	Laboratorio de Microcomputadoras

Profesor(a): Dra. Lourdes Angelica Quiñonez Juárez

Asignatura: Laboratorio de Microcomputadoras

Grupo Laboratorio: 5

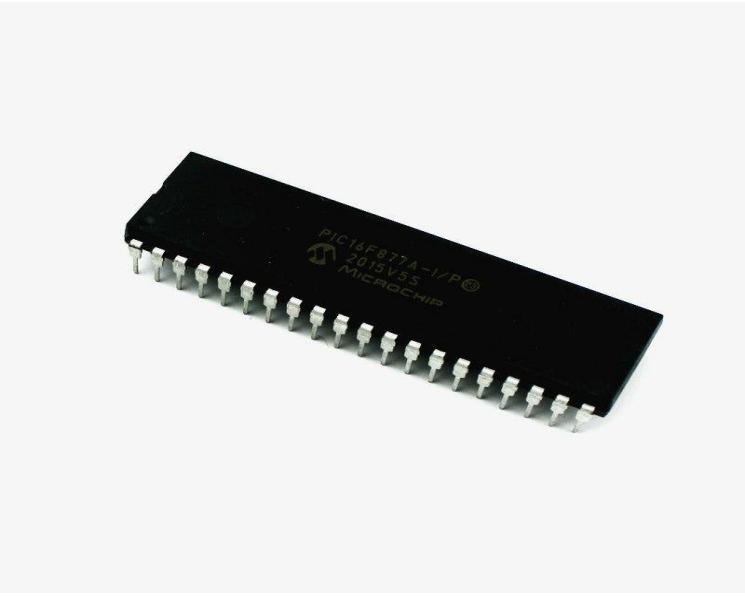
Grupo Teoría: 3

No. de Práctica(s): 3

Integrante(s): Hernández Diaz Sebastián

Semestre: 2025-2

Fecha de entrega: 10 de marzo del 2025



Hernández Diaz Sebastian

Laboratorio de Microcomputadoras

Práctica No 3

Puertos Paralelos II (Control de acciones)

Objetivo. Emplear los puertos paralelos que contiene un microcontrolador para realizar funciones de control, configurando estos como entrada y salida.

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

1. **Empleando dos puertos paralelos del microcontrolador PIC, uno de ellos configurado como entrada y el otro como salida; realizar un programa que de acuerdo al valor del bit menos significativo del puerto A, se genere la acción indicada en el puerto B.**

Valor PA0	Acción puerto B
0	00000000
1	11111111

Tabla Control de salidas controladas por un bit.

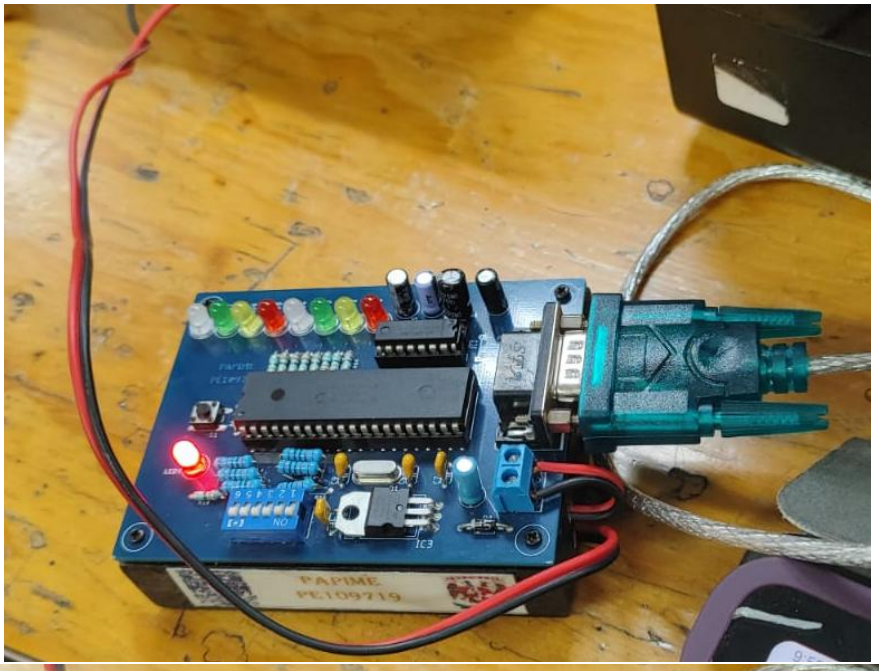
El código de este ejercicio es el siguiente:

