



**Universidad Nacional Autónoma de
México
Facultad de Ingeniería**



Semestre 2023-2

Laboratorio de Microcomputadoras

**Práctica 5:
Control de Actuadores**

Profesor:

M.I. Ruben Anaya García

Alumnos:

Barreiro Valdez Alejandro

Zepeda Baeza Jessica

Número de cuenta:

317520888

317520747

Grupo Laboratorio: 4

Grupo Teoría: 1

Fecha de realización: 25 de abril de 2023

Fecha de entrega: 29 de abril de 2023

Desarrollo

Para esta práctica se buscó crear programas en ensamblador que hicieran uso de los puertos paralelos del microcontrolador PIC16F877 para controlar la operación de motores de corriente directa, motores a pasos y servomotores. Para ello se observaron las terminales que cada motor tiene y a qué puertos del PIC16F877 están conectados. También se analizó el comportamiento de cada motor y qué valores necesitan en sus terminales para girar horario o antihorario. Los puertos B y C fueron los conectados a las terminales de los motores y se definieron como salidas; mientras el puerto A se utilizó como entrada ya que se conectó a un Dip Switch.

Los programas desarrollados fueron:




1. Un programa que según la entrada realiza una acción en dos motores de corriente directa, como se muestra en la tabla:

DATO Puerto Paralelo	ACCION	
	MOTOR M1	MOTOR M2
0x00	PARO	PARO
0x01	PARO	HORARIO
0x02	PARO	ANTI-HORARIO
0x03	HORARIO	PARO
0x04	ANTI-HORARIO	PARO
0x05	HORARIO	HORARIO
0x06	ANTI-HORARIO	ANTI-HORARIO
0x07	HORARIO	ANTI-HORARIO
0x08	ANTI-HORARIO	HORARIO

2. Un programa que controla la cantidad de pasos y el sentido de giro del motor a pasos según la entrada, como se muestra en la tabla:

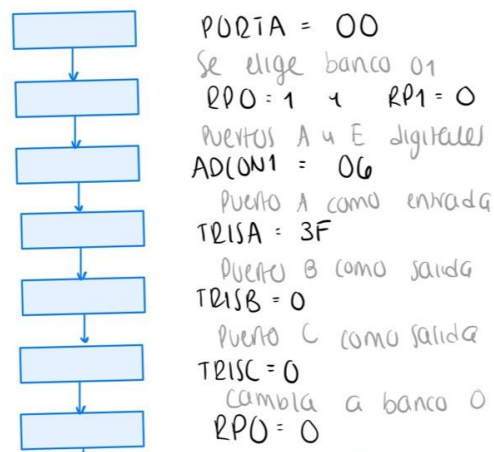
Dato Puerto Paralelo	Motor a pasos
0x00	Motor en paro
0x01	Gira en sentido horario
0x02	Gira en sentido anti horario
0x03	Gira cinco vueltas en sentido horario
0x04	Gira 10 vueltas en sentido anti horario

3. Un programa que mueve un servomotor a 0°, 90° y 180° según la entrada, como se muestra en la tabla:

SW2	SW1	SW0	Posición Servo	Representación
1	0	0	Izquierda	 0°
0	1	0	Central	 90°
0	0	1	Derecha	 180°

Algoritmos

Para los tres ejercicios se tiene la siguiente carta ASM que muestra la configuración de los puertos de entrada y salida. Esto se realiza al principio de cada programa realizado.



Ejercicio 1

Para el Ejercicio 1 primero se definieron los valores que se necesitan en los puertos del microcontrolador según las terminales conectadas del Motor1 y Motor2 para que giren en sentido horario y antihorario.

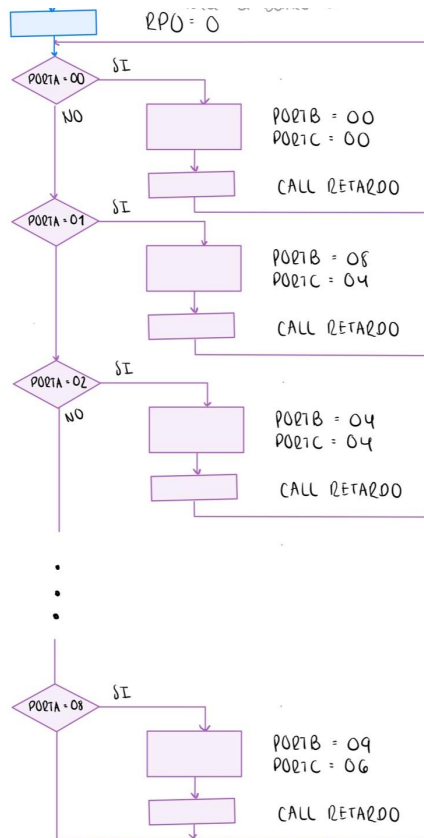
MOTOR2		
PC2	PB3	PB2
ENABLE_M2	DIR1_M2	DIR2_M2

