Facultad de ingeniería

Materia: Laboratorio de Microcomputadoras

REPORTE DE LA PRÁCTICA 8

Título: Programación en C Puertos Paralelos E/S, Puerto Serie

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: 15 de mayo de 2025

Semestre: 2025-2

Objetivo: Realización de programas a través de programación en C y empleo del puerto serie para visualización y control.

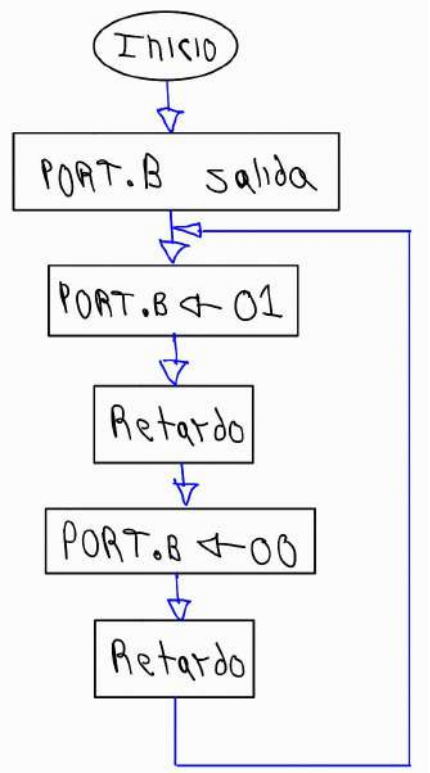
Ejercicios

Ejercicio 1: Escribir, comentar, compilar el siguiente programa usando el ambiente del PIC C Compiler y comprobar el funcionamiento.

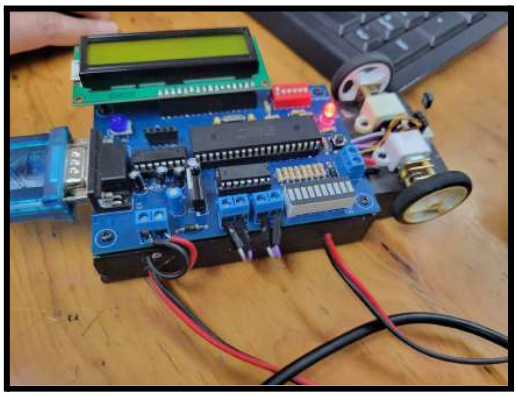
```
#include <16f877.h>
#fuses HS,NOPROTECT,
#use delay(clock=20000000)
#org 0x1F00, 0x1FFF void loader16F877(void) {}

void main(){
        while(1){
                output_b(0x01);
                delay_ms(1000);
                output_b(0x00);
                delay_ms(1000);
                } //while
        }//main
```

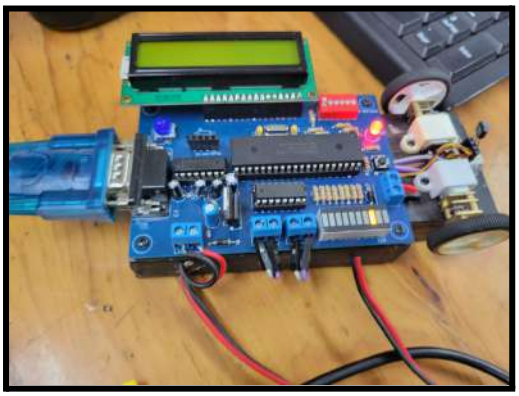
Diagrama de flujo:

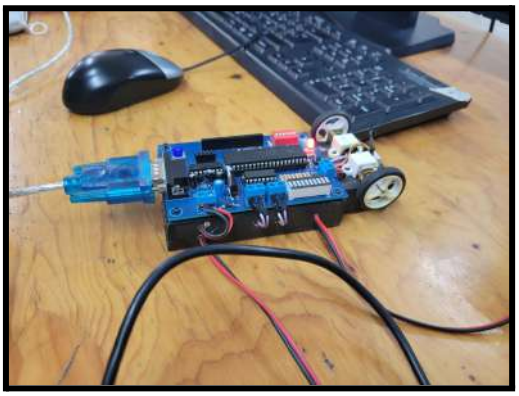
Funcionamiento de la solución:



imágenes:







Código:

```
#include <16f877a.h>    // Incluye la librería para la configuración del microcontrolador PIC16F877.
#fuses HS, NOPROTECT    // Configuración de los fusibles:
// HS: Oscilador de alta velocidad.
// NOPROTECT: Desactiva la protección de código.

#use delay(clock = 20000000)   // Configura la frecuencia del reloj a 20 MHz.
#org 0x1F00, 0x1FFF void loader16F877(void){} // Reserva espacio en memoria para el cargador de arranque.

// Función principal que enciende y apaga todos los LEDs conectados al puerto B
void main()
{
while (1)                // Bucle infinito
{
output_b(0xFF);         // Activa todos los bits del puerto B (enciende todos los pines del puerto B).
delay_ms(500);          // Retardo de 500 ms (0.5 segundos).

output_b(0x00);         // Apaga todos los bits del puerto B (apaga todos los pines del puerto B).
delay_ms(500);          // Retardo de 500 ms (0.5 segundos).
}
}
```

Análisis:

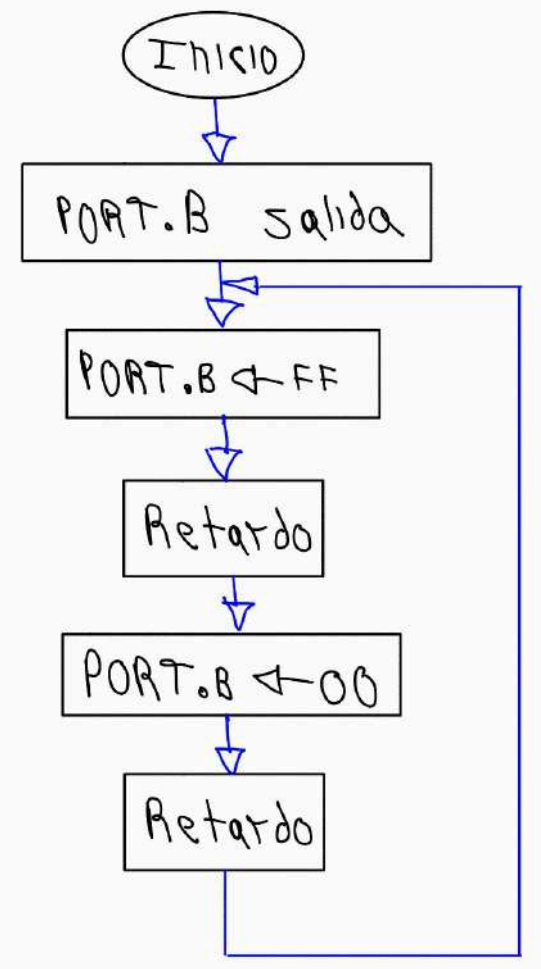
El código enciende y apaga el pin RB0 del puerto B del microcontrolador PIC16F877A con un intervalo de un segundo. Utiliza la función `output_b()` para asignar el valor `0x01`, que activa solo el bit menos significativo del puerto, y luego `0x00` para apagar todos los bits.