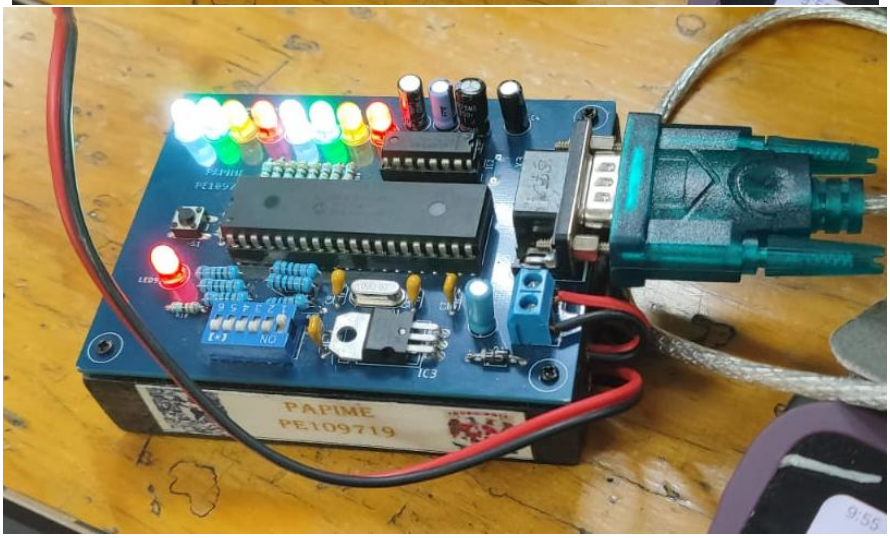
```
processor 16F877      ; Define el procesador que se usará
include<p16f877.inc>

ORG 0                 ; Dirección de inicio
GOTO inicio           ; Salta a la etiqueta "inicio"

ORG 5                 ; Dirección de inicio del código principal

inicio:
    bsf STATUS, RP0   ; Establece el bit RP0 (STATUS 5) en 1 para cambiar al banco 1
    BCF STATUS, RP1   ; Asegura que el bit RP1 (STATUS 6) sea 0 para seleccionar banco 1

    MOVLW H'07'       ; Carga el valor 0x07 en W
    MOVWF ADCON1      ; Configura ADCON1 para que los pines del Puerto A sean digitales

    MOVLW H'ff'       ; Carga 0xFF en W
    MOVWF TRISA       ; Establece el Puerto A como entrada (todos los bits en 1)

    CLRF TRISB        ; Establece el Puerto B como salida (todos los bits en 0)

    BCF STATUS, RP0   ; Regresa al banco 0

switch:
    BTFSC PORTA, 0    ; Revisa el bit 0 del Puerto A, si es 0 sigue a la siguiente instrucción
    GOTO encender     ; Si el bit 0 de PORTA está en 1, salta a "encender"

    CLRF PORTB        ; Si el bit 0 de PORTA está en 0, limpia el Puerto B (apaga los LEDs)
    GOTO switch       ; Regresa a evaluar el estado del switch

encender:
    MOVLW h'ff'       ; Carga 0xFF en W (todos los bits en 1)
    MOVWF PORTB       ; Activa todos los pines del Puerto B (enciende los LEDs)
    GOTO switch       ; Regresa al loop principal para seguir evaluando el switch

END
```

Este programa configura el Puerto A como entrada y el Puerto B como salida. Luego, detecta el estado del bit 0 del Puerto A (PA0). Si el bit está en alto (1), enciende todos los bits del Puerto B (PB0-PB7). Si está en bajo (0), apaga todos los bits del Puerto B.

Este programa demuestra el uso de bancos de memoria, configuración de ADC, uso de puertos de entrada/salida y control de flujo condicional con instrucciones como BTFSC, GOTO, MOVWF y CLRF.

El programa en ejecución es el siguiente:





Primero tenemos los leds apagados donde se puede ver también que el switch esta desactivado o en cero, mientras que en la imagen de abajo se muestra el switch activado en 1 y con esto todos los leds prenden de manera correcta.

2. **Realizar un programa, el cuál realice las siguientes acciones de control, para lo cual requiere trabajar un puerto de entrada y otro puerto de salida, usar los sugeridos en el ejercicio anterior; generar retardos de 1⁄2 seg., en las secuencias que lo requieran.**

DATO	ACCION	*Ejecución*
$00	Todos los leds apagados	*00000000*
$01	Todos los leds encendidos	*11111111*
$02	Corrimiento del bit más significativo hacia la derecha	*10000000* *01000000* *00100000* …….. *00000001*
$03	Corrimiento del bit menos significativo hacia la izquierda	*00000001* *00000010* *00000100* …….. *10000000*
$04	Corrimiento del bit más significativo hacia la derecha y a la izquierda	*10000000* *01000000* …….. *00000001* *00000010* …….. *10000000*
$05	Apagar y encender todos los bits.	*00000000* *11111111*

Tabla Control de salidas completo.