MOTOR1		
PC1	PB1	PB0
ENABLE_M1	DIR1_M1	DIR2_M1

De acuerdo a la imagen anterior se obtuvieron los siguientes valores:

Motor1	PC1	PB1 PB0	Motor 2	PC2	PB3 PB2	PC	PB
Paro	0	00	Paro	0	00	0000 = 00	0000 = 00
Paro	0	00	Horario	1	10	0100 = 04	1000 = 08
Paro	0	00	Antihorario	1	01	0100 = 04	0100 = 04
Horario	1	10	Paro	0	00	0010 = 02	0010 = 02
Antihorario	1	01	Paro	0	00	0010 = 02	0001 = 01
Horario	1	10	Horario	1	10	0110 = 06	1010 = 0A
Antihorario	1	01	Antihorario	1	01	0110 = 06	0101 = 05
Horario	1	10	Antihorario	1	01	0110 = 06	0110 = 06
Antihorario	1	01	Horario	1	10	0110 = 06	1001 = 09

Después se realizó la carta ASM:



El programa compara el valor del Puerto A empezando del 0 hasta 8 para verificar qué movimiento en los motores hacer. Si es igual a alguno de ellos modifica el valor del Puerto B y Puerto C (de acuerdo a los valores mostrados en la tabla anterior) y llama un retardo para que sea visible el movimiento completo de los motores. Una vez que termina el retardo vuelve a verificar el valor del Puerto A.

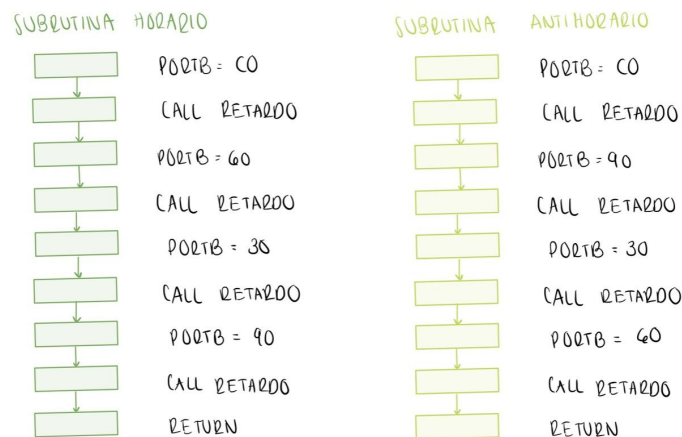
Ejercicio 2

Para el Ejercicio 2 primero se definieron los valores necesarios que debe tomar el puerto B para hacer que el motor a pasos de un giro horario y antihorario según sus bobinas.

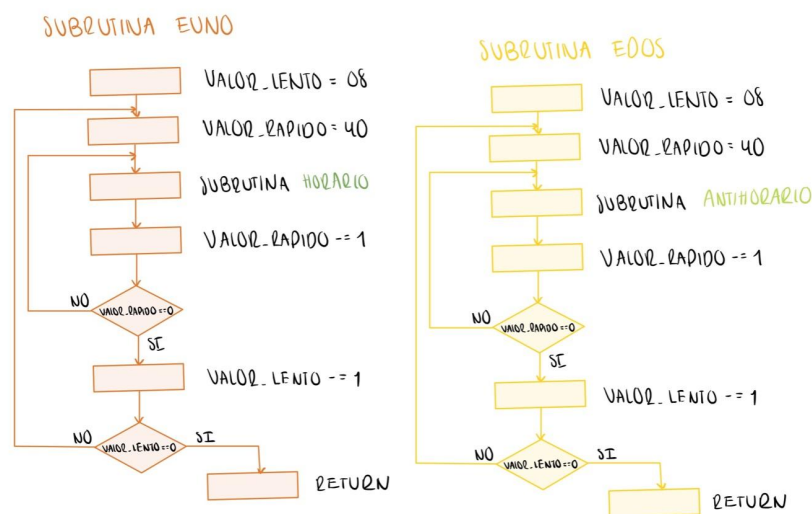
Paso	Bobina A - PB7	Bobina B - PB6	Bobina C - PB5	Bobina D - PB4	PB
1	1	1	0	0	C0
2	0	1	1	0	60
3	0	0	1	1	30
4	1	0	0	1	90

