

Universidad de Costa Rica

Escuela de Ingeniería Eléctrica

Laboratorio 01

# **Introducción a microcontroladores y manejo de GPIOs**

Profesor: MSc. Marco Villalta Fallas

Estudiante: Alison Rivera Cisneros

Carné: C06510

25 de marzo de 2025

# Índice

<b>1. Introducción</b>	<b>2</b>
<b>2. Nota teórica</b>	<b>2</b>
2.1. Microcontrolador PIC12f683 . . . . .	2
2.1.1. Perifericos . . . . .	3
2.1.2. Registros . . . . .	4
2.2. Conceptos teóricos . . . . .	7
2.3. Diseño del circuito . . . . .	7
2.3.1. Diseño de conexiones . . . . .	9
2.3.2. Diseño de componentes . . . . .	11
2.3.3. Presupuesto . . . . .	12
<b>3. Análisis de resultados</b>	<b>13</b>
3.1. Análisis de programación . . . . .	13
3.2. Análisis electrónico . . . . .	15
<b>4. Conclusiones y recomendaciones</b>	<b>16</b>
<b>5. Anexo</b>	<b>18</b>
5.1. Avences git . . . . .	18
5.2. Código para leds . . . . .	18
5.3. hojas del fabricante: . . . . .	19

# 1. Introducción

A lo largo de este documento se plantea desarrollar la creación de un dado usando el microcontrolador(MCU) PIC12f683 el cual se programara de modo que al tocar un botón este encienda una serie de Leeds simulando la cara de un dado. Este resultado debe darse de manera aleatoria la cual se definirá de manera programable, el desarrollo se lleva a cabo en el simulador Simulade el cual permitió un mayor entendimiento del manejo de registros y GPIOs logrando resultados exitosos en la implementacion del programa.

## 2. Nota teórica

Para el correcto entendimiento de este laboratorio es necesario considerar una serie de información técnica, de procesos y conceptos teóricos que se discutirán a continuación:

### 2.1. Microcontrolador PIC12f683

Como se menciona en la introducción el microcontrolador a emplear hacer referencia al PIC12f683 basado en una arquitectura RISC (Reduced Instruction Set Computing) el cual es un microcontrolador de 8 bits de la empresa Microchip, diseñado para aplicaciones compactas y encapsulados pequeños como PDIP, SOIC y DFN-S [1]. Este microcontrolador cuenta con una memoria Flash, RAM y EEPROM, también integra periféricos como temporizadores, un comparador analógico y un módulo de comunicación por modulación de ancho de pulso (PWM) [1]. En cuanto a sus características eléctricas, el PIC12F683 opera con un voltaje de alimentación que va desde 2.0 V hasta 5.5 V. En adición, consume una poca corriente en modo activo (entre 220  $\mu$ A y 1.6 mA) y un corriente extremadamente bajo en modo no activo (aproximadamente 100 nA). Además, cuenta con un oscilador interno ajustable de 8 MHz[1] y su diagrama de bloques se puede apreciar en la figura 1.

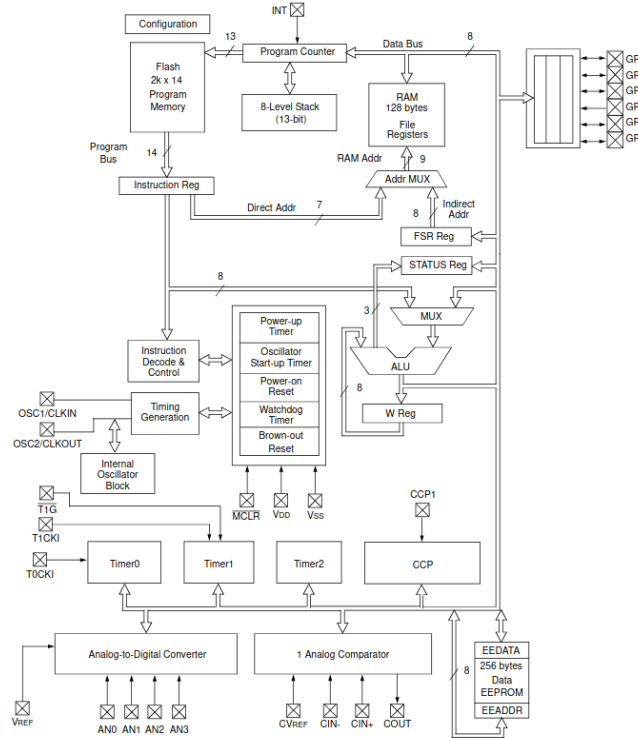
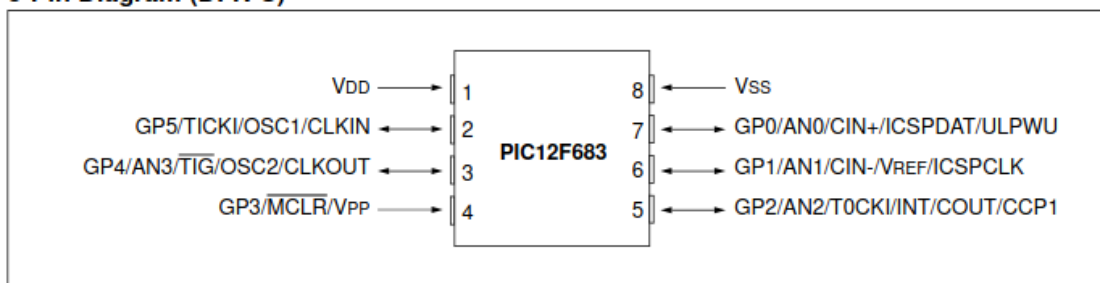