El código utilizado fue el siguiente:

Hernández Diaz Sebastian

```
processor 16F877        ; Define el procesador que se usará
include<p16f877.inc>

contador equ h'20'      ; Define la variable contador en la dirección 0x20
valor1 equ h'21'        ; Define la variable valor1 en la dirección 0x21
valor2 equ h'22'        ; Define la variable valor2 en la dirección 0x22
valor3 equ h'23'        ; Define la variable valor3 en la dirección 0x23

cte1 equ 20h            ; Define la constante cte1 con valor 0x20
cte2 equ 50h            ; Define la constante cte2 con valor 0x50
cte3 equ 60h            ; Define la constante cte3 con valor 0x60

ORG 0                   ; Dirección de inicio del programa
GOTO inicio             ; Salta a la función inicio

ORG 5                   ; Dirección de inicio del código principal
inicio:
    bsf STATUS, RP0     ; Activa el bit RP0 para seleccionar el Banco 1
    BCF STATUS, RP1     ; Asegura que el bit RP1 es 0 para trabajar en el Banco 1

    MOVLW H'07'         ; Carga 0x07 en W
    MOVWF ADCON1        ; Configura ADCON1 para deshabilitar el ADC (modo digital)

    MOVLW H'ff'         ; Carga 0xFF en W
    MOVWF TRISA         ; Configura el Puerto A como entrada

    CLRF TRISB          ; Configura el Puerto B como salida (todos los bits en 0)

    BCF STATUS, RP0     ; Regresa al Banco 0

switch:
    MOVF PORTA, 0       ; Carga el valor de PORTA en W
    XORLW H'00'         ; Compara con 0
    BTFSC STATUS, Z     ; Si es igual a 0, salta a la siguiente instrucción
    call apagar         ; Llama a la función para apagar PORTB

    MOVF PORTA, 0
    XORLW h'01'
    BTFSC STATUS, Z
    call encender       ; Llama a la función para encender PORTB

    MOVF PORTA, 0
    XORLW H'02'
    BTFSC STATUS, Z
    CALL corrDerecha    ; Llama a la función para corrimiento a la derecha
```

```
    MOVF PORTA, 0
    XORLW H'03'
    BTFSC STATUS, Z
    CALL corrIzquier  ; Llama a la función para corrimiento a la izquierda

    MOVF PORTA, 0
    XORLW H'04'
    BTFSC STATUS, Z
    CALL ambCorr      ; Llama a la función para corrimiento bidireccional

    MOVF PORTA, 0
    XORLW H'05'
    BTFSC STATUS, Z
    CALL enceApagar   ; Llama a la función de encender y apagar

    GOTO switch       ; Regresa al bucle para seguir verificando la entrada

encender:
    MOVLW h'ff'       ; Carga 0xFF en W (todos los bits en 1)
    MOVWF PORTB       ; Escribe en PORTB para encender todos los LEDs
    RETURN            ; Regresa al bucle principal

apagar:
    CLRF PORTB        ; Limpia PORTB (apaga todos los LEDs)
    RETURN            ; Regresa al bucle principal

corrDerecha:
    MOVLW h'80'       ; Carga 1000 0000 (bit más alto encendido)
    MOVWF PORTB       ; Lo escribe en PORTB
    CALL retardo      ; Llama a la función de retardo

    MOVLW h'07'       ; Contador de 7 desplazamientos
    MOVWF contador

    corrimi:
        RRF PORTB, 1      ; Desplaza a la derecha con rotación
        CALL retardo
        DECF contador, 1
        BTFSS STATUS, 2  ; Si el contador llega a 0, sale del bucle
        GOTO corrimi

        RETURN
```

```
corrIzquier:
    MOVLW h'01'      ; Carga 0000 0001 (bit más bajo encendido)
    MOVWF PORTB
    CALL retardo

    MOVLW h'07'      ; Contador de 7 desplazamientos
    MOVWF contador

    corrimiento:
        RLF PORTB, 1     ; Desplaza a la izquierda con rotación
        CALL retardo
        DECF contador, 1
        BTFSS STATUS, 2
        GOTO corrimiento

        RETURN

ambCorr:
    CALL corrDerecha  ; Llama a la función de corrimiento a la derecha
    CALL corrIzquier  ; Llama a la función de corrimiento a la izquierda
    RETURN

enceApagar:
    CALL encender   ; Enciende todos los LEDs
    CALL retardo    ; Espera un tiempo
    CALL apagar     ; Apaga todos los LEDs
    CALL retardo    ; Espera otro tiempo
    RETURN
retardo:
    MOVLW cte1
    MOVWF valor1

tres:
    MOVLW cte2
    MOVWF valor2

dos:
    MOVLW cte3
    MOVWF valor3

uno:
    DECFSZ valor3    ; Decrementa valor3, si no es 0, repite
    GOTO uno
    DECFSZ valor2    ; Decrementa valor2, si no es 0, repite
    GOTO dos
    DECFSZ valor1    ; Decrementa valor1, si no es 0, repite
    GOTO tres
    RETURN

END
```

