



**Unidad Académica
Multidisciplinaria
Mante**

Examen Práctico
Unidad académica Multidisciplinaria Mante

Asignatura: Programación De Microprocesadores

Nombre del trabajo: Examen Parcial 1

Nombre del docente: Ing. Daniel López Piña

Nombre del alumnos:

Amaya Montalvo Pablo Iván

José Ricardo Vargas Guillén

Medina Rodríguez Oscar

Carbajal Mejía Adrián

Orta Solís José Aldo

Rodríguez García Vanessa

Olvera Padrón Carlos Michel

Especialidad: Ing. Sistemas Computacionales

Grado y Grupo: 8 “EJ”

Fecha: 25 de febrero del 2025

Actividades por realizar:

Selección de componentes:

Rodríguez García Vanessa

Ensamble del circuito:

Orta Solís José Aldo

Codificación del programa:

Amaya Montalvo Pablo Iván y Olvera Padrón Carlos Michel

Grabador del Microcontrolador:

Vargas Guillén José Ricardo

Prueba del circuito:

Carbajal Mejía Adrián

Documentación:

Medina Rodríguez Oscar

Selección de componentes

Para el desarrollo de nuestro circuito con el PIC18F4550, hemos seleccionado cuidadosamente cada componente, asegurándonos de que cumpla con los requisitos del proyecto. Nuestro equipo ha analizado las necesidades del sistema y hemos decidido utilizar los siguientes elementos esenciales:

Material necesario:

- PIC18F4550
- 4 LEDs
- 4 resistencias de 3300 (protección para los LEDs)
- Fuente de alimentación 5V
- Protoboard y cables
- Condensador cerámico 22pf
- Osilador 20,000

PIC18F4550

Es un microcontrolador que se usa para controlar diferentes dispositivos electrónicos. Se programa para realizar tareas específicas, como encender LEDs, leer sensores o comunicarse con otros dispositivos.

LEDs

Son pequeñas luces que se encienden cuando pasa corriente a través de ellas. Se usan en circuitos electrónicos para indicar el estado de funcionamiento de un sistema.

Resistencias de 3.3K Ω

Se utilizan para limitar la corriente en un circuito y proteger componentes como los LEDs, evitando que reciban demasiada electricidad y se quemen.

