**Facultad de ingeniería**

**Materia:** Laboratorio de Microcomputadoras

**REPORTE DE LA PRÁCTICA 5**

**Título: Control de actuadores**

**Integrantes:**


- Martínez Pérez Brian Erik - 319049792
- Nuñez Rodas Abraham Enrique - 114003546
- Vicenteño Maldonado Jennifer Michel - 317207251


**Profesor:** Moises Melendez Reyes

**Grupo:** 1

**Fecha de Entrega:** 6 de abril de 2025

**Semestre:** 2025-2

**Objetivo**: Emplear los puertos paralelos que contiene un microcontrolador, para controlar la operación de dos motores de corriente directa, motores a pasos y servomotores.

## Ejercicios

**Ejercicio 1: Considerando la asignación de terminales asignadas en la figura 5.1; realizar el programa que ejecute el control indicado en la tabla 5.1.**
**Nota: Las tierras de los ambos circuitos están conectadas entre sí.**

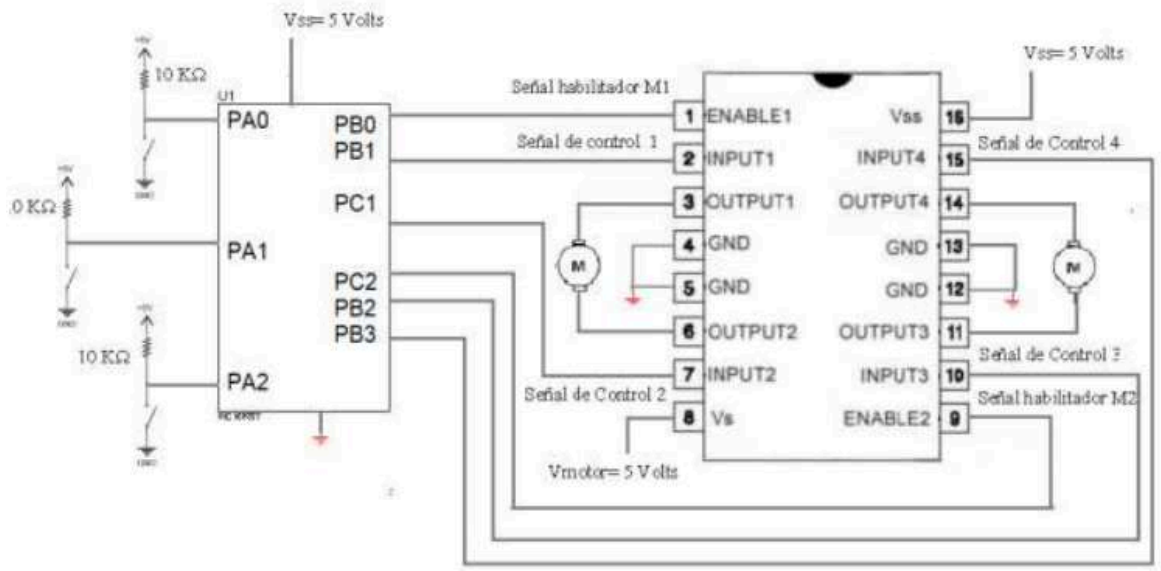



Figura 5.10 Circuito control de motores de CD

**La asignación de las terminales queda de la siguiente manera:**

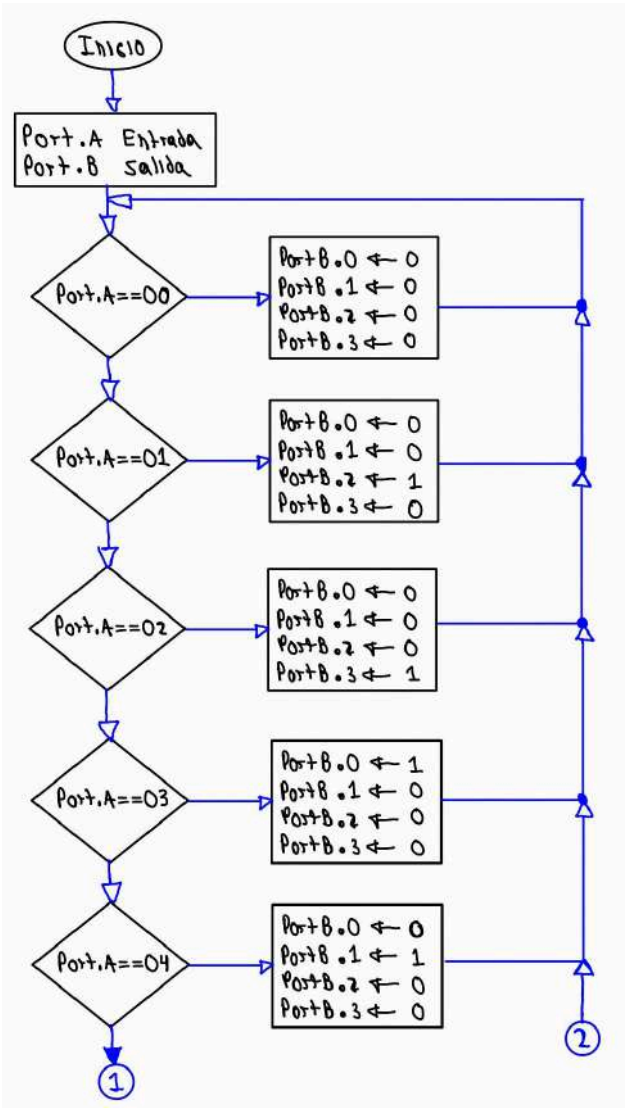
| MOTOR2 | | |
|---|---|---|
| **PC2** | **PB3** | **PB2** |
| ENABLE_M2 | DIR1_M2 | DIR2_M2 |

| MOTOR1 | | |
|---|---|---|
| **PC1** | **PB1** | **PB0** |
| ENABLE_M1 | DIR1_M1 | DIR2_M1 |

| **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 |

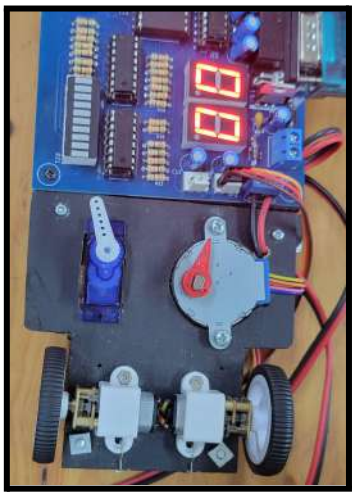
Tabla 5.1 Operación de motores de corriente directa