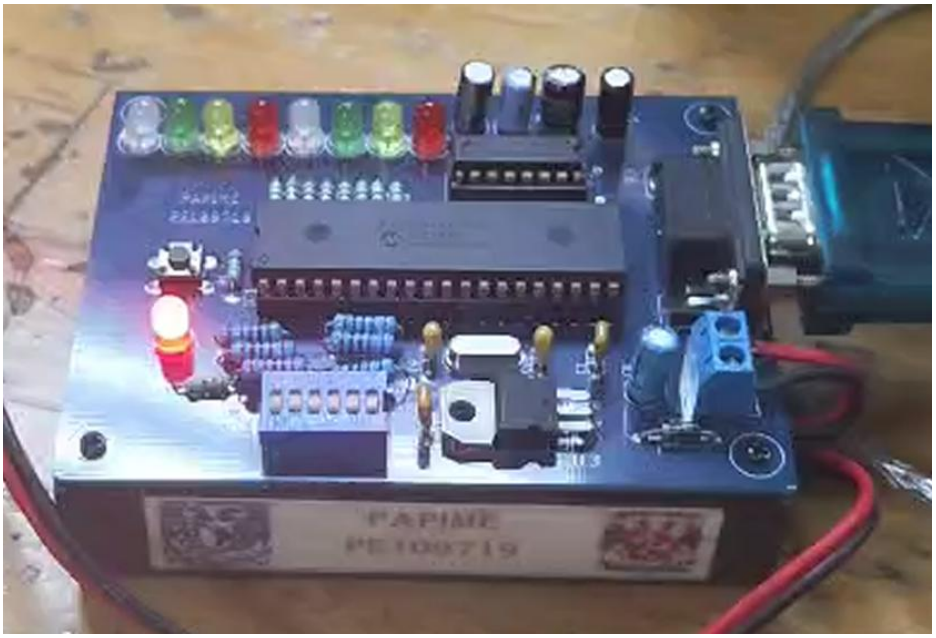
Este programa configura el Puerto A como entrada y el Puerto B como salida. Luego, detecta el estado del Puerto A y, dependiendo del valor leído, ejecuta una de las siguientes funciones en el Puerto B:

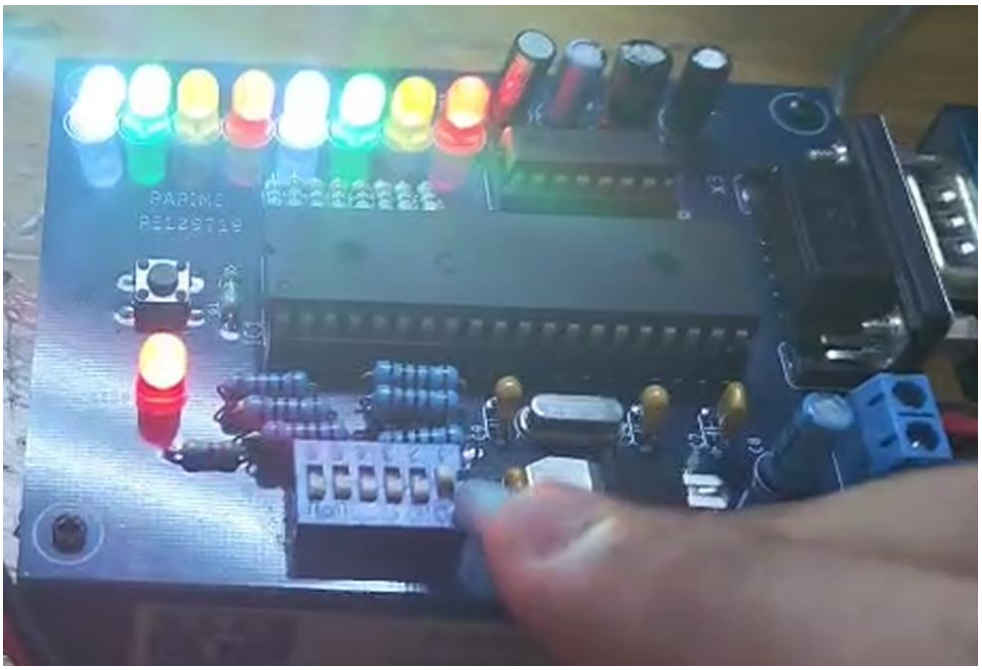
1. **Encender** todos los bits del Puerto B.
2. **Apagar** todos los bits del Puerto B.
3. **Corrimiento a la derecha** (los LEDs se mueven de izquierda a derecha).
4. **Corrimiento a la izquierda** (los LEDs se mueven de derecha a izquierda).
5. **Corrimiento bidireccional** (se mueve en ambas direcciones).
6. **Encender y apagar alternadamente** con retardo.

Este programa usa interacciones básicas con switches y manejo de LEDs mediante instrucciones como BTFSC, XORLW, MOVF y CALL. Aprendemos a configurar puertos de entrada/salida, manejar retardos, realizar desplazamientos bit a bit (RLF y RRF) y estructurar código modular con subrutinas. La lógica de comparación bit a bit con XORLW y BTFSC permite seleccionar diferentes modos de control mediante el switch.

El código en funcionamiento es el siguiente:

Encender todos los bits del Puerto B.