El sentido antihorario se logra haciendo los pasos: 1, 4, 3, 2; y el horario: 1, 2, 3, 4. Estas transiciones se incorporaron en una subrutina que asigna dichos valores al Puerto B y luego

llama un retardo para hacer visible el movimiento. La carta ASM de las subrutinas es la siguiente:

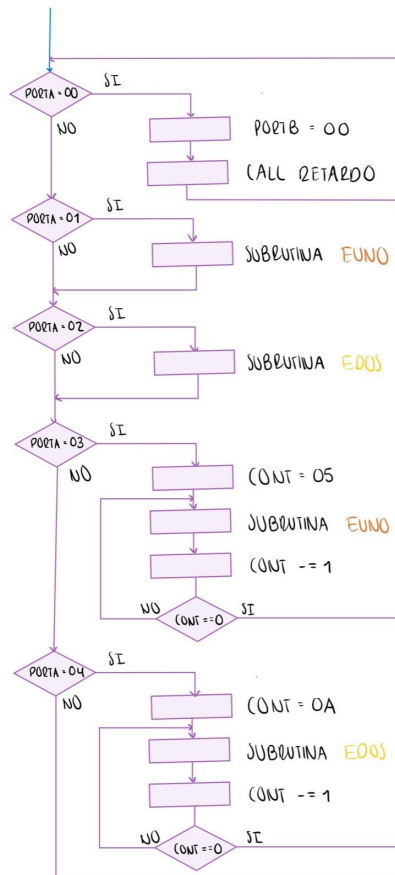


Debido a que se trata de un motor a pasos, se debe realizar todo el movimiento 512 veces para lograr una vuelta en el motor, por lo que se implementaron otras dos subrutinas *EUNO* y *EDOS*. Estas subrutinas realizan instrucciones parecidas a las de la subrutina *RETARDO* ya que hay un ciclo rápido de 64 iteraciones y un ciclo lento de 8 iteraciones; por lo tanto se repite 512 veces. En cada iteración manda a llamar a la subrutina *HORARIO* (para *EUNO*) o *ANTIHORARIO* (para *EDOS*).



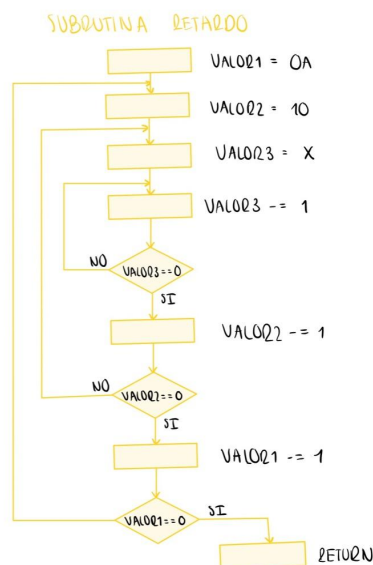
Por último se tiene la carta ASM del programa, que después de la configuración de los puertos, verifica el valor del Puerto A. Si el valor es 0, el Puerto B es 0, se llama un retardo y no se mueve el motor. Si el valor es 1, llama la subrutina *EUNO* para hacer el movimiento horario. Si el valor es 2, llama a la subrutina *EDOS* para hacer el movimiento antihorario. Si el valor es 3, se declara un contador igual a 5 y se hace un ciclo donde hasta que el contador valga 0, se llama la subrutina *EUNO* y se decrementa en 1 el contador; de esta forma se hacen 5 giros horario. Por último, si el valor es 4, también se declara un contador con el valor 10 y hasta que el contador valga 0, se llama la subrutina *EDOS* y se

decrementa en 1 el contador; de esta forma se hacen 10 giros antihorario. Acabando cualquier opción regresa a verificar el valor en el Puerto A.