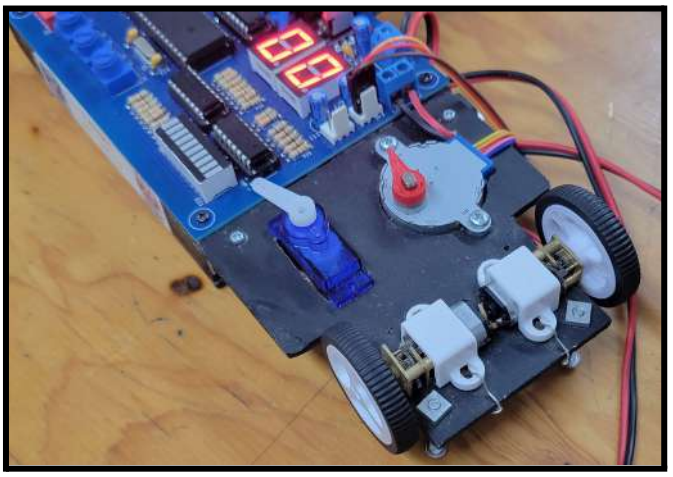
Diagrama de flujo:

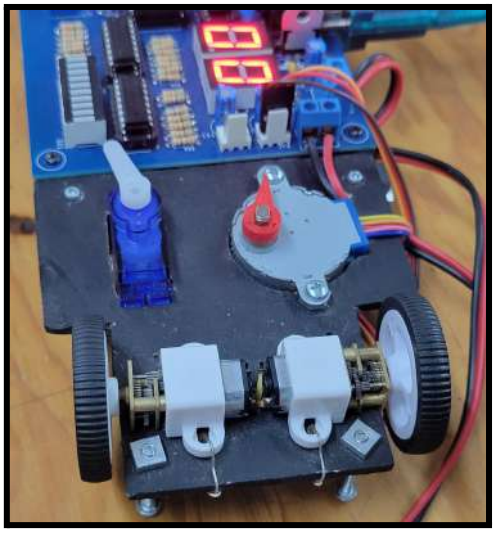


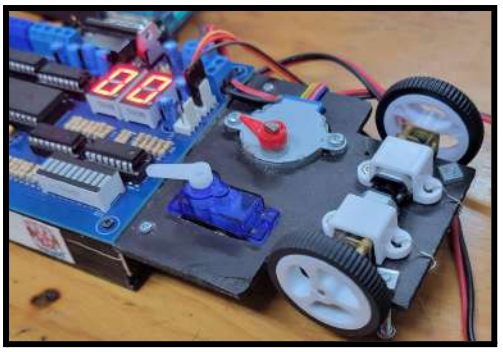

Funcionamiento de la solución:

imágenes:









Código:

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

M1A     equ   0
M1B     equ   1
M2A     equ   2
M2B     equ   3

        ORG   0x0000
        GOTO  INICIO

        ORG   0x0004
        RETFIE

INICIO:
        BSF   STATUS, RP0
        BCF   STATUS, RP1

        CLRF  TRISB
        MOVLW 0xFF
        MOVWF TRISA
        BCF   STATUS, RP0
        CLRF  PORTB

        BSF   STATUS, RP0
        BCF   STATUS, RP1
        MOVLW 0x06
        MOVWF ADCON1
        BCF   STATUS, RP0

MAIN:
        MOVF  PORTA, W
        ANDLW 0x0F
        ADDWF PCL, F

        GOTO  ACCION0
        GOTO  ACCION1
        GOTO  ACCION2
        GOTO  ACCION3
        GOTO  ACCION4
        GOTO  ACCION5
```

```
        GOTO    ACCION6
        GOTO    ACCION7
        GOTO    ACCION8
        GOTO    MAIN
        GOTO    MAIN
        GOTO    MAIN
        GOTO    MAIN
        GOTO    MAIN
        GOTO    MAIN
        GOTO    MAIN

ACCION0:
        BCF     PORTB, M1A
        BCF     PORTB, M1B
        BCF     PORTB, M2A
        BCF     PORTB, M2B
        GOTO    MAIN

ACCION1:
        BCF     PORTB, M1A
        BCF     PORTB, M1B
        BSF     PORTB, M2A
        BCF     PORTB, M2B
        GOTO    MAIN

ACCION2:
        BCF     PORTB, M1A
        BCF     PORTB, M1B
        BCF     PORTB, M2A
        BSF     PORTB, M2B
        GOTO    MAIN

ACCION3:
        BSF     PORTB, M1A
        BCF     PORTB, M1B
        BCF     PORTB, M2A
        BCF     PORTB, M2B
        GOTO    MAIN

ACCION4:
        BCF     PORTB, M1A
        BSF     PORTB, M1B
        BCF     PORTB, M2A
        BCF     PORTB, M2B
        GOTO    MAIN

ACCION5:
        BSF     PORTB, M1A
        BCF     PORTB, M1B
        BSF     PORTB, M2A
        BCF     PORTB, M2B
        GOTO    MAIN

ACCION6:
        BCF     PORTB, M1A
        BSF     PORTB, M1B
        BSF     PORTB, M2A
        BCF     PORTB, M2B
        GOTO    MAIN

ACCION7:
        BSF     PORTB, M1A
        BCF     PORTB, M1B
        BCF     PORTB, M2A
        BSF     PORTB, M2B
        GOTO    MAIN

ACCION8:
        BCF     PORTB, M1A
        BSF     PORTB, M1B
        BCF     PORTB, M2A
        BSF     PORTB, M2B
        GOTO    MAIN