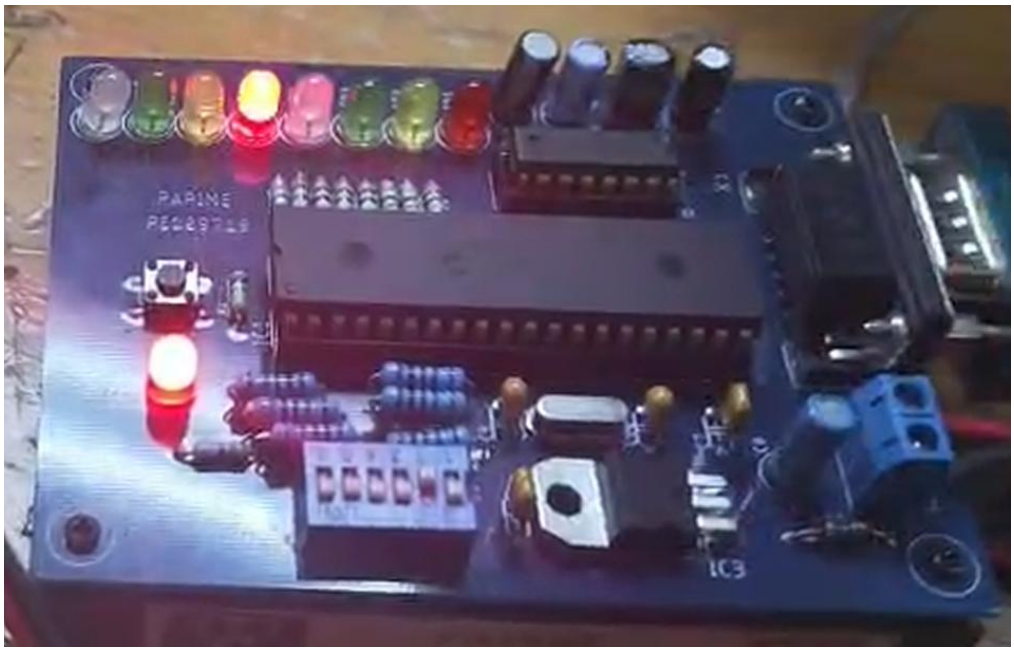


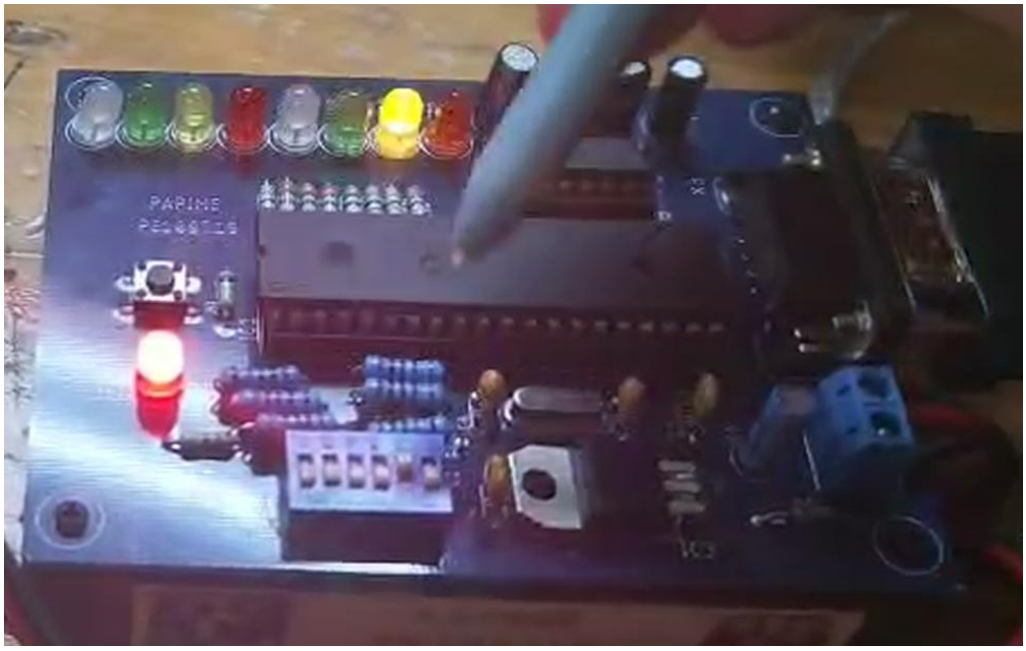
Como se puede ver cuando el switch se coloca en 1 los leds del microprocesador se encienden de la manera solicitada.

Apagar todos los bits del Puerto B.

En este caso al colocar los bits en cero después de encenderlos se puede ver que estos se apagan como era solicitado.