Figura 1: PIC12f683 diagrama de bloques [1]

### 2.1.1. Perifericos

Donde a detalle sus perifericos hacen referencia a:

- **GPIO (General Purpose Input/Output):** Consta de un total de 6 pines configurables como entradas o salidas digitales, a excepci3n del 3 el cual solo es para input, como se destaca en la 2 (de los 8 pines totales, 2 son para alimentaci3n) 1 [1].



**TABLE 1: 8-PIN SUMMARY**

I/O	Pin	Analog	Comparators	Timer	CCP	Interrupts	Pull-ups	Basic
GP0	7	AN0	CIN+	—	—	IOC	Y	ICSPDAT/ULPWU
GP1	6	AN1/VREF	CIN-	—	—	IOC	Y	ICSPCLK
GP2	5	AN2	COUT	T0CKI	CCP1	INT/IOC	Y	—
GP3 <sup>(1)</sup>	4	—	—	—	—	IOC	Y <sup>(2)</sup>	MCLR/VPP
GP4	3	AN3	—	TIG	—	IOC	Y	OSC2/CLKOUT
GP5	2	—	—	T1CKI	—	IOC	Y	OSC1/CLKIN
—	1	—	—	—	—	—	—	VDD
—	8	—	—	—	—	—	—	Vss

**Note 1:** Input only.

**2:** Only when pin is configured for external MCLR.

Figura 2: PIC12f683 resumen de los pines [1]

- **Timer0:** Se refiere al temporizador de 8 bits con pre escalador [1].
- **Timer1:** Hace referencia al temporizador de 16 bits (opcionalmente para funcionar con oscilador externo).[1]
- **Timer2:** El 3ltimo temporizador tambi3n de 8 bits se caracteriza por su postscaler y m3dulo PWM[1].
- **ADC (Conversor Anal3gico-Digital):** Este ADC es de 10 bits con 4 canales de entrada anal3gica[1].
- **Comparador Anal3gico:**El comparador anal3gico cuenta con entrada positiva y negativa configurable[1].
- **PWM (Pulse Width Modulaci3n):** El m3dulo PWM de 10 bits asociados al Timer2 mencionado previamente[1].
- **Watchdog Timer (WDT):** Temporizador de vigilancia el cual se encarga de reiniciar el MCU en casode fallo[1].
- **Oscilador Interno:** Este es ajustable de 8 MHz, con opci3n de dividir la frecuencia[1].

- **Interrupciones:** Soporta múltiples fuentes de interrupción, como cambios en los pines, temporizadores y comparador analógico [1].
- **Memoria Flash:** Está para almacenamiento de programa[1].
- **RAM:** Con función para los datos volátiles[1].
- **EEPROM:** En este caso para el almacenamiento no volátil[1].

### 2.1.2. Registros

Siguiendo el tema del MCU también es importante discutir los registros disponibles, entre los cuales destacan los siguientes:

- **WREG (Working Register):** Registro de trabajo utilizado para operaciones aritméticas y lógicas [1].
- **STATUS:** Registro que contiene flags de estado como Carry (C), Zero (Z), y bits de configuración del banco de memoria [1].
- **FSR (File Select Register):** Registro utilizado para direccionamiento indirecto de la memoria RAM [1].
- **INDF (Indirect File Register):** Registro que permite acceder a la memoria RAM a través del FSR [1].
- **PCL (Program Counter Low):** Parte baja del contador de programa (PC) [1].
- **PCLATH (Program Counter Latch High):** Parte alta del contador de programa, usado para saltos y llamadas a subrutinas [1].
- **PORTA:** Registro que controla el estado de los pines de entrada/salida (GPIO) [1].
- **TRISA:** Registro que configura la dirección (entrada/salida) de los pines de PORTA [1].
- **OPTION\_REG:** Registro que configura opciones como el preescalador del Timer0 y las resistencias de pull-up [1].
- **INTCON:** Registro de control de interrupciones, habilita/deshabilita interrupciones globales y específicas [1].
- **PIE1 (Peripheral Interrupt Enable 1):** Habilita interrupciones de periféricos como el Timer1 y comparador analógico [1].
- **PIR1 (Peripheral Interrupt Register 1):** Registro de flags de interrupción de periféricos [1].
- **TMR0:** Registro del Timer0, utilizado para contar eventos o generar retardos [1].
- **TMR1L y TMR1H:** Registros de 16 bits del Timer1 (parte baja y alta) [1].
- **TMR2:** Registro del Timer2, utilizado para generar PWM y retardos [1].
- **ADRESH y ADRESL:** Registros de resultado del ADC (parte alta y baja) [1].
- **ADCON0:** Registro de control del ADC, habilita el módulo y selecciona el canal [1].

