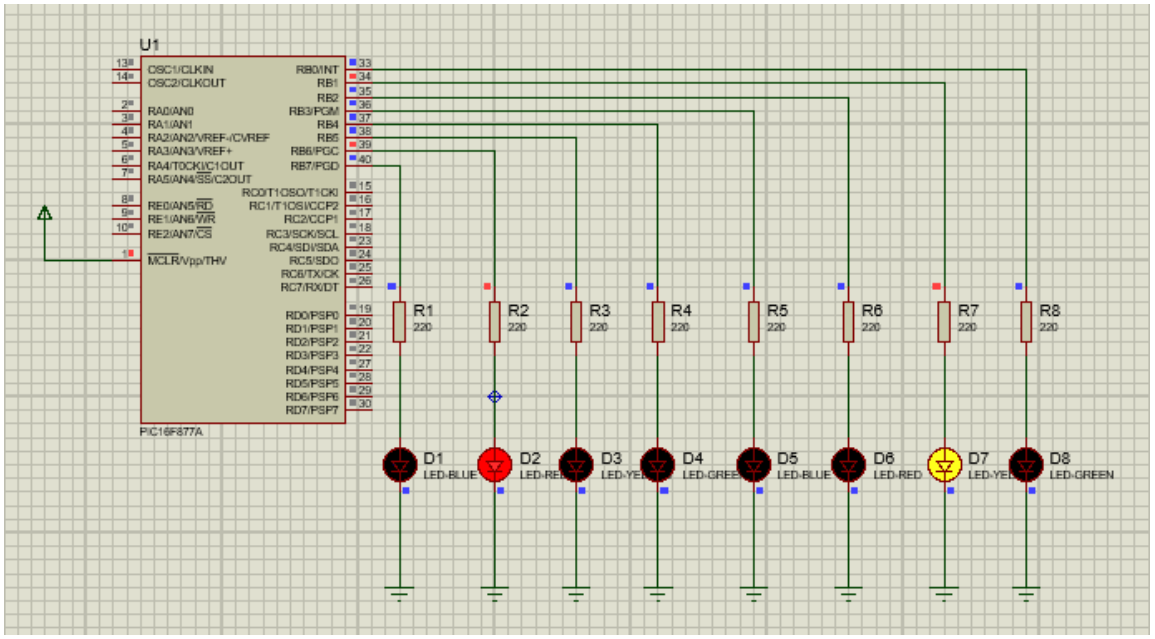
```
valor1  EQU H'21'           ;VALOR1 = 21 hex
valor2  EQU H'22'           ;VALOR1 = 22 hex
valor3  EQU H'22'           ;VALOR1 = 23 hex
ctel    EQU 90h             ;ctel = 90 Hex
cte2    EQU 90h             ;ctel = 90 Hex
cte3    EQU 0A0h            ;ctel = A0 Hex

        ORG 0
        GOTC INICIO
        ORG 5
INICIO:
        BSF     STATUS, RP0 ;RP0 = '1'.
        BCF     STATUS, RP1 ;RP1 = '0'.
        MOVLW   H'0'        ;W = 00H.
        MOVWF   TRISB       ;TRISB = 00H.
        BCF     STATUS, RP0 ;RP0 = '0'.
        CLRF    PORTB       ;Limpia el port B.

LOOP2:
        MOVLW   h'14'       ;Estado 1, verde = 1; rojo = 2.
        MOVWF   PORTB
        CALL    RETARDO

        MOVLW   H'24'       ;Estado 2, amarillo = 1; rojo = 2.
        MOVWF   PORTB
        CALL    RETARDO

        MOVLW   H'41'       ;Estado 3, rojo = 1; verde = 2.
        MOVWF   PORTB
        CALL    RETARDO

        MOVLW   H'42'       ;Estado 4, rojo = 1; verde = 2.
        MOVWF   PORTB
        CALL    RETARDO

        GOTC LOOP2

RETARDO:
        MOVLW   ctel        ;Asigna W=20H.
        MOVWF   valor1      ;Mueve W a valor1;

TRES:
        MOVLW   cte2        ;Asigna 50H a W.
        MOVWF   valor2      ;Mueve W a valor2.
DOS:
        MOVLW   cte3        ;Asigna 60H a W.
        MOVWF   valor3      ;Mueve W a valor 3.
UNO:
        DECFSZ  valor3      ;Decrementa valor3 en 1.
        GOTO    UNO         ;Si el resultado es diferente de 0, ir a uno.
        DECFSZ  valor2      ;Decrementa valor2 en 1.
        GOTO    DOS         ;Si el resultado es diferente de 0, ir a dos.
        DECF    valor1      ;Decrementa valor1 en 1.
        GOTC    TRES        ;Si el resultado es diferente de 0, ir a tres.

        RETURN
        END
```