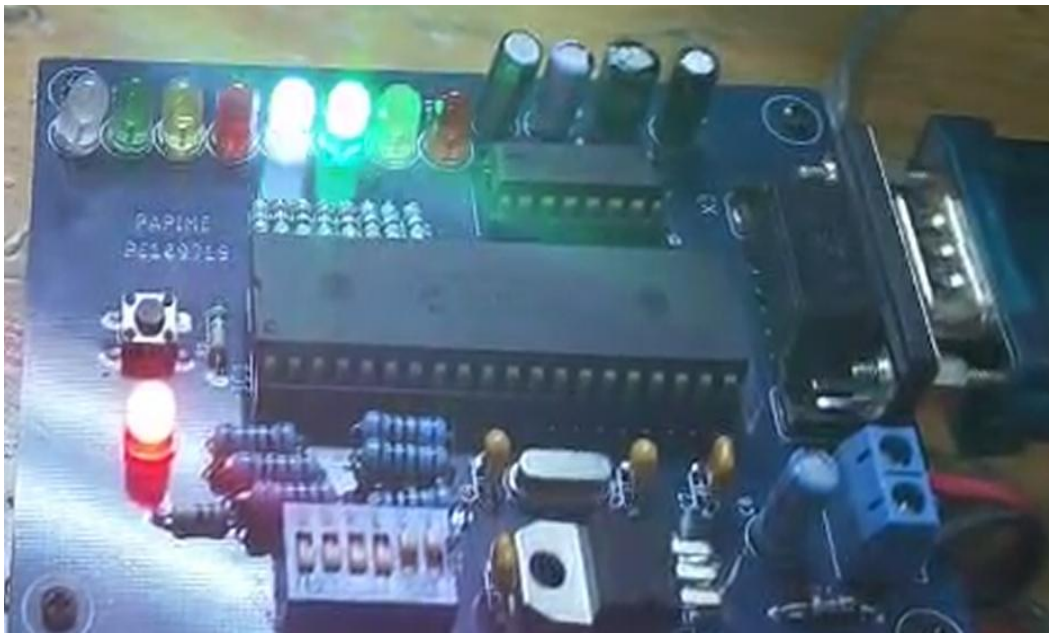
Corrimiento a la derecha (los LEDs se mueven de izquierda a derecha).

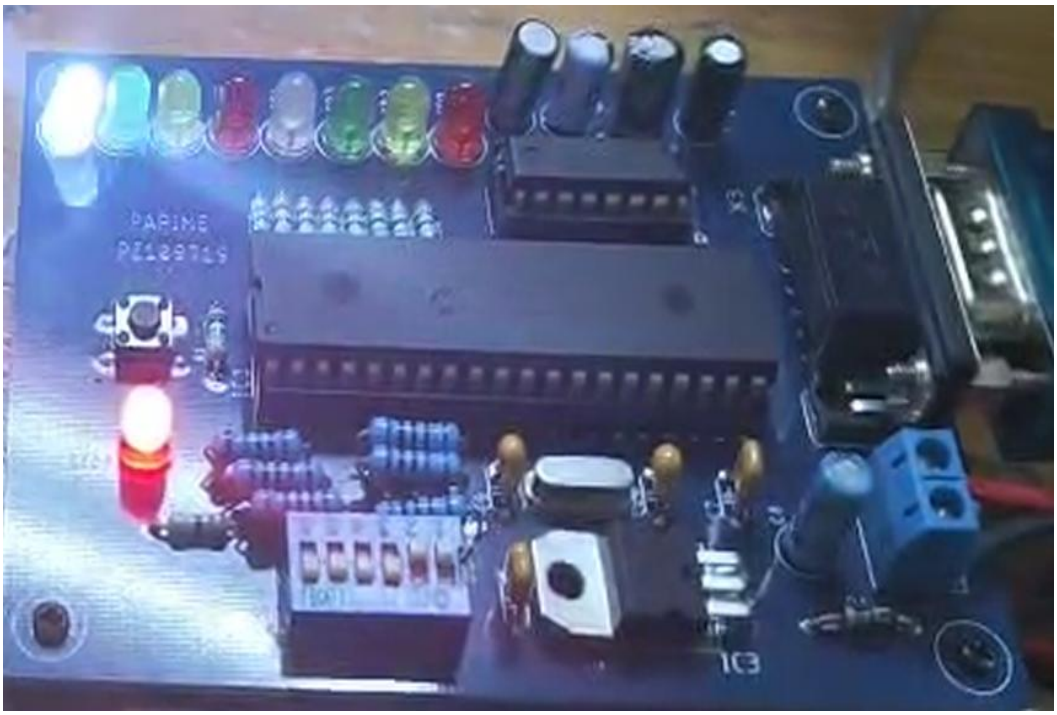




En este caso se puede ver el switch en la posición de 2 en este caso se presenta un corrimiento a la derecha de los leds.