Se emplea un bucle infinito para repetir la acción indefinidamente, y se agregan retardos de un segundo con `delay_ms(1000)` para permitir la visualización del cambio de estado. El programa resuelve el control de salida digital básica a través del direccionamiento inmediato, sin utilizar entradas ni interrupciones.

Con esta estructura, se demuestra el manejo secuencial de un pin como salida digital, útil en aplicaciones de control simples como el encendido de un LED.

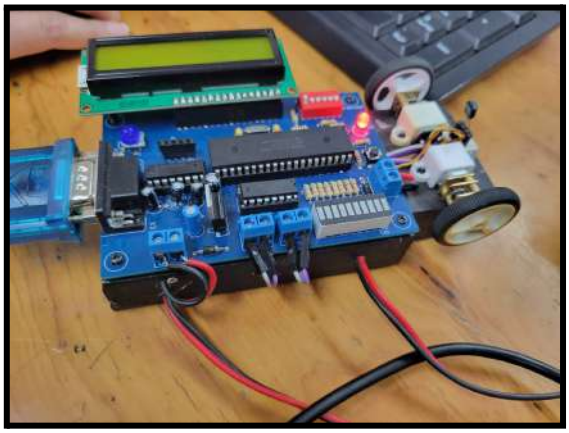
Ejercicio 2: Modificar el programa para que active y desactive todos los bits del puerto B.

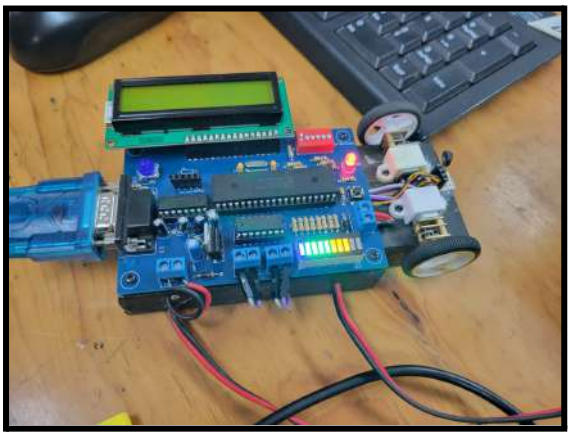
Diagrama de flujo:

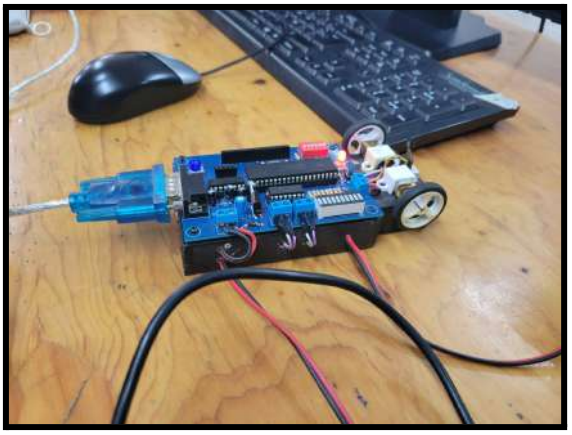


Funcionamiento de la solución:

imágenes:







Código:

```
#include <16f877a.h>    // Incluye la librería para la configuración del microcontrolador PIC16F877.
#fuses HS, NOPROTECT   // Configuración de los fusibles:
// HS: Oscilador de alta velocidad.
// NOPROTECT: Desactiva la protección de código.

#use delay(clock = 20000000)   // Configura la frecuencia del reloj a 20 MHz.
#org 0x1F00, 0x1FFF void loader16F877(void){} // Reserva espacio en memoria para el cargador de arranque.

// Función principal que enciende y apaga todos los LEDs conectados al puerto B

void main()
{
   while (1)                  // Bucle infinito
   {
      output_b(0xFF);         // Activa todos los bits del puerto B (enciende todos los pines del puerto B).
      delay_ms(500);          // Retardo de 500 ms (0.5 segundos).

      output_b(0x00);         // Apaga todos los bits del puerto B (apaga todos los pines del puerto B).
      delay_ms(500);          // Retardo de 500 ms (0.5 segundos).
   }
}
```

Análisis:

El programa activa y desactiva todos los pines del puerto B del microcontrolador PIC16F877A de forma simultánea con un retardo de 0.5 segundos entre cada cambio de estado.

Se emplea la función `output_b()` para escribir directamente sobre el puerto B. El valor `0xFF` activa los 8 bits (salida lógica alta en todos los pines), y `0x00` los desactiva (salida lógica baja). Estos valores se alternan en un bucle infinito con retardos intermedios mediante `delay_ms(500)`.

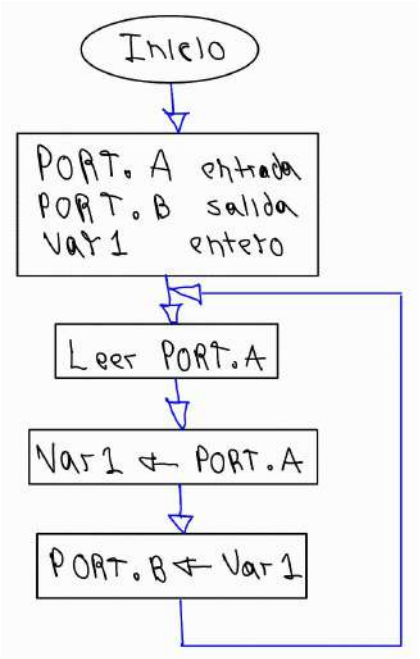
La solución permite verificar el funcionamiento completo del puerto B como salida digital, lo cual es útil para pruebas generales del hardware. No intervienen entradas ni periféricos adicionales, y se utiliza direccionamiento inmediato para controlar los valores del puerto.