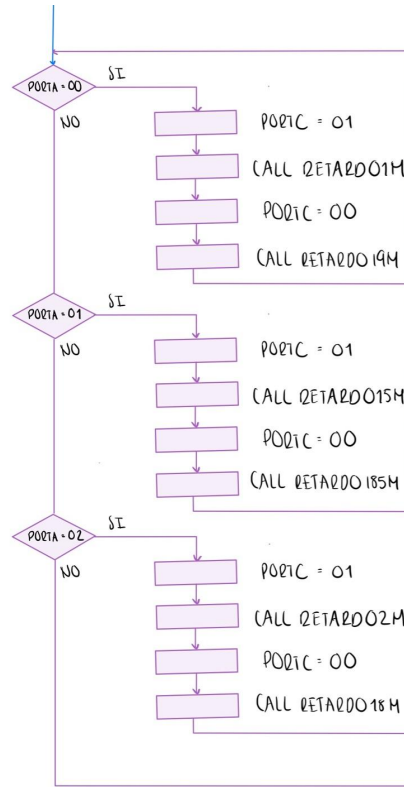
Ejercicio 3

Debido a que este ejercicio utiliza un servomotor que funciona a base de señales PWM, se crearon diferentes subrutinas de *RETARDO* que esperan un número definido de milisegundos. La carta ASM de la subrutina es la siguiente:



Para obtener los distintos intervalos de tiempo el VALOR1 y VALOR2 permanecieron constantes y el VALOR3 fue cambiando dependiendo los milisegundos a esperar. Se realizaron subrutinas para esperar 1 ms, 1.5 ms, 2 ms, 18, ms, 18.5 ms y 19 ms donde el VALOR3 tomó los valores de 05, 0E, 18, 9F, A9 y B2 respectivamente.

La carta ASM del programa es la siguiente:



Después de realizar las configuraciones iniciales, verifica el valor del Puerto A. En caso de ser 0, se activa la señal 1 ms y se apaga 19 ms; esto se logra volviendo 01 el Puerto C, llamando el retardo de 1 ms, volviendo 00 el Puerto C y llamando el retardo de 19 ms. En caso de que Puerto A sea 1 se hace algo parecido pero llamando las subrutinas de retardo de 1.5 y 18.5 ms. Y para el caso donde Puerto A es 2, se realiza lo mismo llamando las subrutinas de 2 y 18 ms. Después de cada opción, el programa regresa a verificar el valor en el Puerto A.

Programas comentados

Ejercicio 1

```

PROCESSOR 16F877
INCLUDE <P16F877.INC>

```

```

;Valores utilizados para generar el retardo
valor1 equ h'21'
valor2 equ h'22'
valor3 equ h'23'
cte1 equ 20h
cte2 equ 50h
cte3 equ 60h

```

```

estado0 equ 0x00 ;Valor entrada estado 0
estado1 equ 0x01 ;Valor entrada estado 1
estado2 equ 0x02 ;Valor entrada estado 2
estado3 equ 0x03 ;Valor entrada estado 3
estado4 equ 0x04 ;Valor entrada estado 4
estado5 equ 0x05 ;Valor entrada estado 5
estado6 equ 0x06 ;Valor entrada estado 6
estado7 equ 0x07 ;Valor entrada estado 7
estado8 equ 0x08 ;Valor entrada estado 8

```

```

ORG 0 ;Vector de reset
GOTO INICIO

```

```

ORG 5
INICIO:
    CLRF PORTA      ; Limpia PORTA
    BSF STATUS,RP0  ; Cambia a banco 1
    BCF STATUS,RP1
    MOVLW 06H       ; Define puertos A y E como digitales
    MOVWF ADCON1
    MOVLW H'3F'     ; Configura puerto A como entrada
    MOVWF TRIA
    CLRF TRISB      ; Configura puerto B como salida
    CLRF TRISC      ; Configura puerto B como salida
    BCF STATUS,RP0  ; Cambia al banco

```

```

LOOP:
    MOVLW estado0
    XORWF PORTA,W
    BTFSC STATUS,Z  ; PORTA == estado0?
    GOTO ECERO      ; if true: Realizar estado 0

    MOVLW estado1
    XORWF PORTA,W
    BTFSC STATUS,Z  ; PORTA == estado1?
    GOTO EUNO       ; if true: Realizar estado 1

    MOVLW estado2
    XORWF PORTA,W
    BTFSC STATUS,Z  ; PORTA == estado2?
    GOTO EDOS       ; if true: Realizar estado 2

    MOVLW estado3
    XORWF PORTA,W
    BTFSC STATUS,Z  ; PORTA == estado3?
    GOTO ETRES      ; if true: Realizar estado 3

    MOVLW estado4
    XORWF PORTA,W
    BTFSC STATUS,Z  ; PORTA == estado4?
    GOTO ECUATRO    ; if true: Realizar estado 4

    MOVLW estado5
    XORWF PORTA,W
    BTFSC STATUS,Z  ; PORTA == estado5?
    GOTO ECINCO     ; if true: Realizar estado 5

    MOVLW estado6
    XORWF PORTA,W
    BTFSC STATUS,Z  ; PORTA == estado6?
    GOTO ESEIS      ; if true: Realizar estado 6

    MOVLW estado7
    XORWF PORTA,W
    BTFSC STATUS,Z  ; PORTA == estado7?
    GOTO ESIETE     ; if true: Realizar estado 7

    MOVLW estado8
    XORWF PORTA,W
    BTFSC STATUS,Z  ; PORTA == estado8?
    GOTO EOCHO      ; if true: Realizar estado 8
    GOTO LOOP       ; Regresar al loop