Simulación:


U1
OSC1/CLKIN
OSC2/CLKOUT
RA0/AN0
RA1/AN1
RA2/AN2/VREF-/CVREF
RA3/AN3/VREF+
RA4/T0CKI/C1OUT
RA5/AN4/SS/C2OUT
RE0/AN5/RD
RE1/AN6/WR
RE2/AN7/CS
MCLR/Vpp/THV
RB0/INT
RB1
RB2
RB3/PGM
RB4
RB5
RB6/PGC
RB7/PGD
RC0/T1OSO/T1CKI
RC1/T1OSI/CCP2
RC2/CCP1
RC3/SCK/SCL
RC4/SDI/SDA
RC5/SDO
RC6/TX/CK
RC7/RX/DT
RD0/PSP0
RD1/PSP1
RD2/PSP2
RD3/PSP3
RD4/PSP4
RD5/PSP5
RD6/PSP6
RD7/PSP7
PIC16F877A
R1 220
R2 220
R3 220
R4 220
R5 220
R6 220
R7 220
R8 220
D1 LED-BLUE
D2
D3
D4 LED-GREE
D5 LED-BLUE
D6 LED-RED
D7
D8 LED-GREEN


U1
OSC1/CLKIN
OSC2/CLKOUT
RA0/AN0
RA1/AN1
RA3/AN3/VREF+
RA4/T0CKI/C1OUT
RA5/AN4/SS/C2OUT
RE0/AN5/RD
RE1/AN6/WR
RE2/AN7/CS
MCLR/Vpp/THV
RB0/INT
RB1
RB2
RB3/PGM
RB4
RB5
RB6/PGC
RB7/PGD
RC0/T1OSO/T1CKI
RC1/T1OSI/CCP2
RC2/CCP1
RC3/SCK/SCL
RC4/SDI/SDA
RC5/SDO
RC6/TX/CK
RD0/PSP0
RD1/PSP1
RD2/PSP2
RD4/PSP4
RD5/PSP5
RD6/PSP6
RD7/PSP7
PIC16F877A
R1 220
R2 220
R3 220
R4 220
R5 220
R6 220
R7 220
R8 220
D1 LED-BLUE
D2
D3
D4 LED-GREE
D5 LED-BLUE
D6 LED-RED
D7
D8 LED-GREEN


U1
OSC1/CLKIN
OSC2/CLKOUT
RA0/AN0
RA1/AN1
RA3/AN3/VREF+
RA4/T0CKI/C1OUT
RA5/AN4/SS/C2OUT
RE0/AN5/RD
RE1/AN6/WR
RE2/AN7/CS
MCLR/Vpp/THV
RB0/INT
RB1
RB2
RB3/PGM
RB4
RB6/PGC
RB7/PGD
RC0/T1OSO/T1CKI
RC1/T1OSI/CCP2
RC2/CCP1
RC3/SCK/SCL
RC4/SDI/SDA
RC5/SDO
RC6/TX/CK
RC7/RX/DT
RD0/PSP0
RD1/PSP1
RD2/PSP2
RD3/PSP3
RD4/PSP4
RD5/PSP5
RD7/PSP7
PIC16F877A
R1 220
R2 220
R3 220
R4 220
R5 220
R6 220
R7 220
R8 220
D1 LED-BLUE
D2
D3
D4 LED-GREE
D5 LED-BLUE
D6 LED-RED
D7
D8 LED-GREEN



Jose Francisco Ugalde Vivo
En la presente práctica aprendimos a relacionar esquemas o diagramas de sistemas mínimos de microcontroladores a través de un software de comunicación, además y como parte práctica realizamos diversos programas con la finalidad de utilizar los puertos de salida del microcontrolador en una modalidad de salida.
Por último, realizamos simulaciones mediante Proteus con la finalidad de representar nuestros programas mediante hardware que nos hiciera explícitos nuestros resultados además de ejecutar programas en tiempo real

Hector Montoya Perez
En esta práctica pudimos interpretar los diagramas o esquemáticos de los sistemas de microcontrolador, así como el software dedicado, por otro lado, se logró observar e implementar la lógica detrás de los programas que utilizan componentes o puertos de salida todo esto mediante simulaciones de Proteus, que nos dan un paso para implementarlo en físico, retomando así la programación a bajo nivel de estos dispositivos.

Lara Salas Kevin Arturo
En la práctica se pudo aterrizar los conceptos vistos en la parte de programación en el simulador de Proteus. Esto permite a los alumnos en la creación de componentes ya programados para, en un futuro, poder implementarlos en físico.
Además, se pudieron revisar los esquemáticos de los componentes para darnos una idea de cómo se manejarían en formato físico