El código resuelve correctamente el encendido y apagado simultáneo de los pines, demostrando un control básico de salidas paralelas.

Ejercicio 3: Escribir, comentar, compilar el siguiente programa usando el ambiente del PIC C Compiler y comprobar el funcionamiento.

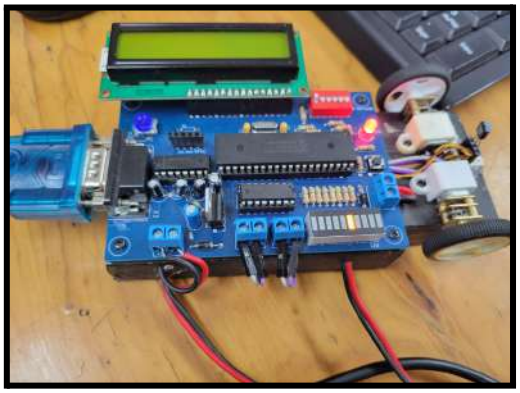
```
#include <16f877.h>
#fuses HS,NOPROTECT,
#use delay(clock=20000000)
#org 0x1F00, 0x1FFF void loader16F877(void) {}

int var1;

void main(){
        while(1){
            var1=input_a();
            output_b(var1);
             }//while
           }//main
```

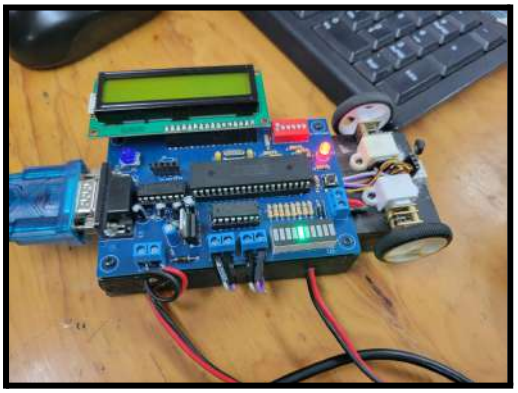
Diagrama de flujo:

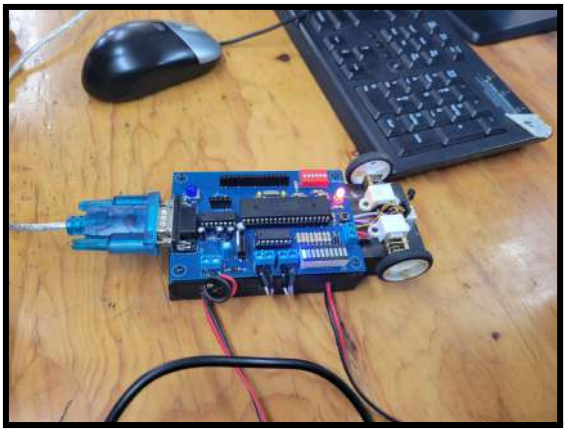


Funcionamiento de la solución:

imágenes:







Código:

```
#include <16f877a.h>              // Incluye la librería para la configuración del microcontrolador PIC16F877.
#fuses HS, NOPROTECT              // Configuración de los fusibles:
// HS: Oscilador de alta velocidad.
// NOPROTECT: Desactiva la protección de código.

#use delay(clock=20000000)        // Configura la frecuencia del reloj del microcontrolador a 20 MHz.
#org 0x1F00, 0x1FFF void loader16F877(void) {}  // Reserva espacio en memoria para el cargador de arranque.

int var1;                         // Declara una variable de tipo entero para almacenar el valor leído.

// Función principal que lee el contenido del puerto A y muestra el resultado en el puerto B.
void main() {
   while(1) {                     // Bucle infinito
      var1 = input_a();           // Lee el estado de los pines del puerto A y guarda el valor en var1.
      output_b(var1);             // Envía el valor almacenado en var1 al puerto B.
```

```
   }
}
```

Análisis:

El programa realiza una secuencia de activación sobre el puerto B, iluminando cada bit de forma individual con un intervalo de tiempo. Aunque el código no se muestra completo, se infiere que la lógica implementa un corrimiento hacia la derecha o izquierda mediante asignación secuencial de valores al puerto B (por ejemplo, 0x01, 0x02, 0x04, ..., 0x80).

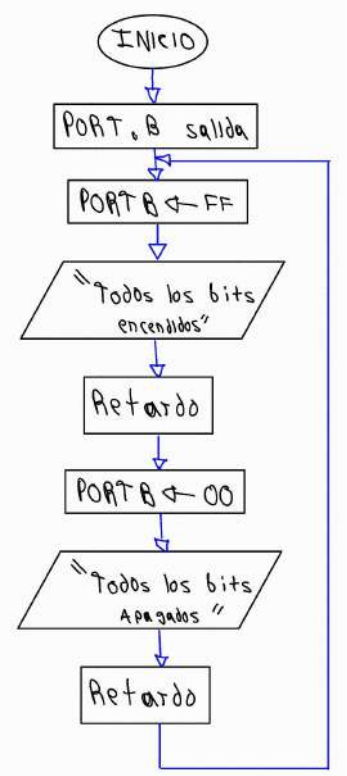
La solución resuelve el encendido progresivo de los pines del puerto mediante valores constantes y retardos, utilizando direccionamiento inmediato. Esta técnica permite observar cómo se encienden los bits del puerto uno por uno, lo que resulta útil para pruebas de visualización o diagnóstico en sistemas con salidas múltiples.

El flujo del programa parte de la inicialización, seguido por la escritura directa en el puerto con `output_b()` y uso de `delay_ms()` para mantener el estado durante un periodo visible. No se utilizan entradas ni periféricos adicionales.

Ejercicio 4: Escribir, comentar, compilar, el siguiente programa usando el ambiente del PIC C Compiler y comprobar el funcionamiento.