```

```

;Para M1 = PARO y M2 = PARO
ECERO:
    CLRF PORTB      ; PORTB = 0
    CLRF PORTC      ; PORTC = 0
    CALL RETARDO    ; Se llama el retardo
    GOTO LOOP       ; Se vuelve a leer entrada

```

```

;Para M1 = PARO y M2 = HORARIO
EUNO:
    MOVLW 0X04
    MOVWF PORTC      ; PORTC = 0100
    MOVLW 0X08
    MOVWF PORTB      ; PORTB = 1000
    CALL RETARDO    ; Se llama el retardo
    GOTO LOOP       ; Se vuelve a leer entrada

```

```

;Para M1 = PARO y M2 = ANTI-HORARIO
EDOS:
    MOVLW 0X04
    MOVWF PORTC      ; PORTC = 0100
    MOVWF PORTB      ; PORTB = 0100
    CALL RETARDO    ; Se llama el retardo
    GOTO LOOP       ; Se vuelve a leer entrada

```

```

;Para M1 = HORARIO y M2 = PARO
ETRES:
    MOVLW 0X02
    MOVWF PORTC      ; PORTC = 0010
    MOVWF PORTB      ; PORTB = 0010
    CALL RETARDO    ; Se llama el retardo
    GOTO LOOP       ; Se vuelve a leer entrada

```

```

;Para M1 = ANTI-HORARIO y M2 = PARO
ECUATRO:
    MOVLW 0X02
    MOVWF PORTC      ; PORTC = 0010
    MOVLW 0X01
    MOVWF PORTB      ; PORTB = 0001
    CALL RETARDO    ; Se llama el retardo
    GOTO LOOP       ; Se vuelve a leer entrada

```

```

;Para M1 = HORARIO y M2 = HORARIO
ECINCO:
    MOVLW 0X06
    MOVWF PORTC      ; PORTC = 0110
    MOVLW 0X0A
    MOVWF PORTB      ; PORTB = 1010
    CALL RETARDO    ; Se llama el retardo
    GOTO LOOP       ; Se vuelve a leer entrada

```

```

;Para M1 = ANTI-HORARIO y M2 = ANTI-HORARIO
ESEIS:
    MOVLW 0X06
    MOVWF PORTC      ; PORTC = 0110
    MOVLW 0X05
    MOVWF PORTB      ; PORTB = 0101
    CALL RETARDO    ; Se llama el retardo
    GOTO LOOP       ; Se vuelve a leer entrada

```

```

;Para M1 = HORARIO y M2 = ANTI-HORARIO
ESIETE:
    MOVLW 0X06
    MOVWF PORTC      ; PORTC = 0110
    MOVWF PORTB      ; PORTB = 0110
    CALL RETARDO    ; Se llama el retardo
    GOTO LOOP       ; Se vuelve a leer entrada

```

```

;Para M1 = ANTI-HORARIO y M2 = HORARIO
EOCHO:
    MOVLW 0X06
    MOVWF PORTC      ; PORTC = 0110
    MOVLW 0X09
    MOVWF PORTB      ; PORTB = 1001
    CALL RETARDO    ; Se llama el retardo
    GOTO LOOP       ; Se vuelve a leer entrada

```

;Subrutina para el retardo

```

RETARDO
    MOVLW cte1
    MOVWF valor1

```

```

tres
    MOVLW cte2
    MOVWF valor2

```

```

dos
    MOVLW cte3
    MOVWF valor3

```

```

uno
    DECFSZ valor3
    GOTO uno
    DECFSZ valor2
    GOTO dos
    DECFSZ valor1
    GOTO tres
    RETURN