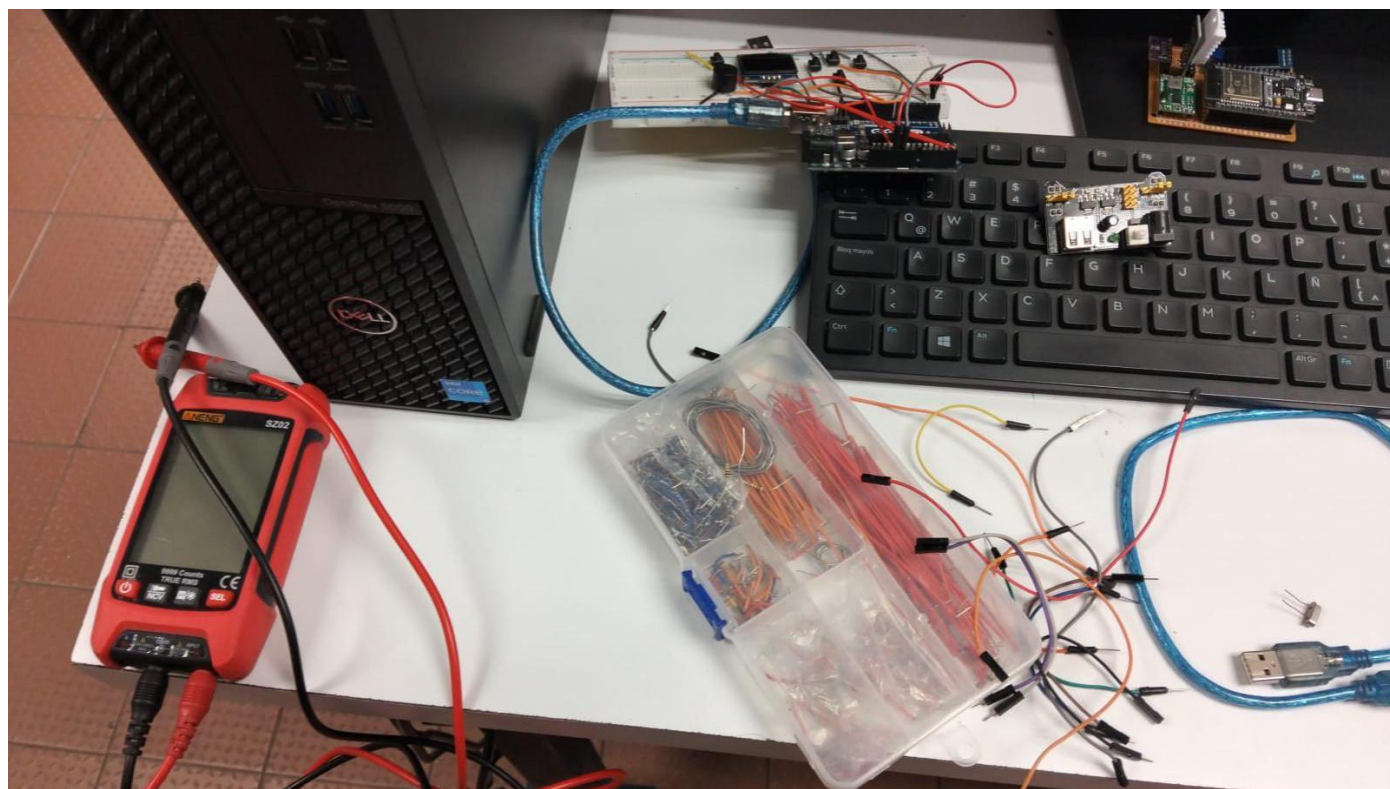
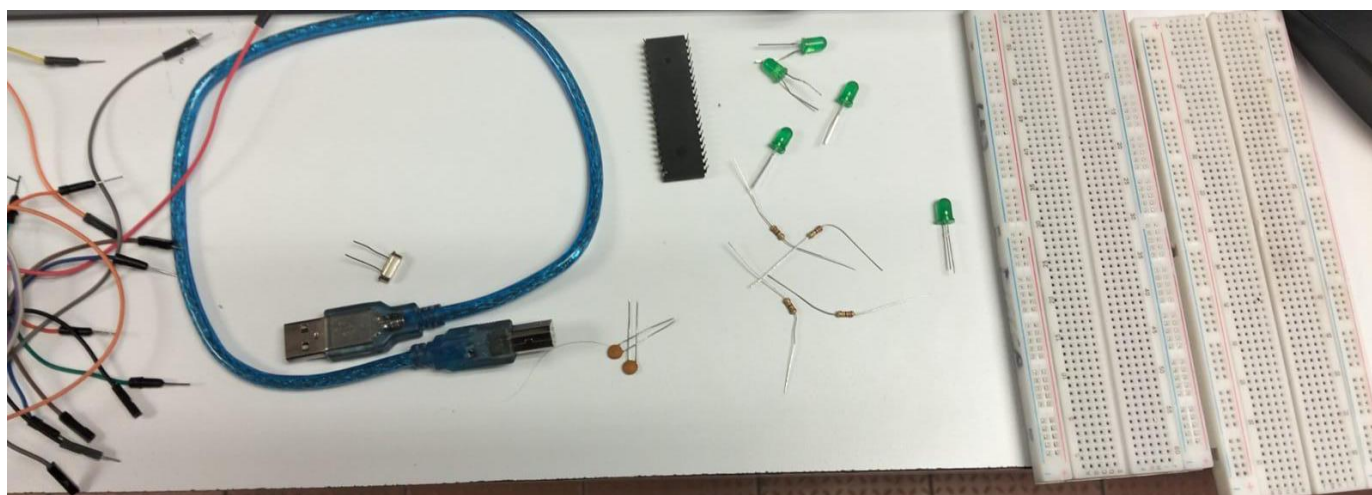
Fuente de alimentación de 5V

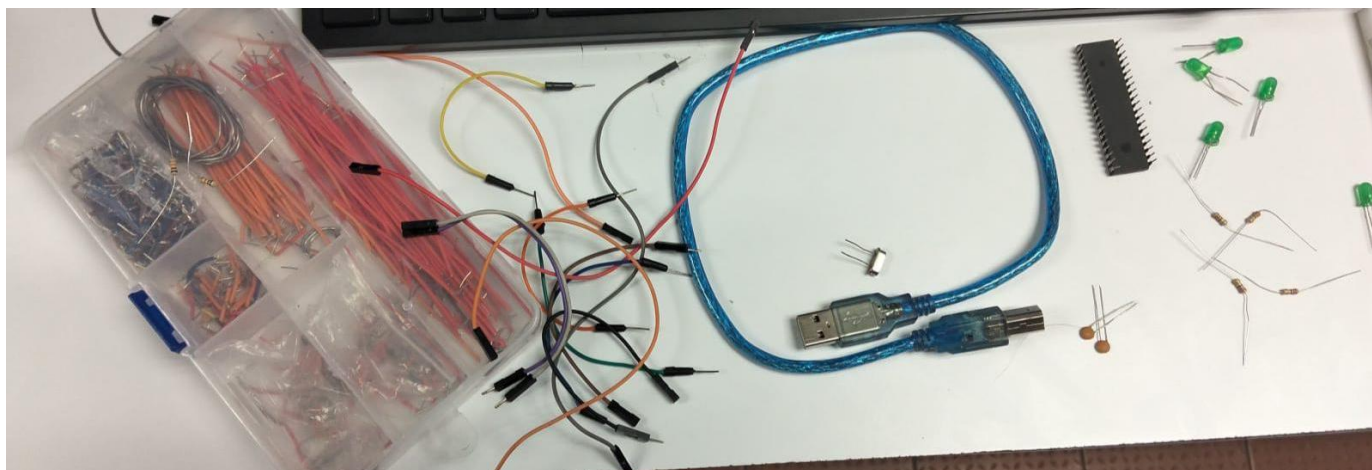
Proporciona la energía necesaria para que el circuito funcione. Puede ser un adaptador, una batería o un puerto USB.