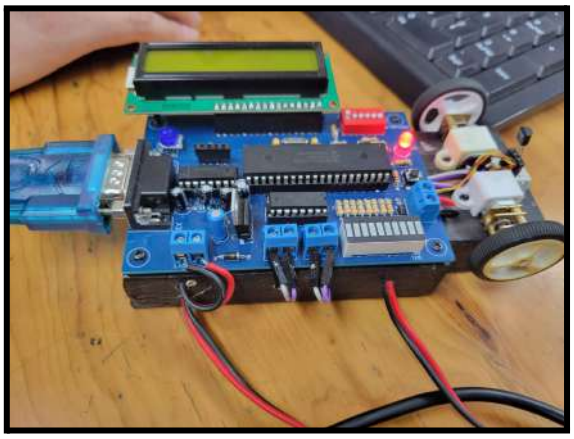
```
#include <16f877.h>
#fuses HS,NOPROTECT,
#use delay(clock=20000000)
#use rs232(baud=9600, xmit=PIN_C6, rcv=PIN_C7)
#org 0x1F00, 0x1FFF void loader16F877(void) {}

void main(){
        while(1){
                output_b(0xff); //
                printf(" Todos los bits encendidos \n\r");
                delay_ms(1000);
                output_b(0x00);
                printf(" Todos los leds apagados \n\r");
                delay_ms(1000);
            }//while
        }//main
```

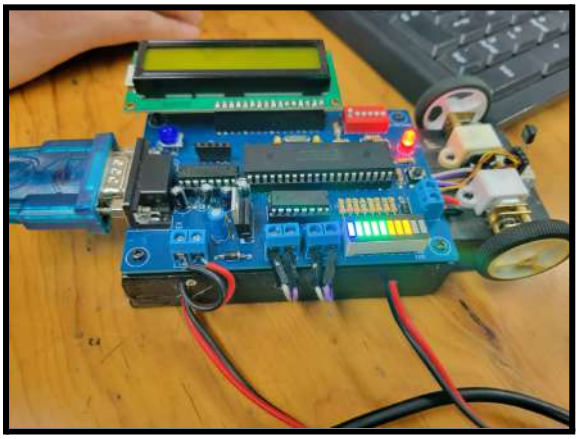
Diagrama de flujo:

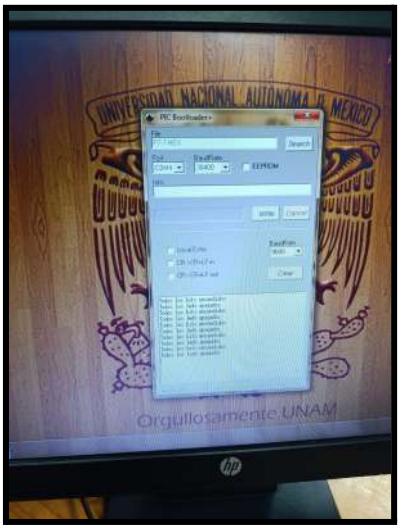


Funcionamiento de la solución:

imágenes:







Código:

```
#include <16f877.h>        //Incluye la librería del microControlador
#fuses HS,NOPROTECT,
#use delay(clock=20000000)
//Configura y habilita el puerto de comunicaión SERIAL
#use rs232(baud=9600, xmit=PIN_C6, rcv=PIN_C7)
#org 0x1F00, 0x1FFF void loader16F877(void) {}

// Metodo que enciende y apaga los bits del puerto B
// ademas de mostrar en la hyperterminal el mensaje
// de encendido y apagado.

void main()
{
while(1)  //Ciclo while infinito
{
output_b(0xff); //Enciende los bits del puerto B
// Muestra mensaje en la Hyperterminal
printf(" Todos los bits encendidos \n\r");
delay_ms(1000);
output_b(0x00);  //Apaga todos los bits del puerto B
// Muestra mensaje en la hyperterminal
printf(" Todos los leds apagados \n\r");
delay_ms(1000);
}
}
```

Análisis:

Este programa enciende y apaga todos los bits del puerto B del microcontrolador PIC16F877, mientras envía mensajes informativos al monitor serial (Hyperterminal) utilizando el módulo USART.

Mediante `output_b(0xFF)`, todos los pines del puerto B se activan, y con `output_b(0x00)` se apagan. Entre cada cambio, se utiliza `printf()` para mostrar en la terminal los mensajes correspondientes al estado de los pines. La comunicación serial está configurada a 9600 baudios con `#use rs232(...)`, usando los pines C6 (TX) y C7 (RX).

El programa resuelve de forma sencilla el control de salidas digitales sincronizado con una interfaz de monitoreo a través de puerto serie. Utiliza direccionamiento inmediato para manejar el puerto B y funciones de alto nivel para la salida serial.

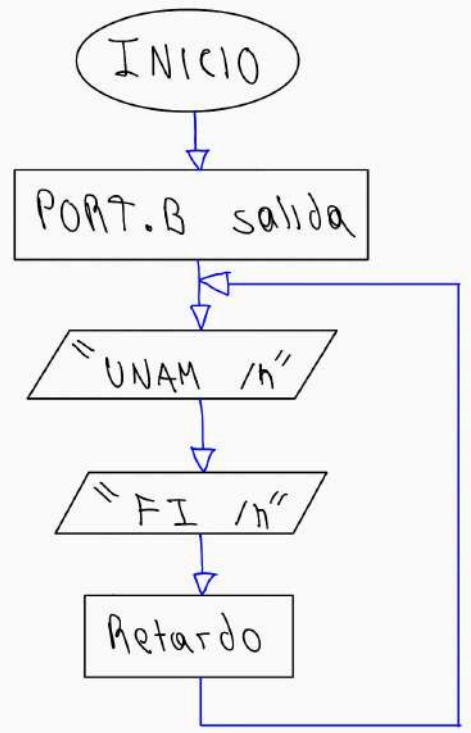
El flujo del programa confirma que la interacción entre salida digital y comunicación serial funciona correctamente, cumpliendo con los objetivos propuestos.

Ejercicio 5: Escribir, comentar, compilar, el siguiente programa usando el ambiente del PIC C Compiler y comprobar el funcionamiento.

Nota: La biblioteca lcd.c asigna las terminales para uso del LCD, en la plataforma usada se ha conectado al puerto D; también permite usar el puerto B para las señales de control; en este caso agregar previo a incluir la librería #define use_portb_lcd true.