```


Ejercicio 2

```

PROCESSOR 16F877
INCLUDE <P16F877.INC>

; Valores para el retardo
valor1 equ h'21'
valor2 equ h'22'
valor3 equ h'23'
cte1 equ 0Ah
cte2 equ 0Ah
cte3 equ 4Bh

; Valores para ciclo anidado para una vuelta
valorlen equ h'25'
valorrap equ h'26'
cte40 equ 40h
cte08 equ 08h

estado0 equ 0x00 ;Valor entrada estado 0
estado1 equ 0x01 ;Valor entrada estado 1
estado2 equ 0x02 ;Valor entrada estado 2
estado3 equ 0x03 ;Valor entrada estado 3
estado4 equ 0x04 ;Valor entrada estado 4
CONT equ h'24' ;Contador de vueltas

ORG 0 ;Vector de reset
GOTO INICIO

ORG 5
INICIO:
    CLRF PORTA ; Limpia PORTA
    BSF STATUS,RP0 ; Cambia a banco 1
    BCF STATUS,RP1
    MOVLW 06H ;Define puertos A y E como digitales
    MOVWF ADCON1
    MOVLW H'3F' ;Configura puerto A como entrada
    MOVWF TRISA
    CLRF TRISB ;Configura puerto B como salida
    BCF STATUS,RP0 ;Cambia al banco

LOOP:
    MOVLW estado0
    XORWF PORTA,W
    BTFSC STATUS,Z ;PORTA == estado0?
    GOTO ECERO ;if true: realiza estado 0

    MOVLW estado1
    XORWF PORTA,W
    BTFSC STATUS,Z ;PORTA == estado1?
    CALL EUNO ;if true: realiza estado 1

    MOVLW estado2
    XORWF PORTA,W
    BTFSC STATUS,Z ;PORTA == estado2?
    CALL EDOS ;if true: realiza estado 2

    MOVLW estado3
    XORWF PORTA,W
    BTFSC STATUS,Z ;PORTA == estado3?
    GOTO ETRES ;if true: realiza estado 3

    MOVLW estado4
    XORWF PORTA,W
    BTFSC STATUS,Z ;PORTA == estado4?
    GOTO ECUATRO ;if true: realiza estado 4
    GOTO LOOP ;Checar entrada de nuevo

;Motor en paro
ECERO:
    CLRF PORTB ;La salida es cero
    CALL RETARDO ;Se llama a retardo
    GOTO LOOP ;Checar entrada de nuevo

;Girar en sentido horario
EUNO:
;Se genera un ciclo anidado para llamar al sentido horario varias veces
;Se da una vuelta
    MOVLW cte08
    MOVWF valorlen
ciclolent1
    MOVLW cte40
    MOVWF valorrap
ciclorapl
    CALL HORARIO ;Se llama al sentido horario
    DECFSZ valorrap
    GOTO ciclorapl
    DECFSZ valorlen
    GOTO ciclolent1
    RETURN

;Girar en sentido anti-horario
EDOS:
;Se genera un ciclo anidado para llamar al sentido anti-horario varias veces
;Se da una vuelta
    MOVLW cte08
    MOVWF valorlen
ciclolent2
    MOVLW cte40
    MOVWF valorrap
ciclorap2
    CALL ANTIHORARIO ;Se llama al sentido antihorario
    DECFSZ valorrap
    GOTO ciclorap2
    DECFSZ valorlen
    GOTO ciclolent2
    RETURN

;Dar cinco vueltas en sentido horario
ETRES:
    MOVLW 0X05
    MOVWF CONT ;CONT = 5
LOOP3:
    CALL EUNO ;Dar una vuelta en sentido horario
    DECF CONT ;CONT -= 1
    CLRW
    BTFSS STATUS,Z ;CONT == 0?
    GOTO LOOP3 ;if false: dar otra vuelta
    GOTO LOOP ;if true: checar entrada de nuevo

;Dar diez vueltas en sentido anti-horario
ECUATRO:
    MOVLW 0X0A
    MOVWF CONT ;CONT = 10
LOOP4:
    CALL EDOS ;Dar una vuelta en sentido anti-horario
    DECF CONT ;CONT -= 1
    CLRW
    XORWF CONT,W
    BTFSS STATUS,Z ;CONT == 0?
    GOTO LOOP4 ;if false: dar otra vuelta
    GOTO LOOP ;if true: checar entrada de nuevo

;Subrutina para el sentido horario
HORARIO
    MOVLW 0XC0
    MOVWF PORTB ; PORTB = 1100 0000
    CALL RETARDO; Se llama el retardo
    MOVLW 0X60
    MOVWF PORTB ; PORTB = 0110 0000
    CALL RETARDO; Se llama el retardo
    MOVLW 0X30
    MOVWF PORTB ; PORTB = 0011 0000
    CALL RETARDO; Se llama el retardo
    MOVLW 0X90
    MOVWF PORTB ; PORTB = 1001 0000
    CALL RETARDO; Se llama el retardo
    RETURN

;Subrutina para el sentido antihorario
ANTIHORARIO
    MOVLW 0XC0
    MOVWF PORTB ; PORTB = 1100 0000
    CALL RETARDO; Se llama el retardo
    MOVLW 0X90
    MOVWF PORTB ; PORTB = 1001 0000
    CALL RETARDO; Se llama el retardo
    MOVLW 0X30
    MOVWF PORTB ; PORTB = 0011 0000
    CALL RETARDO; Se llama el retardo
    MOVLW 0X60
    MOVWF PORTB ; PORTB = 0110 0000
    CALL RETARDO; Se llama el retardo
    RETURN

;Subrutina para el retardo
RETARDO
    MOVLW cte1
    MOVWF valor1
tres
    MOVLW cte2
    MOVWF valor2
dos
    MOVLW cte3
    MOVWF valor3
uno
    DECFSZ valor3
    GOTO uno
    DECFSZ valor2
    GOTO dos
    DECFSZ valor1
    GOTO tres
    RETURN

END