Protoboard y cables

La protoboard permite conectar componentes sin necesidad de soldar, y los cables sirven para unir las diferentes partes del circuito.

Evidencia:





Ensamble del circuito:

Conexión de los LEDs con resistencias:

1. LED 1 (en RA0 - Pin 2):

- Tomamos una resistencia de 220Ω .
- Conectamos un extremo de la resistencia al pin RA0 (Pin 2) del microcontrolador.
- El otro extremo de la resistencia va al ánodo del LED 1 (el lado largo del LED).
- El cátodo del LED 1 (el lado corto del LED) se conecta a GND (tierra) del sistema.

2. LED 2 (en RA1 - Pin 3):

- Nuevamente, usamos una resistencia de 220Ω .
- Un extremo de la resistencia se conecta al pin RA1 (Pin 3) del microcontrolador.
- El otro extremo de la resistencia se conecta al ánodo del LED 2.
- El cátodo del LED 2 se conecta a GND.

3. LED 3 (en RB0 - Pin 33):

- Usamos una resistencia de 220Ω .
- Conectamos un extremo de la resistencia al pin RB0 (Pin 33) del microcontrolador.
- El otro extremo de la resistencia va al ánodo del LED 3.

- El cátodo del LED 3 se conecta a GND.
- 4. LED 4 (en RC0 - Pin 15):
 - Usamos una resistencia de 220Ω .
 - Conectamos un extremo de la resistencia al pin RC0 (Pin 15) del microcontrolador.
 - El otro extremo de la resistencia va al ánodo del LED 4.
 - El cátodo del LED 4 se conecta a GND.

Conexión del oscilador de 20 MHz:

1. Colocamos el cristal de 20 MHz entre los pines OSC1 (Pin 13) y OSC2 (Pin 14) del microcontrolador.
2. Colocamos dos capacitores de 22pF:
 - Un capacitor va entre OSC1 (Pin 13) y GND.
 - El otro capacitor va entre OSC2 (Pin 14) y GND.

Conexión de la resistencia de $10k\Omega$:

1. Colocamos una resistencia de $10k\Omega$.
 - Un extremo de la resistencia se conecta al Pin 1 del microcontrolador.
 - El otro extremo de la resistencia se conecta a +5V (la alimentación).

Conexión de la alimentación (VDD y VSS):

1. El pin VDD (Pin 11 y Pin 32) del microcontrolador se conecta a +5V de la fuente de alimentación.
2. El pin VSS (Pin 12 y Pin 31) del microcontrolador se conecta a GND.

Evidencia:

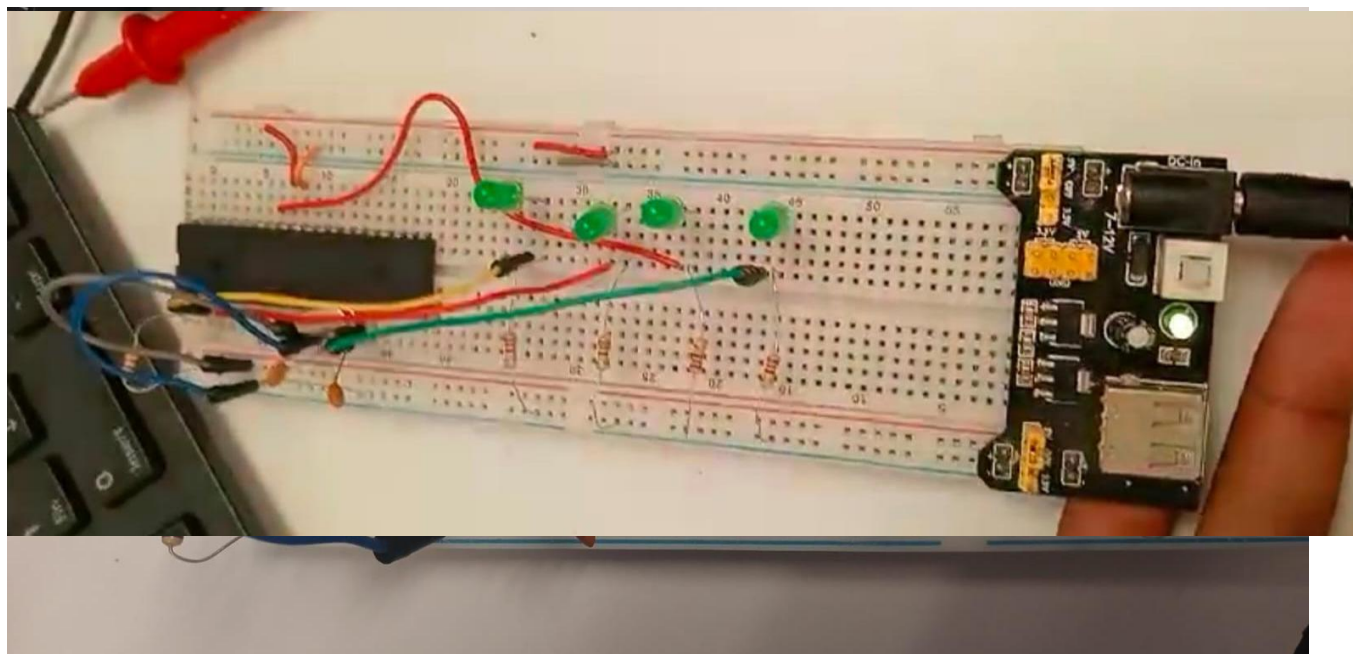
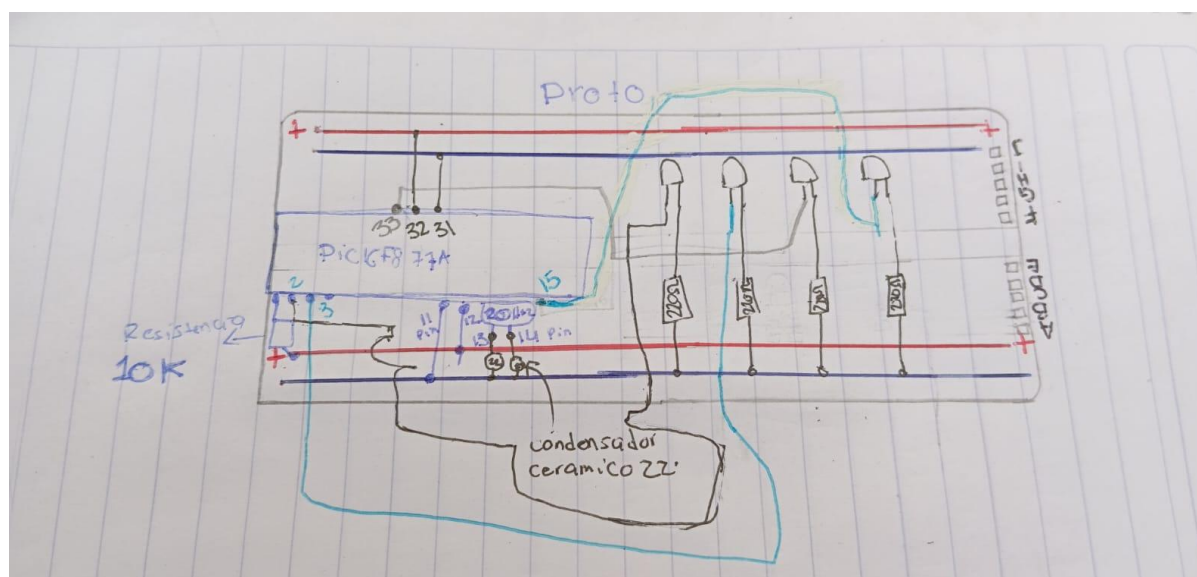


Diagrama:



Codificación del programa:

Código:

```
// Configuración de fuses para PIC16F877A

#pragma config FOSC = HS    // Oscilador de alta velocidad con cristal externo
#pragma config WDTE = OFF   // Desactivar el Watchdog Timer
#pragma config PWRTE = OFF  // Desactivar el Power-up Timer
#pragma config BOREN = OFF  // Desactivar Brown-out Reset
#pragma config LVP = OFF    // Desactivar Low Voltage Programming
#pragma config CPD = OFF    // Desactivar protección de datos EEPROM
#pragma config WRT = OFF    // Desactivar protección de escritura en memoria de programa
#pragma config CP = OFF     // Desactivar protección del código

void main() {
    // Configuración de pines como salida
    TRISA = 0x00; // RA0 y RA1 como salida (puerto A)
    TRISB = 0x00; // RB0 como salida (puerto B)
    TRISC = 0x00; // RC0 como salida (puerto C)

    // Apaga todos los LEDs inicialmente
    PORTA = 0x00;
    PORTB = 0x00;
    PORTC = 0x00;

    // Bucle principal
    while (1) {
```

```
// Enciende todos los LEDs al mismo tiempo
```

```
PORTA = 0x03; // Enciende RA0 y RA1
```

```
PORTB = 0x01; // Enciende RB0
```

```
PORTC = 0x01; // Enciende RC0
```

```
Delay_ms(500); // Espera 500 ms
```

```
// Apaga todos los LEDs al mismo tiempo
```

```
PORTA = 0x00; // Apaga RA0 y RA1
```

```
PORTB = 0x00; // Apaga RB0
```

```
PORTC = 0x00; // Apaga RC0
```

```
Delay_ms(500); // Espera 500 ms
```

```
}
```

```
}
```

Descripción del código:

1. Fuses de configuración: El código establece varios fuses que definen el comportamiento del microcontrolador:
 - FOSC = HS: Se usa un oscilador externo de alta velocidad (20 MHz), lo que indica que se utilizará un cristal externo para la frecuencia de operación.
 - WDTE = OFF: Desactiva el Watchdog Timer, lo que significa que el PIC no se reiniciará automáticamente si hay un error.
 - PWRTE = OFF: Desactiva el Power-up Timer, que previene el arranque del PIC hasta que se estabiliza la alimentación.
 - BOREN = OFF: Desactiva el Brown-out Reset, por lo que el PIC no se reiniciará si el voltaje de alimentación cae por debajo de un umbral.

- LVP = OFF: Desactiva la Low Voltage Programming, lo que evita la reprogramación del PIC a voltajes bajos.
- CPD = OFF: Desactiva la protección de la memoria EEPROM.
- WRT = OFF: Desactiva la protección de escritura en la memoria de programa.
- CP = OFF: Desactiva la protección del código, lo que permite modificar el código del PIC.

2. Configuración de los pines:

- TRISA = 0x00: Establece todos los pines del puerto A (RA0, RA1) como salidas.
- TRISB = 0x00: Establece el pin RB0 como salida.
- TRISC = 0x00: Establece el pin RC0 como salida.

Esto configura los pines RA0, RA1, RB0 y RC0 para controlar los LEDs.

3. Apagar todos los LEDs: Se asegura de que todos los LEDs estén apagados al principio:

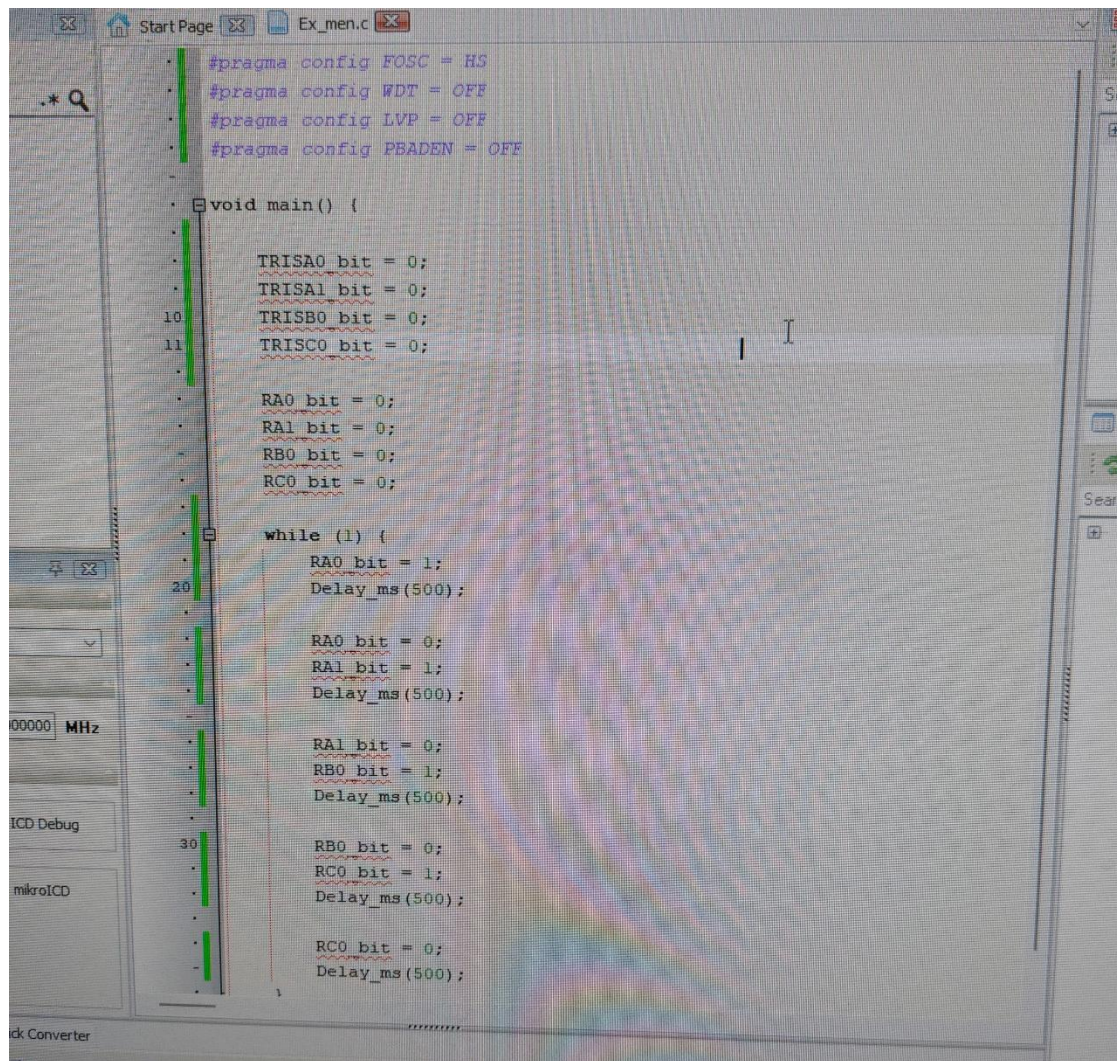
- PORTA = 0x00;; Apaga los LEDs conectados a los pines RA0 y RA1.
- PORTB = 0x00;; Apaga el LED conectado al pin RB0.
- PORTC = 0x00;; Apaga el LED conectado al pin RC0.