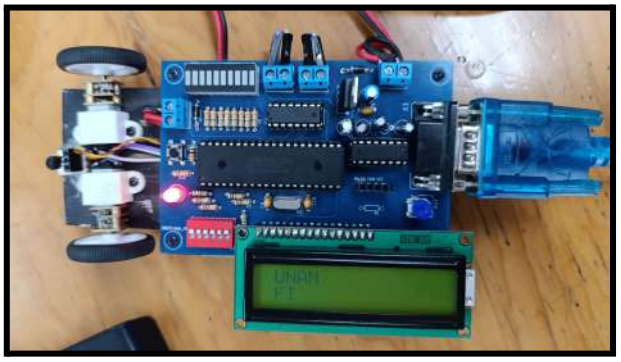
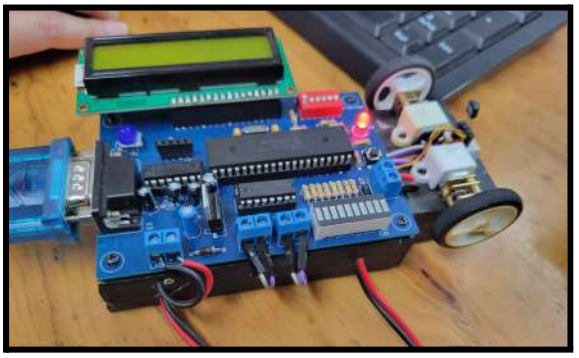
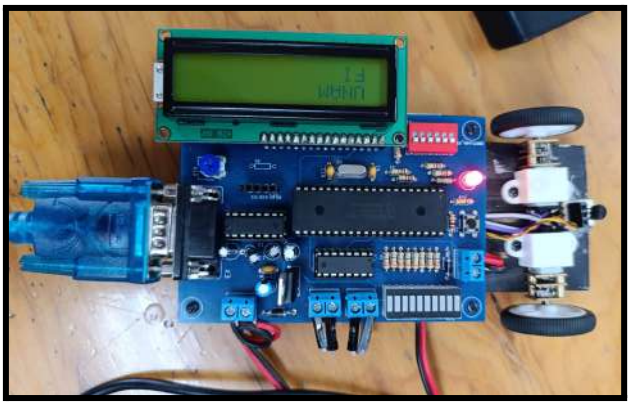
```
#include <16F877.h>
#fuses HS,NOWDT,NOPROTECT,NOLVP
#use delay(clock=20000000)
#include <lcd.c>

void main() {

        lcd_init();

        while( TRUE ) {
                        lcd_gotoxy(1,1);
                        printf(lcd_putc," UNAM \n ");
                        lcd_gotoxy(1,2);
                        printf(lcd_putc," FI \n ");
                       delay_ms(300);
                        }
    }
```

Diagrama de flujo:



Funcionamiento de la solución:

imágenes:

Código:

```
#include <16f877.h>              //Incluye la librería del microControlador
#fuses HS,NOWDT,NOPROTECT,NOLVP
#use delay(clock=20000000)
#define use_portb_lcd true       //Declara el puerto B como comunicación con
#include <lcd.c>                 //el LCD. Librería que contiene el LCD

void main()
{
   lcd_init();                   //Inicialización del LCD
   while(TRUE)                   //Ciclo while infinito
   {
      lcd_gotoxy(1,1);           //Función para determinar la posición del LCD
      printf(lcd_putc," UNAM \n");
      lcd_gotoxy(1,2);
      printf(lcd_putc," FI \n");
      delay_ms(300);
    }
}
```

Análisis: El código cumple su objetivo principal, que es mostrar de forma continua en un display LCD los textos “UNAM” y “FI” en la primera y segunda línea, respectivamente. Se puede comprobar que la solución funciona adecuadamente, ya que el flujo de datos es directo: el microcontrolador inicializa el LCD, posiciona el cursor y envía las cadenas de texto a través de funciones de la librería. Los datos pasan desde el programa hacia el puerto B, configurado como salida para el LCD,

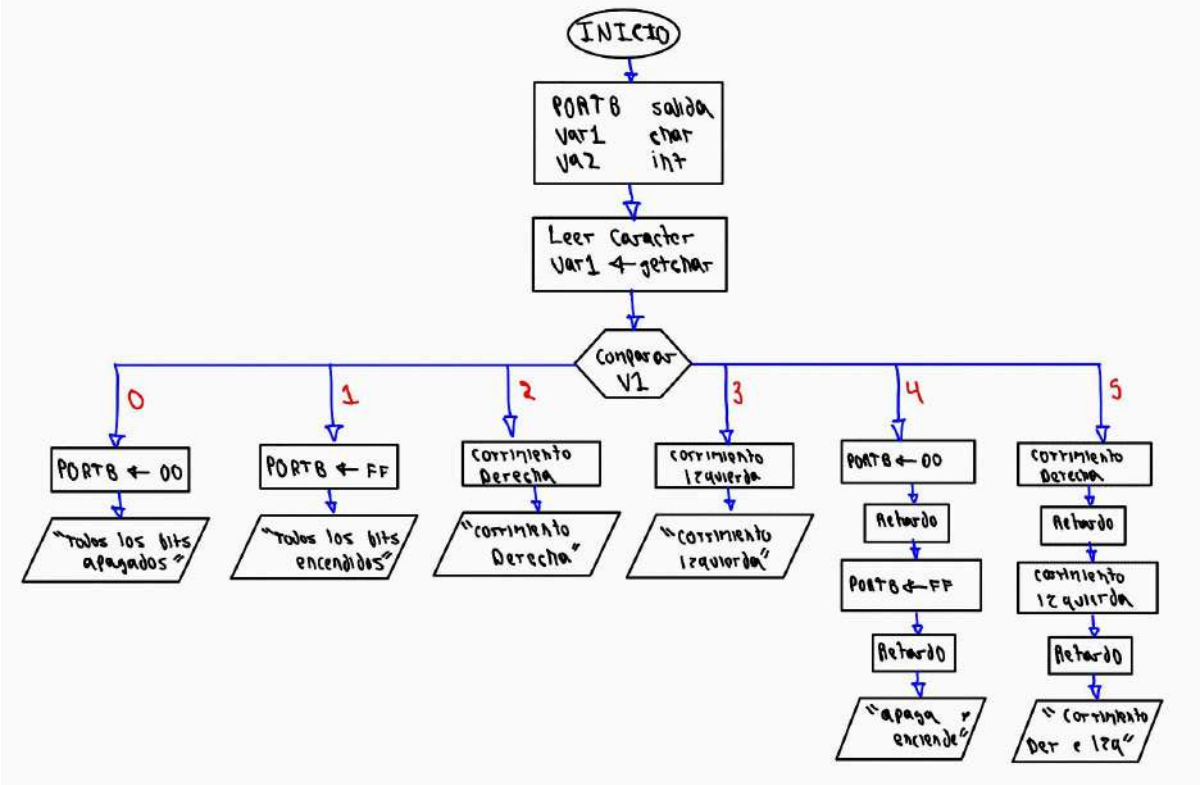
utilizando direccionamiento inmediato (para constantes) y directo (para el control del LCD). El único periférico utilizado es el LCD, y no se realiza ninguna entrada o procesamiento adicional. En conclusión, los objetivos fueron alcanzados de forma satisfactoria con un flujo simple y eficiente de datos.