```

Ejercicio 3

```

PROCESSOR 16F877
INCLUDE <P16F877.INC>

;Valores y constantes para el retardo
valor1 equ h'21'
valor2 equ h'22'
valor3 equ h'23'
cte1 equ 0Ah
cte2 equ 10h
cte1M equ 05h ;Constante para 1ms
cte1SM equ 0Eh ;Constante para 1.5ms
cte2M equ 18h ;Constante para 2ms
cte19M equ 0XB2 ;Constante para 19ms
cte18SM equ 0XA9 ;Constante para 18.5ms
cte18M equ 0X9F ;Constante para 18ms

estado0 equ 0x01 ;Valor entrada estado 0
estado1 equ 0x02 ;Valor entrada estado 1
estado2 equ 0x04 ;Valor entrada estado 2

ORG 0 ;Vector de reset
GOTO INICIO

ORG 5
INICIO:
    CLRF PORTA ; Limpia PORTA
    BSF STATUS,RP0 ; Cambia a banco 1
    BCF STATUS,RP1
    MOVWL 06H ;Define puertos A y E como digitales
    MOVWF ADCON1
    MOVWL H'3F' ;Configura puerto A como entrada
    MOVWF TRISA
    CLRF TRISC ;Configura el puerto C como salida
    BCF STATUS,RP0 ;Cambia al banco

LOOP:
    MOVWL estado0
    XORWF PORTA,W
    BTFSC STATUS,Z ;PORTA == estado0?
    GOTO ECERO ;if true: realizar estado 0

    MOVWL estado1
    XORWF PORTA,W
    BTFSC STATUS,Z ;PORTA == estado1?
    GOTO EUNO ;if true: realizar estado 1

    MOVWL estado2
    XORWF PORTA,W
    BTFSC STATUS,Z ;PORTA == estado2?
    GOTO EDOS ;if true: realizar estado 2
    GOTO LOOP ;Checar la entrada de nuevo

;Rotación hacia la izquierda (0°)
ECERO:
    MOVWL 0X01
    MOVWF PORTC ;PORTC = 1
    CALL RETARDO1M ;1ms en uno
    CLRF PORTC ;PORTC = 0
    CALL RETARDO19M ;19ms en cero
    GOTO LOOP ;Checar la entrada de nuevo

;Rotación central (90°)
EUNO:
    MOVWL 0X01
    MOVWF PORTC ;PORTC = 1
    CALL RETARDO15M ;1.5ms en uno
    CLRF PORTC ;PORTC = 0
    CALL RETARDO18SM ;18.5ms en cero
    GOTO LOOP ;Checar la entrada de nuevo

;Rotación hacia la derecha (180°)
EDOS:
    MOVWL 0X01
    MOVWF PORTC ;PORTC = 1
    CALL RETARDO2M ;2ms en uno
    CLRF PORTC ;PORTC = 0
    CALL RETARDO18M ;18ms en cero
    GOTO LOOP ;Checar la entrada de nuevo

;Retardo para 1ms
RETARDO1M
    MOVWL cte1
    MOVWF valor1
tres1M
    MOVWL cte2
    MOVWF valor2
dos1M
    MOVWL cte1SM
    MOVWF valor3
uno1M
    DECFSZ valor3
    GOTO uno1M
    DECFSZ valor2
    GOTO dos1M
    DECFSZ valor1
    GOTO tres1M
    RETURN

;Retardo para 1.5ms
RETARDO15M
    MOVWL cte1
    MOVWF valor1
tres15M
    MOVWL cte2
    MOVWF valor2
dos15M
    MOVWL cte1SM
    MOVWF valor3
uno15M
    DECFSZ valor3
    GOTO uno15M
    DECFSZ valor2
    GOTO dos15M
    DECFSZ valor1
    GOTO tres15M
    RETURN

;Retardo para 2ms
RETARDO2M
    MOVWL cte1
    MOVWF valor1
tres2M
    MOVWL cte2
    MOVWF valor2
dos2M
    MOVWL cte2M
    MOVWF valor3
uno2M
    DECFSZ valor3
    GOTO uno2M
    DECFSZ valor2
    GOTO dos2M
    DECFSZ valor1
    GOTO tres2M
    RETURN

;Retardo para 19ms
RETARDO19M
    MOVWL cte1
    MOVWF valor1
tres19M
    MOVWL cte2
    MOVWF valor2
dos19M
    MOVWL cte19M
    MOVWF valor3
uno19M
    DECFSZ valor3
    GOTO uno19M
    DECFSZ valor2
    GOTO dos19M
    DECFSZ valor1
    GOTO tres19M
    RETURN

;Retardo para 18.5ms
RETARDO18SM
    MOVWL cte1
    MOVWF valor1
tres18SM
    MOVWL cte2
    MOVWF valor2
dos18SM
    MOVWL cte18SM
    MOVWF valor3
uno18SM
    DECFSZ valor3
    GOTO uno18SM
    DECFSZ valor2
    GOTO dos18SM
    DECFSZ valor1
    GOTO tres18SM
    RETURN

;Retardo para 18ms
RETARDO18M
    MOVWL cte1
    MOVWF valor1
tres18M
    MOVWL cte2
    MOVWF valor2
dos18M
    MOVWL cte18M
    MOVWF valor3
uno18M
    DECFSZ valor3
    GOTO uno18M
    DECFSZ valor2
    GOTO dos18M
    DECFSZ valor1
    GOTO tres18M
    RETURN

END