- **CCPR1L y CCPR1H:** Registros de captura y comparación para el módulo PWM [1].
- **CCP1CON:** Registro de control del módulo CCP (Capture/Compare/PWM) [1].
- **WDTCON:** Registro de control del Watchdog Timer [1].
- **OSCCON:** Registro de control del oscilador interno [1].
- **EEADR y EEDATA:** Registros para acceder a la memoria EEPROM (dirección y dato) [1].
- **EECON1 y EECN2:** Registros de control para la escritura/lectura de la EEPROM [1].

Estos también se aprecian de forma gráfica mapeados en memoria en la figura 3.

File Address		File Address	
Indirect addr. <sup>(1)</sup>	00h	Indirect addr. <sup>(1)</sup>	80h
TMR0	01h	OPTION_REG	81h
PCL	02h	PCL	82h
STATUS	03h	STATUS	83h
FSR	04h	FSR	84h
GPIO	05h	TRISIO	85h
	06h		86h
	07h		87h
	08h		88h
	09h		89h
PCLATH	0Ah	PCLATH	8Ah
INTCON	0Bh	INTCON	8Bh
PIR1	0Ch	PIE1	8Ch
	0Dh		8Dh
TMR1L	0Eh	PCON	8Eh
TMR1H	0Fh	OSCCON	8Fh
T1CON	10h	OSCTUNE	90h
TMR2	11h		91h
T2CON	12h	PR2	92h
CCPR1L	13h		93h
CCPR1H	14h		94h
CCP1CON	15h	WPU	95h
	16h	IOC	96h
	17h		97h
WDTCN	18h		98h
CMCON0	19h	VRCON	99h
CMCON1	1Ah	EEDAT	9Ah
	1Bh	EEADR	9Bh
	1Ch	EECON1	9Ch
	1Dh	EECON2 <sup>(1)</sup>	9Dh
ADRESH	1Eh	ADRESL	9Eh
ADCON0	1Fh	ANSEL	9Fh
General Purpose Registers 96 Bytes	20h	General Purpose Registers 32 Bytes	A0h
			BFh
			C0h
			EFh
		Accesses 70h-7Fh	F0h
	7Fh		FFh
BANK 0		BANK 1	



Unimplemented data memory locations, read as '0'.

**Note 1:** Not a physical register.

Figura 3: PIC12F683 mapeo de banco de memoria  
[1]

## 2.2. Conceptos teóricos

- **Estados flotantes en entradas digitales:** Cuando un pin de entrada de un MCU no está conectado a un nivel lógico definido (VDD o GND), puede adoptar un estado flotante susceptible a ruido eléctrico, generando lecturas aleatorias. Para evitarlo se usan resistencias pull-up/pull-down que fijan el estado en reposo [2].
- **Escritura de bits/registros en C para MCUs:** En programación embebida, los registros del MCU se manipulan mediante operaciones bitwise. [3].
- **Rebote mecánico en interruptores:** Fenómeno donde los contactos metálicos de un botón generan múltiples transiciones rápidas (5-50ms) al presionarse. Se mitiga con filtros hardware (RC) o software (retardos debounce algorithms) para evitar falsas detecciones [4].
- **LFSR (Linear-Feedback Shift Register):** Es un tipo de registro de desplazamiento en el que el valor de cada bit se calcula a partir de una función lineal, generalmente una combinación XOR de ciertos bits del registro. Se utiliza en la generación de secuencias pseudo-aleatorias, codificación de datos y criptografía. Los bits del registro se desplazan hacia la derecha (o izquierda) en cada ciclo, y el bit de salida se retroalimenta para crear la nueva secuencia [5].
- **Configuración de GPIOs:** Los pines de propósito general requieren inicialización: establecer dirección (TRISx), estado inicial (LATx), y en algunos casos resistencias internas (WPU). Por ejemplo, `TRISA = 0b00001000` configura el pin RA3 como entrada [1].