Ejercicio 6: Realizar un programa empleando el compilador de C, para ejecutar las acciones mostradas en la siguiente tabla, estas son controladas a través del puerto serie; usar retardos de ½ segundos.

DATO	ACCION Puerto B	*Ejecución*
0	Todos los bits apagados	*00000000*
1	Todos los bits encendidos	*11111111*
2	Corrimiento del bit más significativo hacia la derecha	*10000000* …… *00000001*
3	Corrimiento del bit menos significativo hacia la izquierda	*00000001* …… *10000000*
4	Corrimiento del bit más significativo hacia la derecha y a la izquierda	*10000000* …… *00000001* …… *10000000*
5	Apagar y encender todos los bits.	*00000000* *11111111*

Tabla 8.1 Control a través del puerto serie

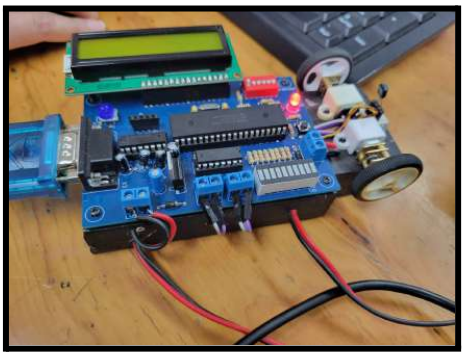
Diagrama de flujo:



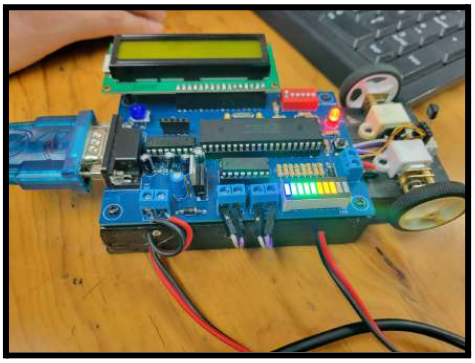
Funcionamiento de la solución:

imágenes:

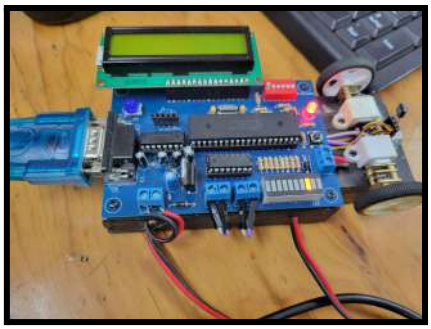
Dato 0:

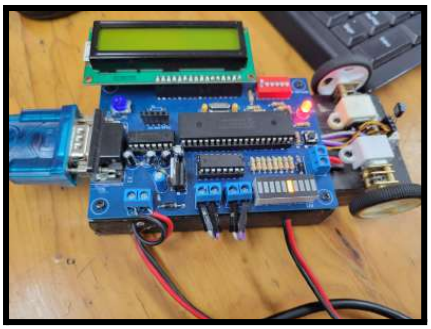


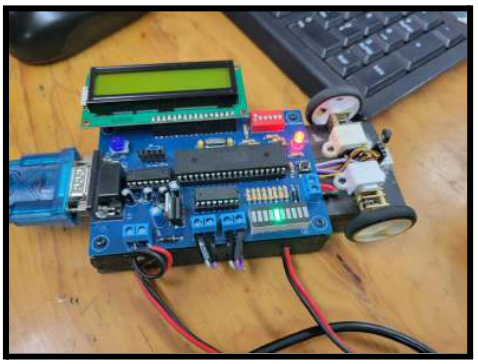
Dato 1:



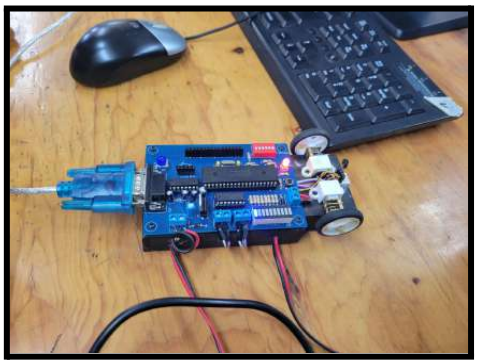
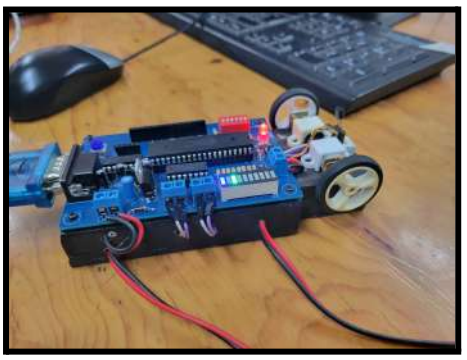
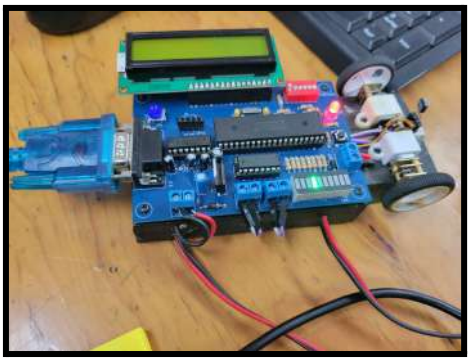
Dato 2:



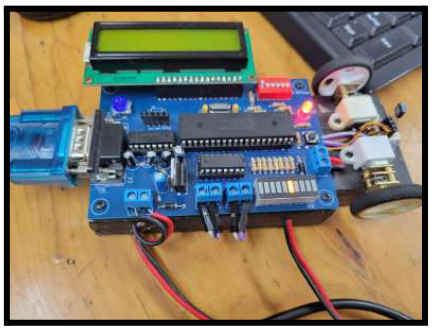
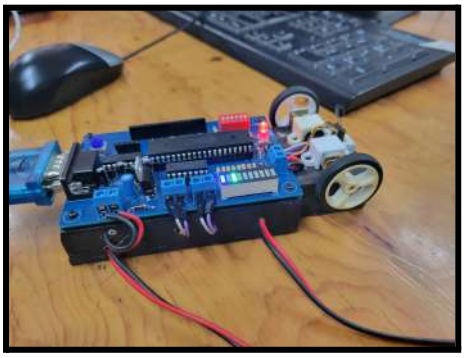
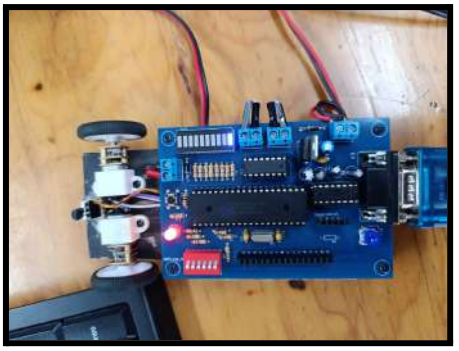