RUTINA_STEPPER:
        RETURN

RUTINA_SERVO:
        RETURN

RETARDO:
        RETURN

        END
```

Análisis: La solución propuesta utiliza una tabla de saltos condicional para que cada combinación del DIP switch (PORTA) dirija el flujo del programa a una acción específica (ACCION0 a ACCION8). PORTA se coloca como entrada para leer el DIP switch y PORTB como salida para controlar los motores DC (M1 y M2).

Cada acción configura los pines de PORTB que controlan los motores DC, activando una dirección de giro u otra, o deteniéndolos. Dependiendo del valor leído,

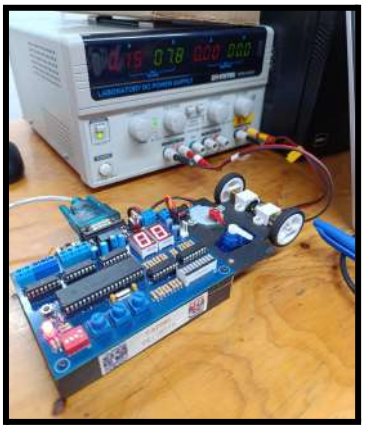
se activa una rutina ACCION, que modifica los bits correspondientes de PORTB para controlar los motores M1 y M2. Después de cada acción, el programa vuelve a MAIN y repite el proceso de lectura, evaluación y acción.

Los modos de direccionamiento usados son, Direccionamiento inmediato: MOVLW 0xFF, Se carga un valor literal directo al registro de trabajo (WREG). Direccionamiento directo: MOVWF TRISA, Se transfiere el contenido de W a una dirección específica del registro. Direccionamiento indirecto al contador de programa (PCL): ADDWF PCL, F, Permite redirigir el flujo de ejecución a una instrucción dependiendo del valor en W.

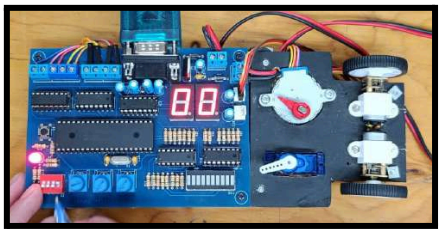
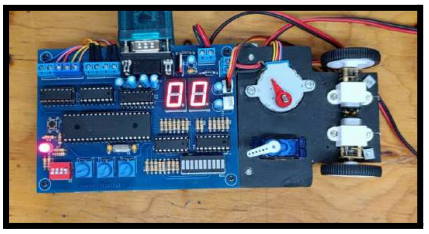
**Ejercicio 2: Realizar un programa que controle la cantidad de pasos que debe dar un motor, así como el sentido de giro.**

Funcionamiento de la solución:

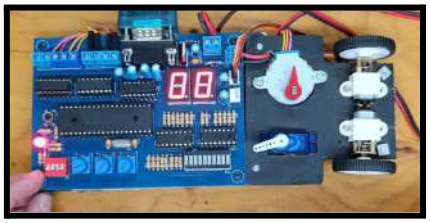
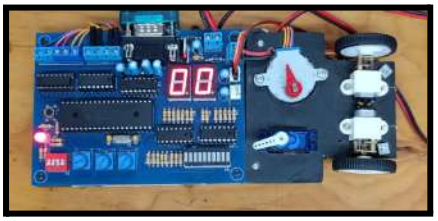
imágenes:



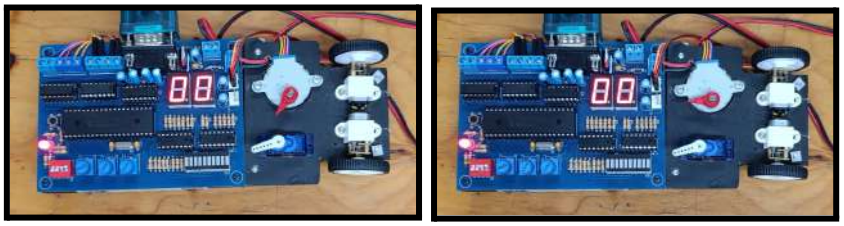
conexiones

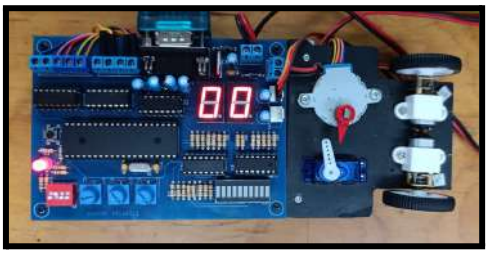
0x01 sentido horario

0x02 sentido anti-horario



0x03 vuelta y media



0x04 5 vueltas sentido antihorario

Código:

```
PROCESSOR 16f877a
INCLUDE <p16f877a.inc>

; Definición de estados del motor a pasos
EDO1    EQU 0XC0
EDO2    EQU 0X60
EDO3    EQU 0X30
EDO4    EQU 0X90

; Variables para contadores de pasos
CONTADOR1 EQU 0X22
CONTADOR2 EQU 0X23

; Variables para retardos
valor1 EQU h'10'
valor2 EQU h'11'
valor3 EQU h'12'

ORG 0
GOTO INICIO
ORG 0X05

INICIO:
   CALL CONFIG_INICIAL

LOOP:
   MOVF PORTA, W
   ANDLW 0x0F
   MOVWF valor3
   CALL OPCIONES
   GOTO LOOP

OPCIONES:
   MOVF valor3, W
   SUBLW 0x00
   BTFSC STATUS,Z
   GOTO PARO
   MOVF valor3, W
   SUBLW 0x01
   BTFSC STATUS,Z
   GOTO HORARIO
   MOVF valor3, W
   SUBLW 0x02
   BTFSC STATUS,Z
   GOTO ANTIHORARIO
   MOVF valor3, W
   SUBLW 0x03
   BTFSC STATUS,Z
   GOTO UNO_Y_MEDIA
   MOVF valor3, W
   SUBLW 0x04
   BTFSC STATUS,Z
   GOTO DOS_ANTIHOR
   RETURN

PARO:
   CLRF PORTB
   CALL RETARDO_18ms
   RETURN

HORARIO:
   MOVLW 0x01
   XORWF PORTA, W
   BTFSS STATUS, Z
   RETURN

HOR_LOOP:
   MOVLW EDO1
   MOVWF PORTB
   CALL RETARDO_05ms
   MOVLW EDO2
   MOVWF PORTB
   CALL RETARDO_05ms
   MOVLW EDO3
```

```
    MOVWF PORTB
    CALL RETARDO_05ms
    MOVLW EDO4
    MOVWF PORTB
    CALL RETARDO_05ms
    GOTO HORARIO

ANTIHORARIO:
    MOVLW 0x02
    XORWF PORTA, W
    BTFSS STATUS, Z
    RETURN

ANTIHOR_LOOP:
    MOVLW EDO4
    MOVWF PORTB
    CALL RETARDO_05ms
    MOVLW EDO3
    MOVWF PORTB
    CALL RETARDO_05ms
    MOVLW EDO2
    MOVWF PORTB
    CALL RETARDO_05ms
    MOVLW EDO1
    MOVWF PORTB
    CALL RETARDO_05ms
    GOTO ANTIHORARIO