4. Bucle principal (while 1): El bucle infinito ejecuta lo siguiente repetidamente:

- Encender LEDs:
 - PORTA = 0x03;; Enciende los LEDs conectados a RA0 y RA1 (los dos primeros bits de PORTA).
 - PORTB = 0x01;; Enciende el LED conectado a RB0.
 - PORTC = 0x01;; Enciende el LED conectado a RC0.
- Se espera 500 ms con la función Delay_ms(500);.
- Apagar LEDs:
 - PORTA = 0x00;; Apaga los LEDs conectados a RA0 y RA1.
 - PORTB = 0x00;; Apaga el LED conectado a RB0.
 - PORTC = 0x00;; Apaga el LED conectado a RC0.
- Se espera 500 ms con la función Delay_ms(500);.

Este proceso se repite continuamente, haciendo que los LEDs parpadeen, encendiéndose y apagándose cada medio segundo.

Evidencia:



```
#pragma config FOSC = HS
#pragma config WDT = OFF
#pragma config LVP = OFF
#pragma config PBADEN = OFF

void main() {
    TRISA0 bit = 0;
    TRISA1 bit = 0;
    TRISB0 bit = 0;
    TRISC0 bit = 0;

    RA0 bit = 0;
    RA1 bit = 0;
    RB0 bit = 0;
    RC0 bit = 0;

    while (1) {
        RA0 bit = 1;
        Delay_ms(500);

        RA0 bit = 0;
        RA1 bit = 1;
        Delay_ms(500);

        RA1 bit = 0;
        RB0 bit = 1;
        Delay_ms(500);

        RB0 bit = 0;
        RC0 bit = 1;
        Delay_ms(500);

        RC0 bit = 0;
        Delay_ms(500);
    }
}
```



```
Start Page Ex_men.c
#pragma config FOSC = H
#pragma config WDT = OFF
#pragma config LVP = OFF
#pragma config PBADEN = OFF

void main() {
    TRISA0_bit = 0;
    TRISA1_bit = 0;
    TRISB0_bit = 0;
    TRISC0_bit = 0;

    RA0_bit = 0;
    RA1_bit = 0;
    RB0_bit = 0;
    RC0_bit = 0;

    while (1) {
        RA0_bit = 1;
        RA1_bit = 1;
        RB0_bit = 1;
        RC0_bit = 1;
        Delay_ms(500);

        RA0_bit = 0;
        RA1_bit = 0;
        RB0_bit = 0;
        RC0_bit = 0;
        Delay_ms(500);
    }
}
```


Grabador del microcontrolador:

Pasos realizados:

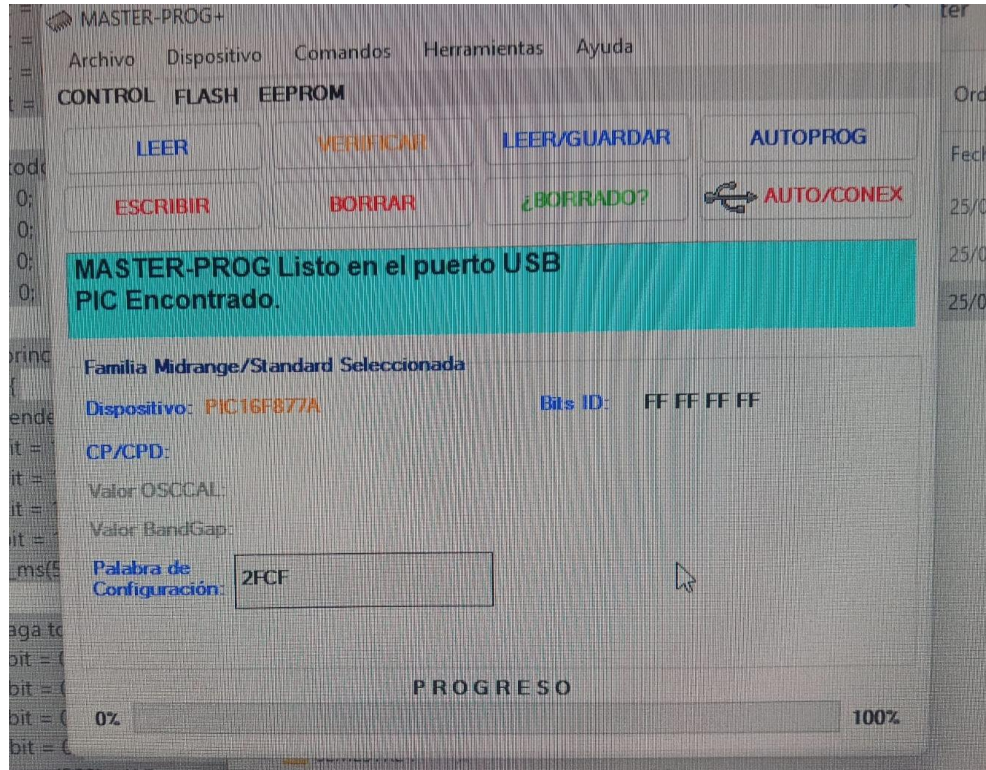
- ☐ Preparación del Código: Primero, escribí el código en lenguaje C utilizando MPLAB X IDE. Después de compilarlo, se generó un archivo .hex que contiene el código binario que el microcontrolador puede entender y ejecutar. Este archivo es lo que realmente vamos a grabar en el PIC.
- ☐ Conexión del Programador: Conecté el programador PICkit 3 al puerto USB de mi PC. Luego, conecté el programador al microcontrolador PIC16F877A a través de los pines de programación:
 - MCLR (pin de reset) lo conecté al pin correspondiente del PICkit 3.
 - VDD lo conecté a la fuente de alimentación de 5V.
 - VSS lo conecté a GND.
 - PGD y PGC los conecté a los pines de comunicación de datos y reloj del programador y del PIC.
- ☐ Configuración en MPLAB IPE: Abrí MPLAB IPE (Integrated Programming Environment), seleccioné el microcontrolador PIC16F877A de la lista, y cargué el archivo .hex que había generado previamente con el código.
- ☐ Grabación del Código: Con todo conectado, hice clic en el botón de Program en MPLAB IPE. El PICkit 3 comenzó a transferir el archivo .hex al microcontrolador. En este paso, el programador envía los datos al PIC a través de los pines de programación. El microcontrolador entra en modo de programación, gracias a la señal en el pin MCLR, lo que permite que se grabe el código en la memoria flash.
- ☐ Verificación: Una vez que la programación se completó, MPLAB IPE automáticamente verificó que los datos se hayan grabado correctamente. El software revisó que el código que se había escrito en la memoria del PIC coincidiera con el archivo .hex original.

Prueba del circuito:

El proceso para probar el circuito sería el siguiente:

1. Conectar la alimentación:
 - Conecto la fuente de 5V al circuito, asegurándome de que los pines VDD y VSS del PIC estén bien conectados a la alimentación.
2. Encender el microcontrolador:
 - Al encender la alimentación, el PIC16F877A debería arrancar automáticamente y empezar a ejecutar el código que grabé anteriormente.
3. Observar el comportamiento de los LEDs:
 - Observo los LEDs conectados a los pines RA0, RA1, RB0 y RC0. Si todo está funcionando bien, los LEDs deberían comenzar a parpadear de forma sincronizada: se encienden todos al mismo tiempo y luego se apagan, repitiendo este ciclo cada 500 ms.
4. Confirmar funcionamiento:
 - el circuito está funcionando correctamente y el código fue cargado con éxito.