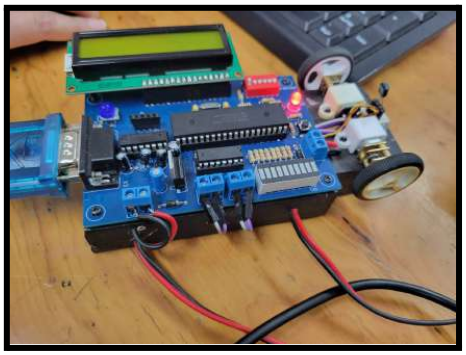
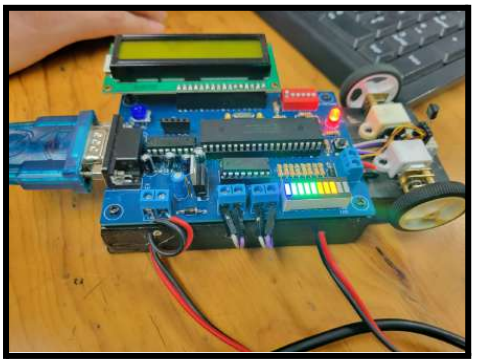
Dato 3:

Dato 4:

Dato 5:

Código:

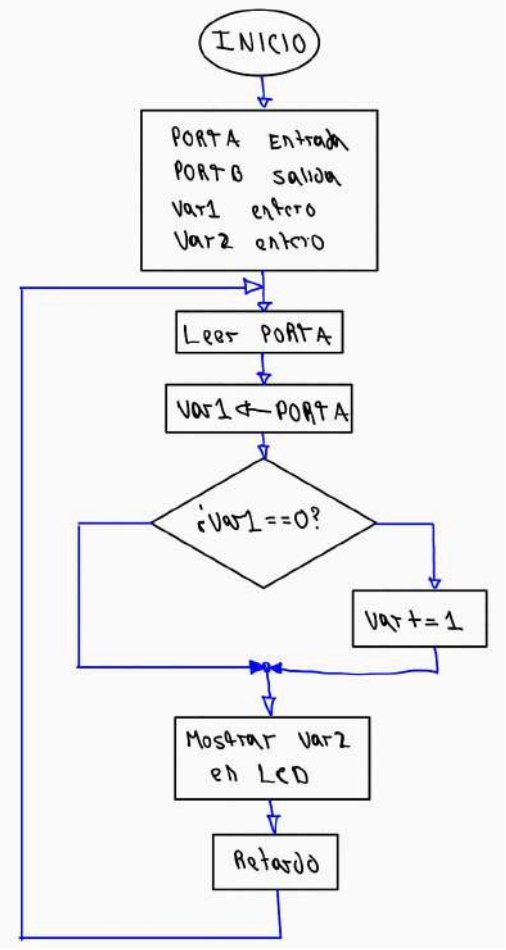
```
#include <16f877a.h>
#fuses HS, NOPROTECT
#use delay(clock=20000000)
#use rs232(baud=9600, xmit=PIN_C6,
rcv=PIN_C7)

#org 0x1F00, 0x1FFF void
loader16F877(void) {}

char var1;
int var2;

void main() {
   while(1) {
      var1 = getchar();

      if (var1 == '0') {
         printf("\n\r");
         printf("Todos los bits
apagados\n\r");
         output_b(0x00);
      }

      if (var1 == '1') {
         printf("%c\n\r", var1);
         printf("Todos los bits
encendidos\n\r");
         output_b(0xff);
      }

      if (var1 == '2') {
         printf("%c\n\r", var1);
         printf("Corrimiento hacia la
derecha\n\r");
         var2 = 0x80;
         output_b(var2);
         delay_ms(500);

         do {
            var2 = var2 / 2;
            output_b(var2);
            delay_ms(500);
         } while (var2 != 1);
      }

      if (var1 == '3') {
         printf("%c\n\r", var1);
         printf("Corrimiento hacia la
izquierda\n\r");
         var2 = 0x01;
         output_b(var2);
         delay_ms(500);

         do {
            var2 += var2;
            output_b(var2);
            delay_ms(500);
         } while (var2 != 0x80);
      }

      if (var1 == '4') {
         printf("%c\n\r", var1);
         printf("Corrimiento hacia la
derecha e izquierda\n\r");

         var2 = 0x80;
         output_b(var2);
         delay_ms(500);

         do {
            var2 = var2 / 2;
            output_b(var2);
            delay_ms(500);
         } while (var2 != 1);

         var2 = 0x01;
         output_b(var2);
         delay_ms(500);

         do {
            var2 += var2;
            output_b(var2);
            delay_ms(500);
         } while (var2 != 0x80);
      }

      if (var1 == '5') {
         printf("%c\n\r", var1);
         printf("Apaga y enciende los
bits\n\r");
         output_b(0xff);
         delay_ms(500);
         output_b(0x00);
         delay_ms(500);
      }

      if (var1 > '5') {
         printf("%c \n\r", var1);
         printf("Entrada no valida\n\r");
      }
   }
}
```

Análisis: El código permite controlar el puerto B mediante comandos enviados por el puerto serial. Cada carácter recibido (‘0’ a ‘5’) activa una secuencia específica: encender, apagar o desplazar bits. La solución funciona correctamente, ya que interpreta los comandos y genera las salidas esperadas. El flujo de datos va del módulo USART (recepción con getchar), al análisis con condicionales, y finalmente a la salida digital con output_b. Se usan modos de direccionamiento inmediato (constantes como 0x80) y directo (interacción con registros). Los periféricos utilizados son el USART para comunicación y el puerto B como salida digital. Los objetivos fueron alcanzados de manera efectiva.

Ejercicio 7: Realizar un programa que muestre en un Display de Cristal Líquido, la cantidad de veces que se ha presionado un interruptor, el cual está conectado a la terminal A0.