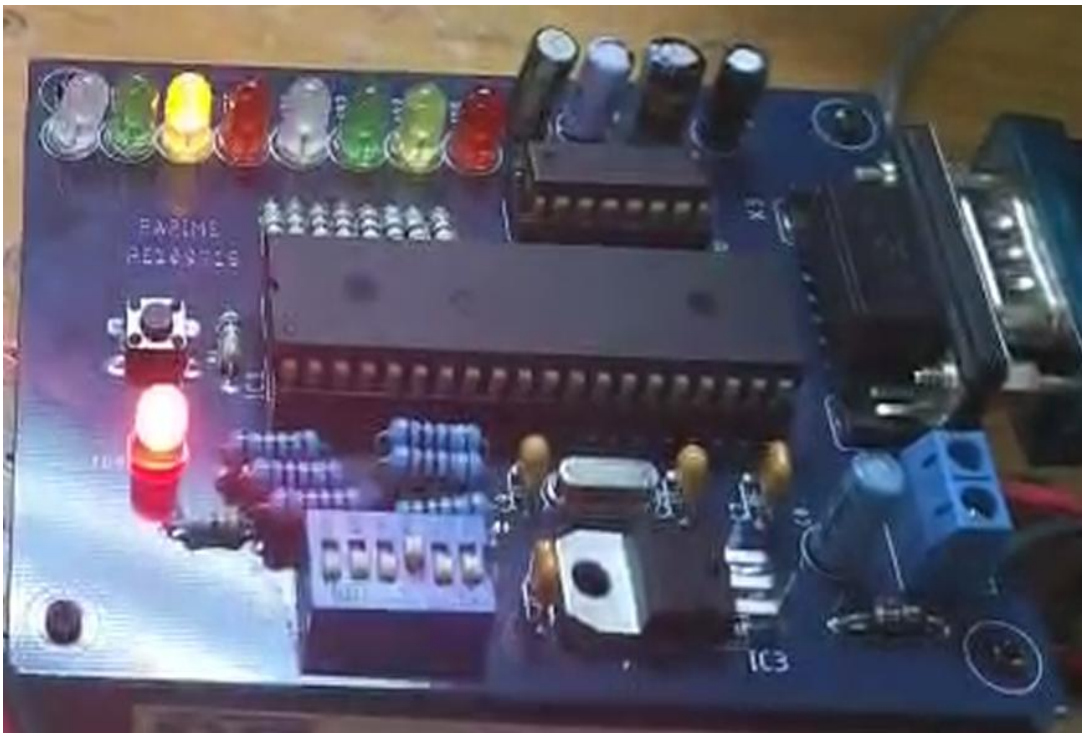
Corrimiento a la izquierda (los LEDs se mueven de derecha a izquierda).

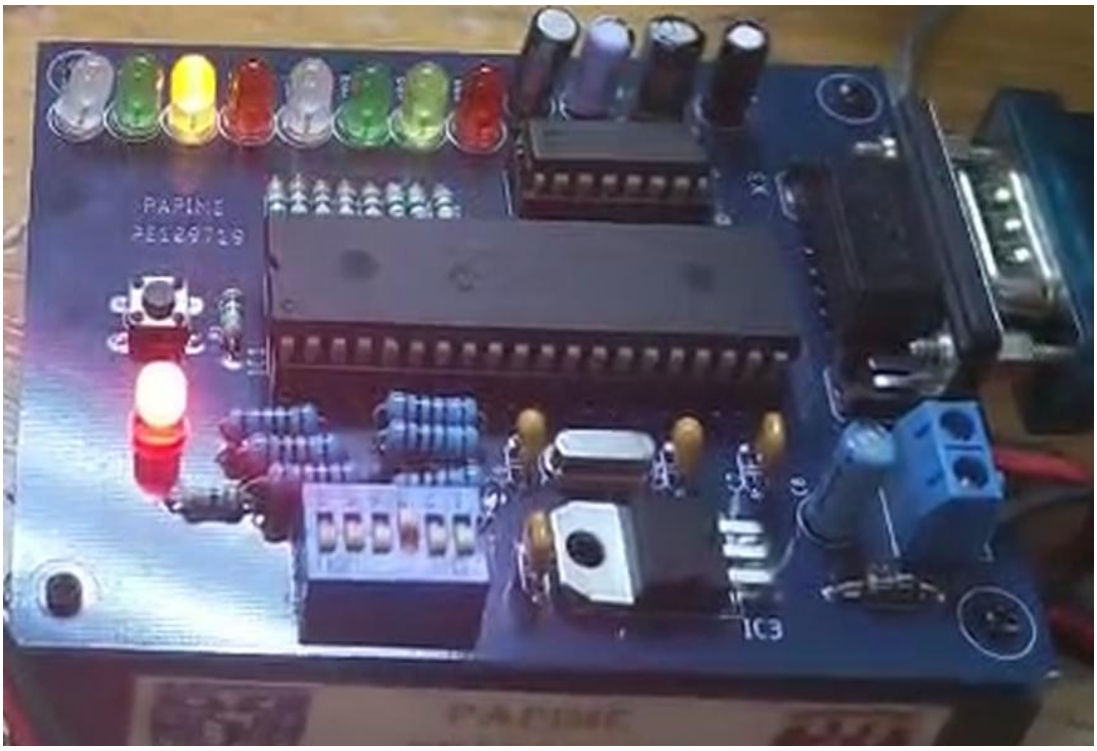




En este caso se puede ver el switch en la posición de 3 en este caso se presenta un corrimiento a la izquierda de los leds.