```

Conclusiones y/o comentarios

Zepeda Baeza Jessica:

En esta práctica pudimos usar los puertos paralelos B y C como salidas para controlar distintos motores y el puerto A como entrada para elegir el movimiento de estos motores. En este caso, se nos proporcionó un circuito que ya tenía las terminales de los motores de corriente directa, un motor a pasos y un servomotor, conectadas a distintos puertos de la PIC16F877. A través de los distintos programas se observó que un mismo puerto puede estar conectado a dos salidas diferentes y al modificar ciertos bits del puerto se produce, en este caso, el movimiento deseado de un motor específico. Los tres programas realizados generaban un movimiento de acuerdo a la entrada por lo que su construcción fue muy similar utilizando XOR y BTFSS para comparar la entrada con las posibles opciones. Lo que cambió en cada uno de ellos fue el número de opciones y lo que se realizaba en cada una, aunque todas ellas terminaban con la asignación de un valor específico al Puerto B y/o C. Para cada ejercicio fue necesario entender cómo funcionaba cada motor (por corriente, por bobinas, por PWM) y qué efecto tenían las terminales en su funcionamiento; de esta forma se pudieron hacer tablas para obtener los valores en hexadecimal necesarios o calcular el tiempo activo de una señal.

Barreiro Valdez Alejandro:

En esta práctica se pudo entender cómo se programa el movimiento de tres tipos de motores utilizando el microcontrolador PIC16F877. Se pudo realizar el movimiento de un motor de bobinas, de un servomotor y de un motor de corriente directa. Para ello se tuvo que identificar qué puertos estaban actuando como salida en el sistema que se proporcionó en el laboratorio. Posteriormente, para el motor de corriente directa y para el motor a pasos se realizó una tabla donde se puso qué bits de cada puerto se debe prender y cuáles se deben de apagar para cada uno de los estados que se pidieron. Con estas tablas se pudo programar cada uno de los estados que se pedían. Además, otro elemento importante para la realización de esta práctica fue el cálculo de los tiempos de retardo. Para esto se realizó una fórmula donde se pudo calcular los ciclos de reloj que toma cada una de las instrucciones y con la frecuencia del reloj se pudo calcular el tiempo de cada retardo. Esto fue especialmente útil para el servomotor donde se tuvo que enviar una señal en prendido y apagado por cierto tiempo para controlar el movimiento del motor. Con todos estos conceptos se logró realizar el control de cada uno de los motores y generar cada uno de los estados que se pidieron.