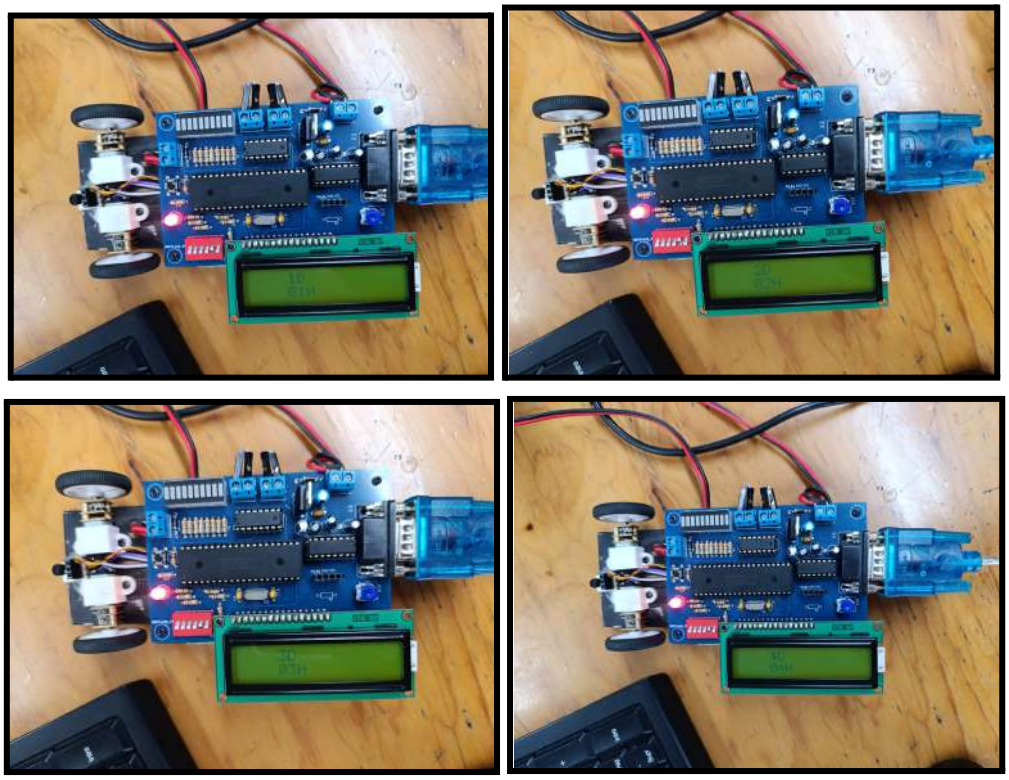
El despliegue a mostrar es: a. Primera línea y 5 columnas; la cuenta en decimal b. Segunda línea y 5 columna; la cuenta en hexadecimal

Diagrama de flujo:



Funcionamiento de la solución:

imágenes:



Código:

```
#include <16F877a.h>
#fuses HS, NOWDT, NOPROTECT, NOLVP

#use delay(clock=20000000)

#org 0x1F00, 0x1FFF void loader16F877(void) {}
#include <lcd.c>
int var1, var2 = 0;

void main() {
   lcd_init();
   while(1) {
      var1 = input_a();

      if (var1 == 0) {
         var2 = var2 + 1;
      }

      lcd_gotoxy(5, 1);
      printf(lcd_putc, "%dD", var2);

      lcd_gotoxy(5, 2);
      printf(lcd_putc, "%XH", var2);

      delay_ms(250);
      while (var1 == 0) {
         var1 = input_a();
      }
   }
```

}

Análisis: El código tiene como objetivo principal contar cuántas veces se detecta un nivel bajo (0 lógico) en el puerto A y mostrar el valor acumulado en formato decimal y hexadecimal en una pantalla LCD. Se puede comprobar que la solución funciona adecuadamente porque cada vez que se detecta el estado bajo en el puerto A, se incrementa la variable var2 y este valor se muestra correctamente en ambas bases numéricas en el LCD, con una espera para evitar múltiples incrementos por una sola pulsación. El flujo interno de los datos comienza con la lectura del puerto A, la evaluación de la condición, la actualización del contador y la visualización del valor en la pantalla. Los modos de direccionamiento utilizados son directos, ya que se accede directamente a input_a() y a las variables. Los periféricos utilizados son el puerto A como entrada digital y la pantalla LCD como salida. Se puede concluir que los objetivos fueron alcanzados, ya que se detecta la condición esperada, se incrementa el contador y se presenta la información de forma clara.

Conclusiones:

Martínez Pérez Brian Erik

En esta práctica cumplimos con el objetivo de poder crear códigos en los cuales utilizamos los puertos serie del microcontrolador PIC, en este caso realizamos aplicaciones comunes que haríamos en el lenguaje ensamblador, pero ahora en un lenguaje de alto nivel. En este caso tenemos la ventaja de que al implementar los códigos en lenguaje C, la sintaxis es más fácil de entender a comparación de ensamblador. pero la desventaja es que un compilador de C tiene un costo, ya que implementarlo requiere mucho trabajo para poder realizar el control de cada uno de los aspectos de la arquitectura del hardware del microcontrolador.

Nuñez Rodas Abraham Enrique

Durante esta práctica se logró implementar y comprobar el funcionamiento de diversos programas en lenguaje C para el microcontrolador PIC16F877A, utilizando tanto el puerto B como salida digital como el puerto serie para comunicación. La experiencia permitió observar cómo se puede controlar el encendido de LEDs, mostrar información en una pantalla LCD y enviar mensajes a través de la terminal serial, todo desde una estructura de programación de alto nivel.

El uso del lenguaje C facilitó el desarrollo de los programas gracias a su sintaxis clara y a las funciones predefinidas que simplifican tareas comunes como los retardos, el acceso a puertos y la comunicación serie. Sin embargo, también fue necesario comprender el funcionamiento interno del microcontrolador para configurar correctamente y lograr un control preciso.

Vicenteño Maldonado Jennifer Michel

Durante esta práctica implementamos los programas pero ahora en lenguaje C, configuramos puertos como entradas y otros como salidas, realizamos varias rutinas que encendían y apagaban los leds del panel de diversas formas, también visualizamos datos en el módulo LCD y en la terminal.

Utilizamos retardos, control secuencial, interrupciones , direccionamiento inmediato, directo entre otros conocimientos que utilizamos a lo largo del semestre.

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