UNO_Y_MEDIA:
    MOVLW 0xFF
    MOVWF CONTADOR1
    CLRF CONTADOR2

GIRO_1M:
    MOVLW 0x09      ; Umbral cambiado de
0x03 a 0x09 para obtener 1.5 vueltas
    XORWF CONTADOR2, W
    BTFSC STATUS, Z
    GOTO PARO

    DECF CONTADOR1
    BTFSS STATUS, Z
    GOTO PASO_1M

    MOVLW 0xFF
    MOVWF CONTADOR1
    INCF CONTADOR2

PASO_1M:
    MOVLW EDO1
    MOVWF PORTB
    CALL RETARDO_05ms
    MOVLW EDO2
    MOVWF PORTB
    CALL RETARDO_05ms
    MOVLW EDO3
    MOVWF PORTB
    CALL RETARDO_05ms
    MOVLW EDO4
    MOVWF PORTB
    CALL RETARDO_05ms
    GOTO GIRO_1M

DOS_ANTIHOR:
    MOVLW 0xFF
    MOVWF CONTADOR1
    CLRF CONTADOR2

GIRO_2A:
    MOVLW 0x04
    XORWF CONTADOR2, W
    BTFSC STATUS, Z
    GOTO PARO

    DECF CONTADOR1
    BTFSS STATUS, Z
    GOTO PASO_2A

    MOVLW 0xFF
    MOVWF CONTADOR1
    INCF CONTADOR2

PASO_2A:
    MOVLW EDO4
    MOVWF PORTB
    CALL RETARDO_05ms
    MOVLW EDO3
    MOVWF PORTB
    CALL RETARDO_05ms
    MOVLW EDO2
    MOVWF PORTB
    CALL RETARDO_05ms
    MOVLW EDO1
    MOVWF PORTB
    CALL RETARDO_05ms
    GOTO GIRO_2A

CONFIG_INICIAL:
    BCF STATUS, RP1
    BSF STATUS, RP0
    CLRF TRISB
    CLRF TRISC
    MOVLW 0x06
    MOVWF ADCON1
    MOVLW 0x3F
    MOVWF TRISA
    BCF STATUS, RP0
    CLRF PORTB
    CLRF PORTA
    CLRF PORTC
    RETURN

RETARDO_05ms:
    MOVWF valor1
```

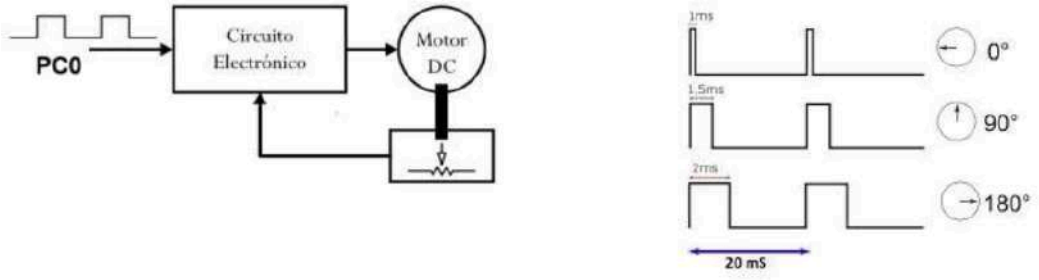
```
DOS_05:
   MOVWF valor2
UNO_05:
   DECFSZ valor2
   GOTO UNO_05
   DECFSZ valor1
   GOTO DOS_05
   RETURN

RETARDO_18ms:
   MOVWF valor1
DOS_18:
   MOVWF valor2
UNO_18:
   DECFSZ valor2
   GOTO UNO_18
   DECFSZ valor1
   GOTO DOS_18
   RETURN