Evidencia:



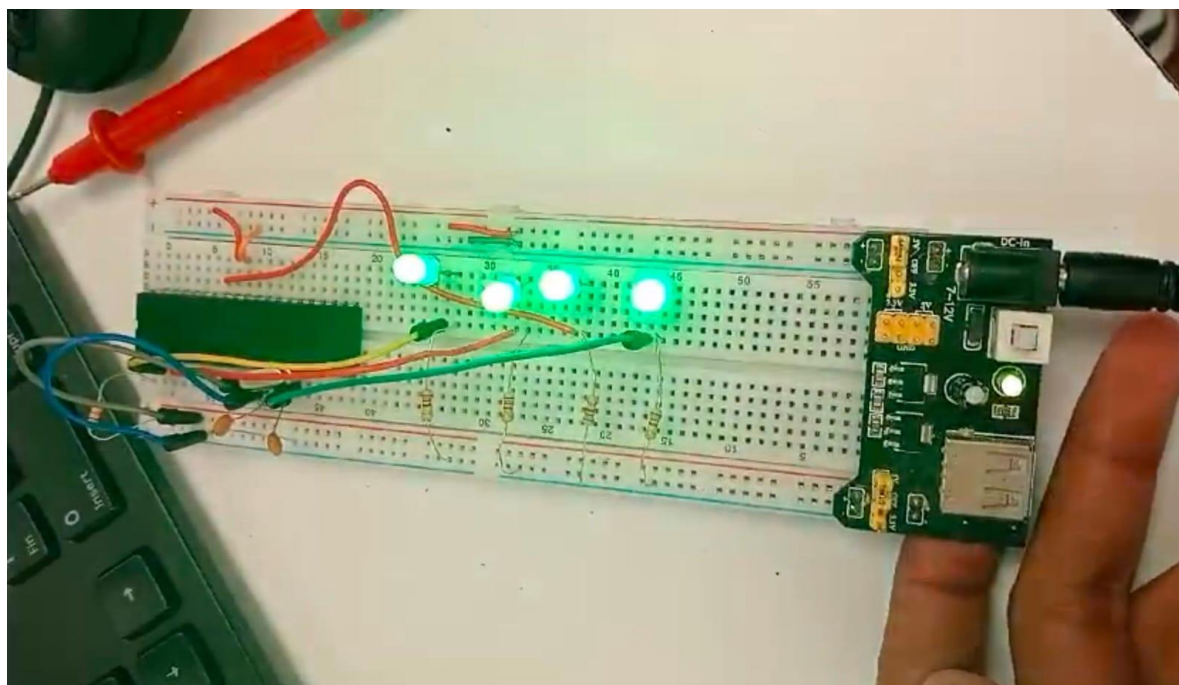
Documentación:

En la **documentación** del proyecto, lo que se hizo fue describir de manera detallada todo el proceso del proyecto, desde la planificación y el código hasta las pruebas realizadas para verificar que el circuito funciona correctamente. Esto incluyó:

1. **Objetivo del Proyecto:** Explicar brevemente lo que se quería lograr, que en este caso era controlar 4 LEDs con un microcontrolador **PIC16F877A**.
2. **Materiales Utilizados:** Listar todos los componentes necesarios, como el microcontrolador, los LEDs, las resistencias, el oscilador y el programador.

3. **Esquema del Circuito:** Describir cómo estaban conectados los componentes, como los LEDs a los pines RA0, RA1, RB0 y RC0 del microcontrolador, las conexiones de alimentación y el oscilador.
4. **Configuración de Fuses:** Explicar cómo se configuraron los fuses del microcontrolador (por ejemplo, oscilador externo, desactivación del Watchdog Timer, etc.).
5. **Código del Microcontrolador:** Incluir el código escrito en C para controlar los LEDs, donde se encienden y apagan cada 500 ms.
6. **Procedimiento de Programación:** Describir los pasos realizados para programar el microcontrolador usando el programador **PICKit 3** y el software **MPLAB IPE**.
7. **Proceso de Prueba:** Explicar cómo se conectó la fuente de alimentación, se encendió el microcontrolador y se verificó que los LEDs parpadearan según lo esperado.
8. **Conclusiones:** Resumir el éxito del proyecto, indicando que los LEDs parpadearon correctamente y que el proceso de programación y prueba fue exitoso.

Evidencia de práctica terminada:



Conclusión:

El proyecto de control de LEDs utilizando el microcontrolador PIC16F877A fue exitoso. A través del código desarrollado y la configuración adecuada del hardware, logramos que los 4 LEDs conectados a los pines del microcontrolador parpadearan de manera sincronizada, con un ciclo de encendido y apagado de 500 ms. El proceso de programación fue realizado correctamente utilizando el programador PICKit 3 y el software MPLAB IPE. Durante las pruebas, el circuito respondió según lo esperado, confirmando que tanto la programación como las conexiones de hardware fueron correctas.