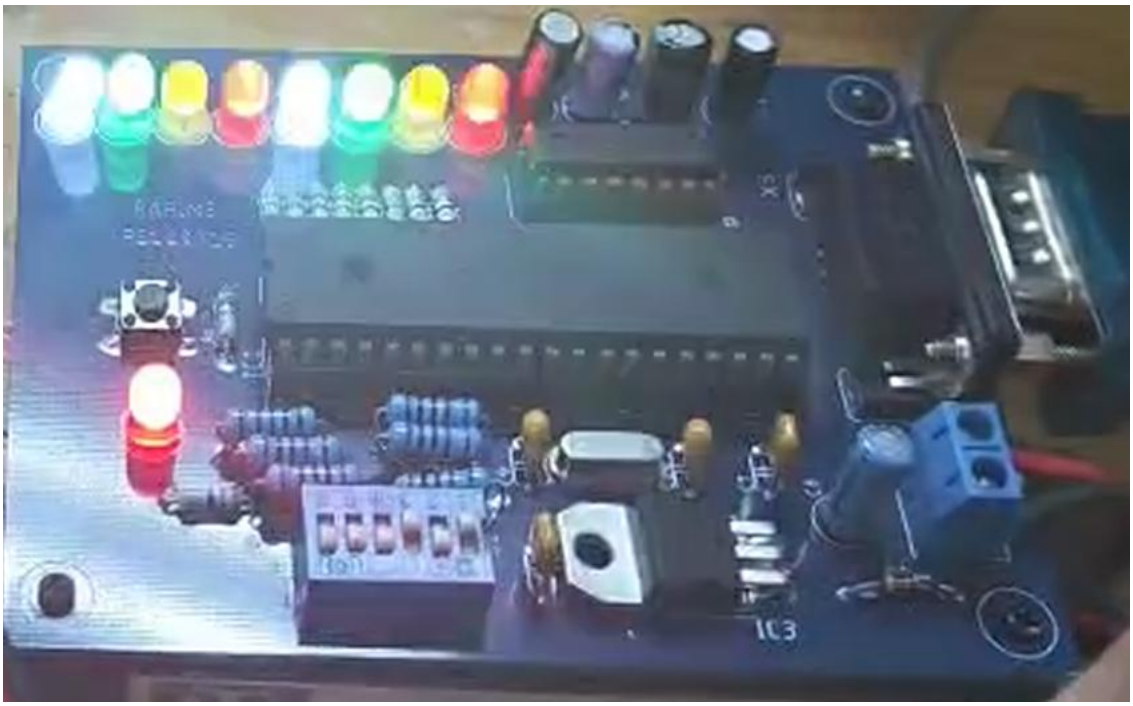
Corrimiento bidireccional (se mueve en ambas direcciones).





En este caso se puede ver el switch en la posición de 4 en este caso se presenta un corrimiento a la derecha de los leds y después de que este recorrido termina comienza a recorrerlos hacia la izquierda y así sucesivamente de derecha a izquierda.

Encender y apagar alternadamente con retardo.



En este caso se puede ver el switch en la posición de 5 en este caso se presenta un comportamiento de encendido y apagado cada medio segundo como era solicitado en la práctica.

Hernández Diaz Sebastian

Conclusiones

En estos ejercicios aprendí a configurar y manipular los puertos de entrada y salida del microcontrolador PIC16F877, específicamente el Puerto A como entrada y el Puerto B como salida, lo que me permitió interactuar con switches y LEDs para generar diferentes secuencias de encendido y apagado. Comprendí la importancia del banco de memoria y cómo cambiar entre bancos utilizando los bits RP0 y RP1 del registro STATUS para acceder a los registros de configuración como TRISA, TRISB y ADCON1. También trabajé con instrucciones de control de flujo como BTFSC y GOTO, que permiten evaluar el estado de un puerto y ejecutar diferentes funciones en consecuencia. El uso de operaciones lógicas como XORLW me permitió identificar combinaciones específicas de entradas y activar la función correspondiente. Para el control de los LEDs, utilicé manipulaciones directas de bits con BSF, BCF, MOVLW, MOVWF y CLRF, que permitieron encender, apagar y modificar patrones de salida en el Puerto B. Además, implementé desplazamientos de bits con RRF y RLF, que generaron efectos visuales de corrimiento a la izquierda y derecha. También aprendí a modular el código en funciones reutilizables, facilitando su mantenimiento y claridad. El uso de retardos fue clave para la visualización de los cambios en los LEDs, utilizando bucles de decremento con DECFSZ para generar pausas perceptibles. En general, estos ejercicios me ayudaron a comprender la estructura básica de programación en ensamblador para microcontroladores PIC.

Bibliografía

Microchip Technology Inc. (n.d.). *PIC16F877A Data Sheet*. Recuperado de https://www.microchip.com

Enginnering Projects. (n.d.). *PIC16F877A LED Blinking Program in Assembly*. Recuperado de https://www.engineersgarage.com

Gooligum Electronics. (n.d.). *PIC Assembly Language Programming*. Recuperado de http://www.gooligum.com.au/tutorials.html

Pabón, E. (2015). *Guía de programación en ensamblador para PIC16F877A*. Recuperado de https://electroensaimada.com