END
```

Análisis:

Se configura a el puerto A como entrada del switch, al puerto B como salida para mandar las señales al motor a pasos , se utiliza ADCON1 para pasar de analógico a digital. En el loop se lee lo que manda el switch (4 bits) y se almacenan en valor3, después pasa a opciones, compara el valor3 y ejecuta la acción correspondiente, este procesos lo repite indefinidamente, por lo que es importante colocar el resultado y después dejarlo en 0 si deseamos que este se detenga al dar el número de vueltas correspondiente en el 0x03 y 0x04.

En caso de estar en 0x00 se mantiene en paro, trabaja a través de secuencias, contadores y retardos.

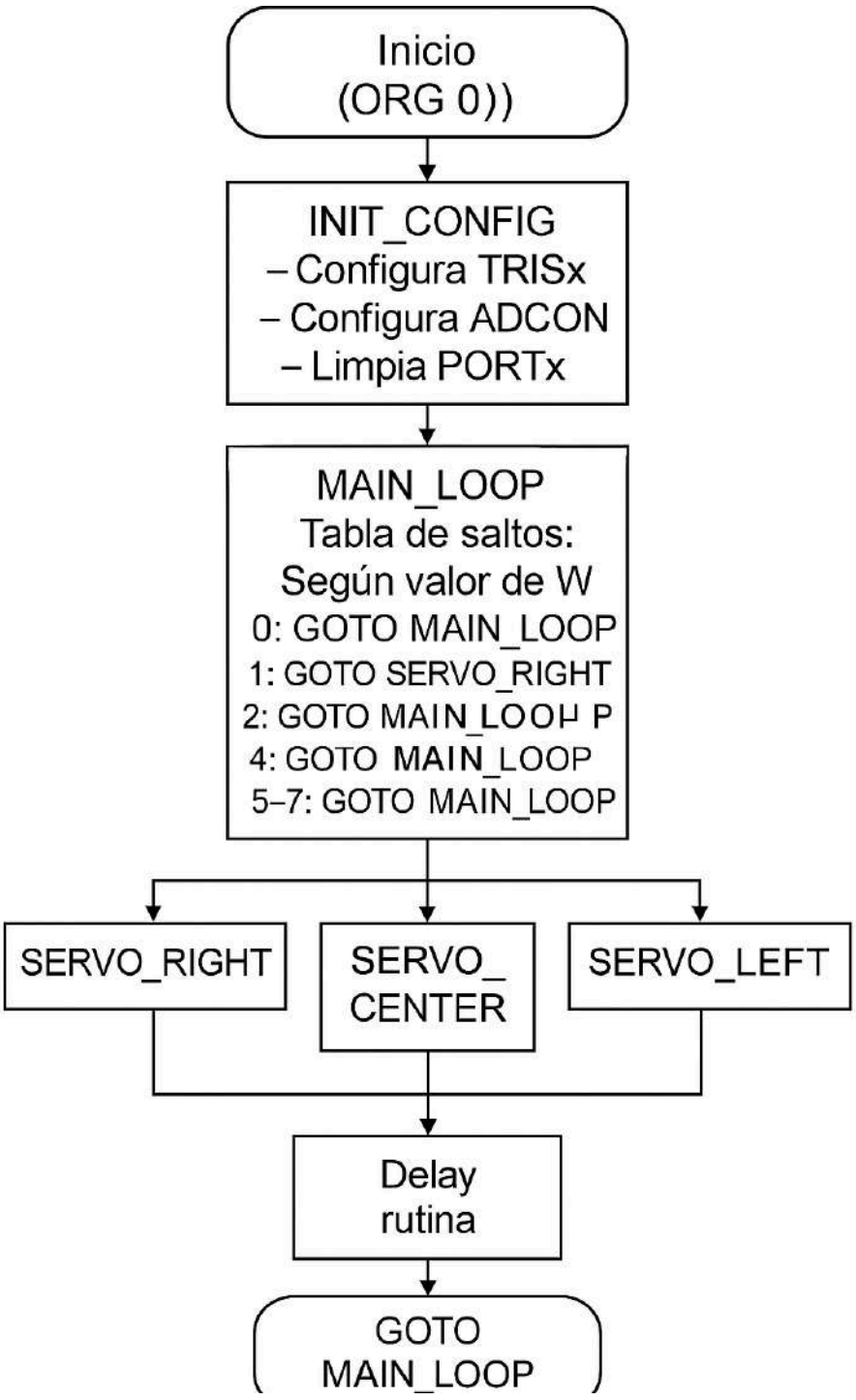
**Ejercicio 3: Utilizando un servo motor realizar el control mostrado en la tabla No. 5.3**



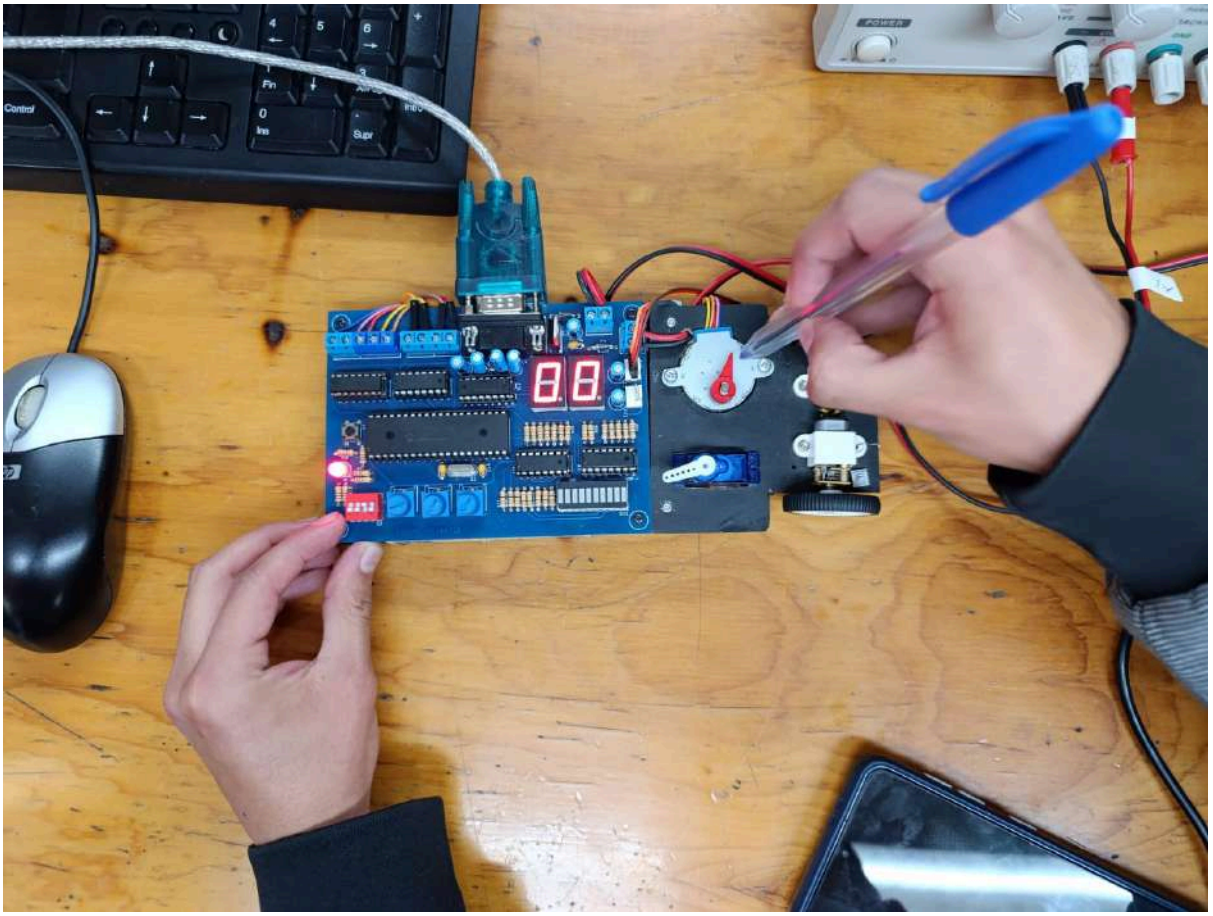

| SW2 | SW1 | SW0 | Posición Servo | Representación |
|---|---|---|---|---|
| 1 | 0 | 0 | Izquierda | 0° |
| 0 | 1 | 0 | Central | 90° |
| 0 | 0 | 1 | Derecha | 180° |

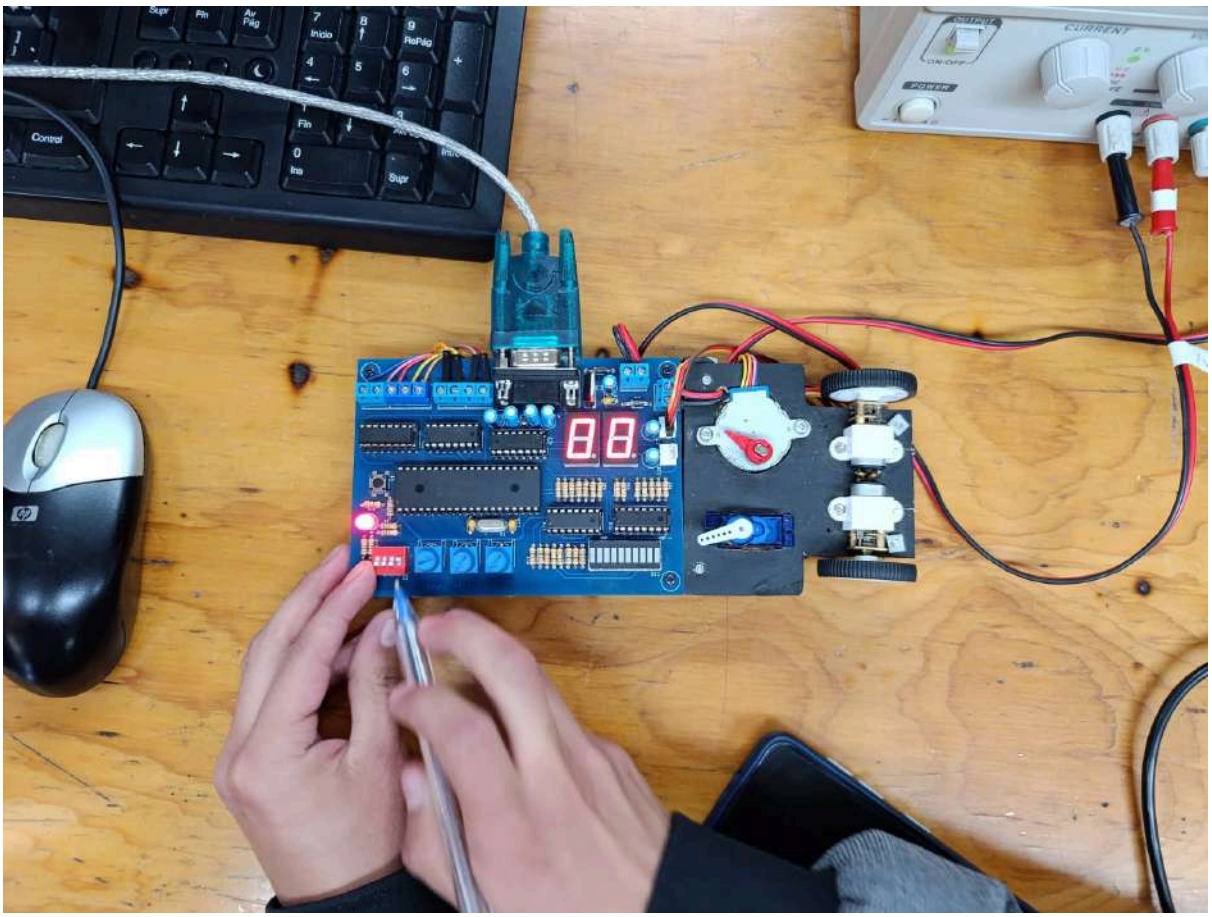
Tabla 5.3 Funcionamiento del servo motor

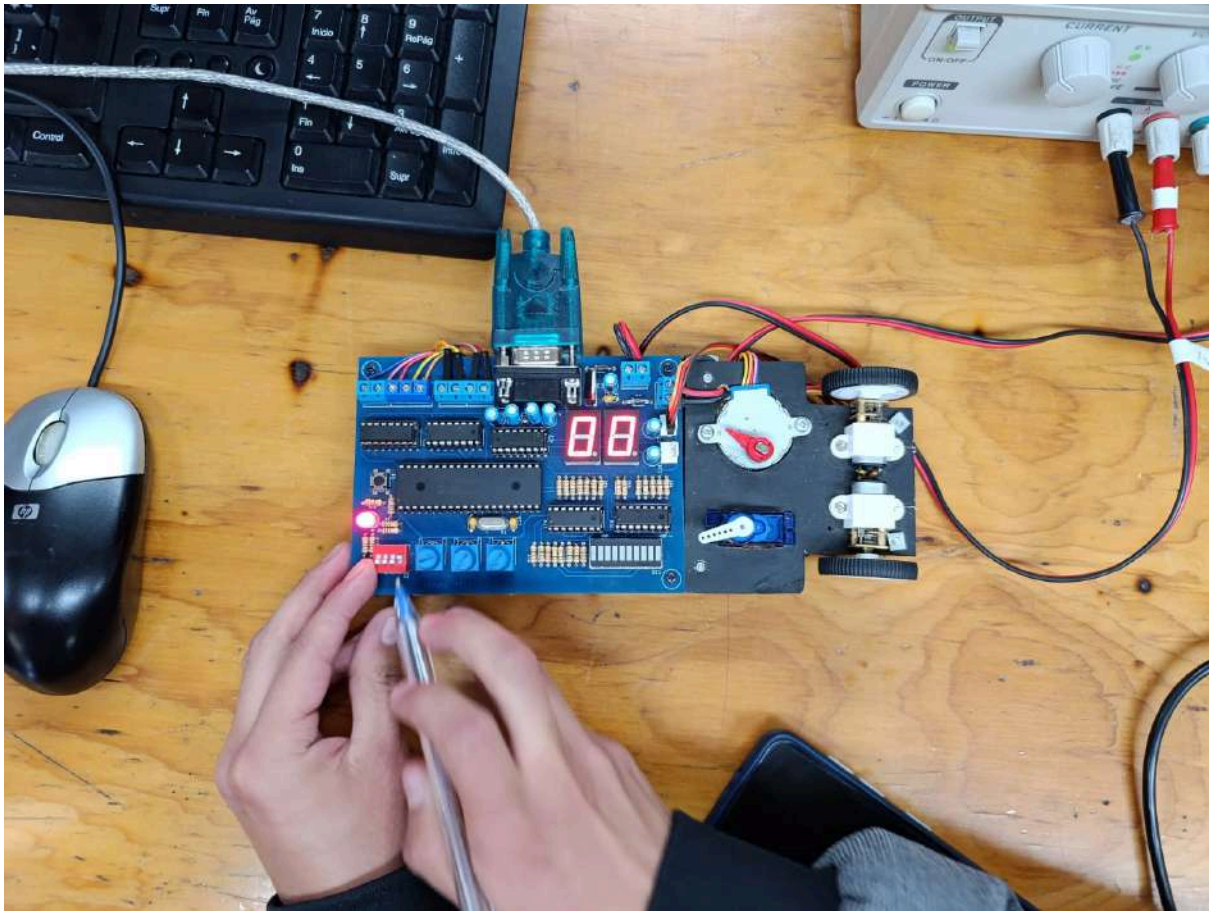
Diagrama de flujo:

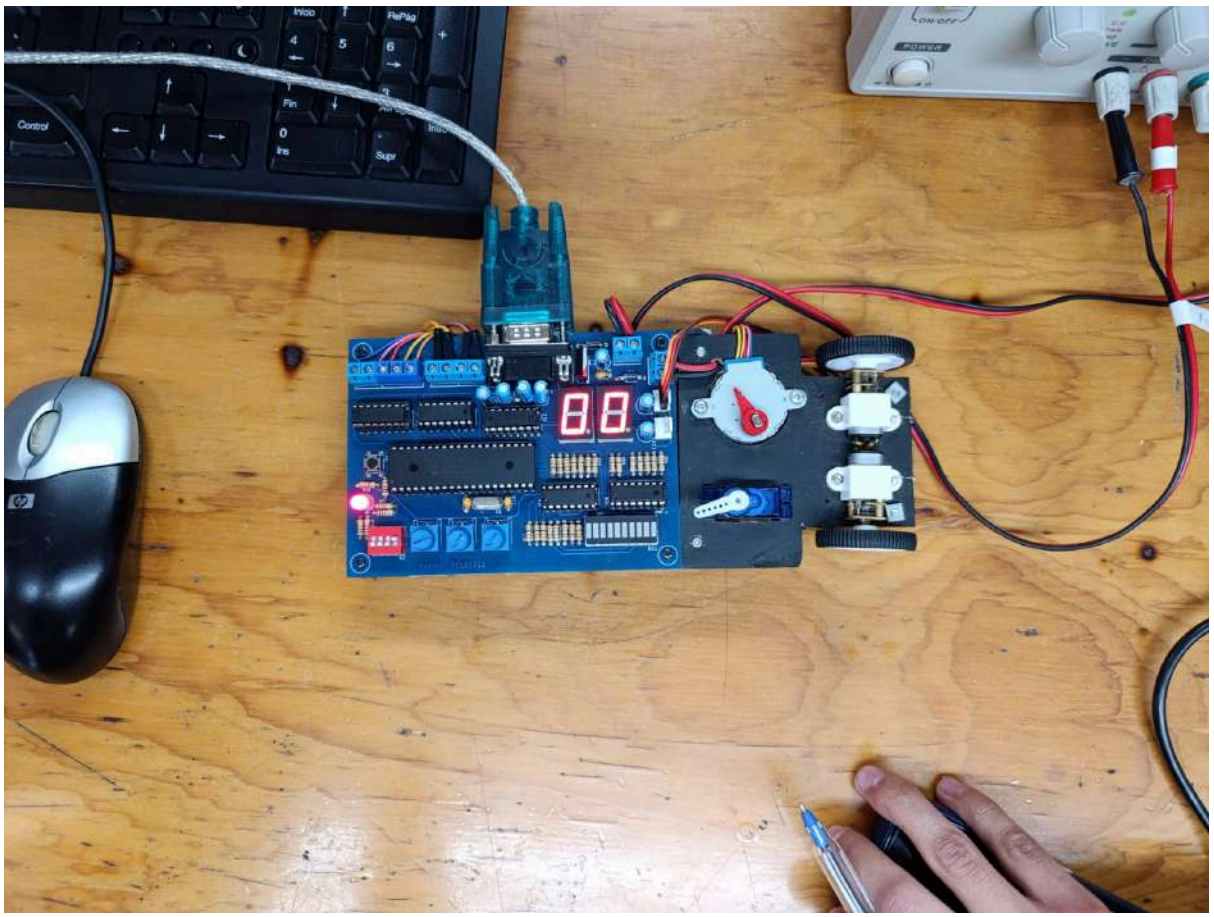


imágenes:









Código:

```
PROCESSOR 16f877a
INCLUDE <p16f877a.inc>                    DELAY_VAR1 equ h'41'
```

```
DELAY_VAR2 equ h'42'
DELAY_VAR3 equ h'43'

   ORG 0
   GOTO MAIN
   ORG 5

MAIN:
   CALL INIT_CONFIG

MAIN_LOOP:
   MOVLW   0x07
   ANDWF   PORTA, W
   ADDWF   PCL, F

   GOTO    MAIN_LOOP     ; 0
   GOTO    SERVO_RIGHT   ; 1
   GOTO    SERVO_CENTER  ; 2
   GOTO    MAIN_LOOP     ; 3
   GOTO    SERVO_LEFT    ; 4
   GOTO    MAIN_LOOP     ; 5
   GOTO    MAIN_LOOP     ; 6
   GOTO    MAIN_LOOP     ; 7

SERVO_RIGHT:
   MOVLW   0xFF
   MOVWF   PORTC
   CALL    DELAY_2ms
   CALL    DELAY_05ms
   CLRF    PORTC
   CALL    DELAY_18ms
   GOTO    MAIN_LOOP

SERVO_CENTER:
   MOVLW   0xFF
   MOVWF   PORTC
   CALL    DELAY_1ms
   CALL    DELAY_05ms
   CLRF    PORTC
   CALL    DELAY_05ms
   CALL    DELAY_18ms
   GOTO    MAIN_LOOP

SERVO_LEFT:
   MOVLW   0xFF
   MOVWF   PORTC
   CALL    DELAY_05ms
   CLRF    PORTC
   CALL    DELAY_1ms
   CALL    DELAY_05ms
   CALL    DELAY_18ms
   GOTO    MAIN_LOOP

INIT_CONFIG:
   BCF     STATUS, RP1
   BSF     STATUS, RP0
   CLRF    TRISB
   CLRF    TRISC
   MOVLW   0x06
   MOVWF   ADCON1
   MOVLW   0x3F
   MOVWF   TRISA
   BCF     STATUS, RP0
   CLRF    PORTB
   CLRF    PORTA
   CLRF    PORTC
   RETURN

DELAY_18ms:
   MOVLW   .255
   MOVWF   DELAY_VAR1

DELAY_18ms_LOOP1:
   MOVLW   .118
   MOVWF   DELAY_VAR2

DELAY_18ms_LOOP2:
   DECFSZ  DELAY_VAR2
   GOTO    DELAY_18ms_LOOP2
   DECFSZ  DELAY_VAR1
   GOTO    DELAY_18ms_LOOP1
   RETURN

DELAY_1ms:
   MOVLW   .150
   MOVWF   DELAY_VAR1

DELAY_1ms_LOOP1:
   MOVLW   .11
   MOVWF   DELAY_VAR2

DELAY_1ms_LOOP2:
   DECFSZ  DELAY_VAR2
   GOTO    DELAY_1ms_LOOP2
```

```
    DECFSZ  DELAY_VAR1
    GOTO    DELAY_1ms_LOOP1
    RETURN

DELAY_2ms:
    MOVLW   .255
    MOVWF   DELAY_VAR1

DELAY_2ms_LOOP1:
    MOVLW   .12
    MOVWF   DELAY_VAR2

DELAY_2ms_LOOP2:
    DECFSZ  DELAY_VAR2
    GOTO    DELAY_2ms_LOOP2
    DECFSZ  DELAY_VAR1
    GOTO    DELAY_2ms_LOOP1
    RETURN

DELAY_05ms:
    MOVLW   .82
    MOVWF   DELAY_VAR1

DELAY_05ms_LOOP1:
    MOVLW   .10
    MOVWF   DELAY_VAR2

DELAY_05ms_LOOP2:
    DECFSZ  DELAY_VAR2
    GOTO    DELAY_05ms_LOOP2
    DECFSZ  DELAY_VAR1
    GOTO    DELAY_05ms_LOOP1
    RETURN

    END

loop_stay MACRO valuein,
compare_with, ret_loop, inner_loop
    MOVLW   valuein
    XORWF   compare_with, W
    BTFSS   STATUS, Z
    GOTO    ret_loop
    GOTO    inner_loop
ENDM
```

Análisis:

En este ejercicio se implementó el control de un **servomotor** utilizando señales PWM simuladas por software mediante retardos. La lógica de control parte de leer el valor del puerto `PORTA`, utilizando únicamente los tres bits menos significativos, que se comparan para seleccionar una de las posibles posiciones del servomotor: derecha, centro o izquierda. A través de una tabla de saltos (`ADDWF PCL, F`), el programa redirige la ejecución a una rutina específica según el valor de entrada.

Cada rutina de movimiento (SERVO_RIGHT, SERVO_CENTER, SERVO_LEFT) genera un **pulso de duración variable** en el pin de control del servomotor conectado al `PORTC`, que simula una señal PWM. La duración de este pulso determina el ángulo al que se moverá el servo:

- **Derecha**: Pulso de ~2 ms
- **Centro**: Pulso de ~1.5 ms
- **Izquierda**: Pulso de ~1 ms

Después de enviar el pulso correspondiente, se coloca en bajo la señal del servo (CLRF PORTC), y se genera un retardo de aproximadamente 18 ms para completar

el periodo de la señal (aproximadamente 20 ms), cumpliendo con el requerimiento típico de señales PWM para servomotores.

Este método, aunque no usa módulos CCP del PIC, demuestra que es posible generar señales PWM básicas utilizando solo retardos y control de pines, lo cual es útil para comprender cómo se comporta un servomotor a bajo nivel.

Además, se observa un correcto uso del direccionamiento inmediato, directo y relativo a PCL, lo cual permite un flujo de control dinámico y estructurado sin estructuras de control complejas.

## Conclusiones:

**Martínez Pérez Brian Erik**
Para esta práctica se cumplió con el objetivo de configurar los puertos paralelos del PIC ya sea como entrada, recibir valores o colocarlos como salida, en este se presenta un movimiento en distintos motores. dependiendo el tipo de motor se especificaba la entrada y el tipo de movimiento. para el motor de DC dependía de las revoluciones y el sentido de giro que enviamos, el motor a pasos depende de la señal de PWM que indicaba el sentido de giro y la cantidad de vueltas que daba. para el último motor, el cual era un servomotor, se especificaba el sentido de giro y el ángulo para donde giraba.

**Nuñez Rodas Abraham Enrique**

Durante esta práctica reforcé mis conocimientos sobre el manejo de puertos paralelos en el microcontrolador PIC16F877A para controlar distintos tipos de actuadores. Me resultó interesante observar cómo una correcta configuración de registros como `TRIS` y `ADCON1` permite transformar entradas analógicas en señales digitales útiles para el control de motores. Además, entendí la importancia del direccionamiento y el uso de retardos precisos en el funcionamiento de motores DC, motores a pasos y servomotores. Implementar cada rutina de control me ayudó a visualizar el impacto de las instrucciones ensamblador en el comportamiento físico de los dispositivos, y me permitió comprender cómo pequeñas variaciones en el código generan diferentes respuestas en los actuadores. Esta práctica fue clave para consolidar la relación entre programación a bajo nivel y la manipulación de hardware real.

**Vicenteño Maldonado Jennifer Michel**
Durante esta práctica logramos controlar diferentes dispositivos como el motor de corriente directa, motor a pasos y servomotor con ayuda de los puertos paralelos del microcontrolador, logramos configurar las entradas análogicas como salidas digitales, creamos rutinas para los diferentes comportamientos de los dispositivos, así como usar retardos para hacer más evidentes.
Utilizamos registros específicos como TRIS y ADCON1.

## Referencias:

- del PIC, 2. 1-La Familia. (s/f). 2.- Descripción General del PIC16F877. Edu.ar., de https://exa.unne.edu.ar/ingenieria/sysistemas/public_html/Archi_pdf/HojaDatos/Microcontrolador es/PIC16F877.pdf
- (S/f). Newark.com. Recuperado de https://mexico.newark.com/microchip/pic16f877a-i-p/microcontroller-mcu-8-bit pic16/dp/69K7640?srsltid=AfmBOorWLTceQMTppGk0OMGmmjB6UpIiiw55U 28F2qBZH2pUYET 3EInu
- *Descripción General del PIC16F877 1 2.-Descripción General del PIC16F877 2.1.-La Familia del PIC16F877*. (n.d.). https://exa.unne.edu.ar/ingenieria/sysistemas/public_html/Archi_pdf/HojaDatos/Microcontroladores/PIC16F877.pdf
- *Manual de usuario Microchip PIC16F877A (280 páginas)*. (2025). Manual.cr. https://www.manual.cr/microchip/pic16f877a/manual?utm_
- *PIC16F87XA Devices Included in this Data Sheet: High-Performance RISC CPU*. (n.d.). https://ww1.microchip.com/downloads/en/DeviceDoc/39582C.pdf
- *puertos-de-entradasalida - MIKROE*. (2024). MIKROE. https://www.mikroe.com/ebooks/microcontroladores-pic-programacion-en-c-con-ejemplos/puertos-de-entradasalida