## 2.3. Diseño del circuito

Para esta sección se explorará el desarrollo del diseño del circuito en donde primeramente se desarrollaran los componentes electrónicos complementarios a utilizar:

- **Diodo emisor de luz (LED):** Componente optoelectrónico que emite luz al ser polarizado en directa, utilizado como indicador visual en circuitos electrónicos [6] este se apreciara en la simulación como se observa en la tabla comparativa 1.


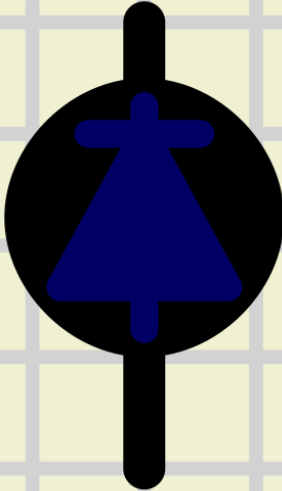
Componente físico	Componente Simulide
	

Tabla 1: Tabla comparativa Diodo emisor de luz



- **Resistencias:** Elementos pasivos que limitan el flujo de corriente en un circuito, protegiendo componentes sensibles como LEDs [7].


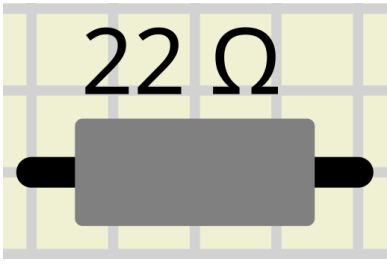
Componente físico	Componente Simulide
	

Tabla 2: Tabla comparativa Resistencia

- **Botón:** Interruptor momentáneo que permite o interrumpe el flujo de corriente al ser presionado, útil para entradas de usuario [8].


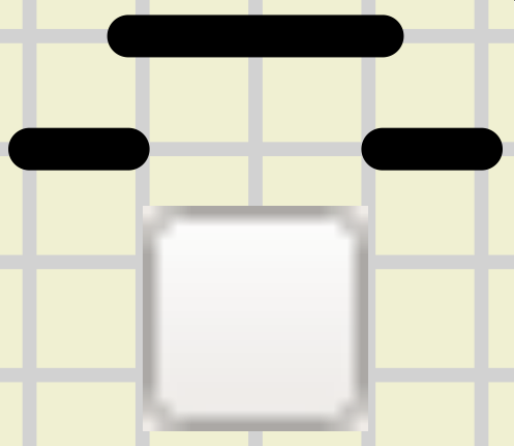
Componente físico	Componente Simulide
	

Tabla 3: Tabla comparativa boton

- **Capacitor:** Almacena energía eléctrica en un campo electrostático, usado para filtrar ruido, estabilizar voltajes o temporizar circuitos [9].


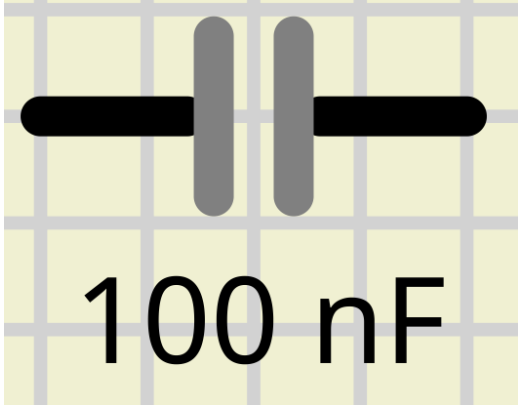
Componente físico	Componente Simulide
	

Tabla 4: Tabla comparativa capacitor

### 2.3.1. Diseño de conexiones

Para el diseño inicial del dado se consideró emplear 6 LEDs para ilustrar **el número resultante** en el dado, sin embargo, posteriormente se interiorizó que el requerimiento hacía referencia a ilustrar **las caras del dado** las cuales se pueden apreciar en la figura 4 en donde como mínimo se requieren 7 LEDs. De esta forma para el siguiente paso del diseño se analizó la lógica en las caras del dado, como se aprecia en la tabla 5, donde a partir de la evolución de casos se identificaron ciertos pares de LEDs que independientemente de la cara si uno encendía el otro también, permitiendo identificar las conexiones en serie.



Figura 4: Caras posibles de un dado



no está presionado, evitando estados flotantes que pueden resultar en lecturas aleatorias por ruido eléctrico y paralelo a esta resistencia agregó un capacitor para filtrar el rebote mecánico del botón, eliminando transiciones falsas al presionar o soltar [10]. Los valores requeridos para estos componentes se discutirán en la siguiente sección.

### 2.3.2. Diseño de componentes

En secciones anteriores se discutieron los componentes requeridos en la implementación de laboratorio, cuyos valores elegidos se demostraran a continuación:

- **Diseño de resistencia "pull-down":** Para este componente se diseña un valor típico de 10k  $\Omega$

- **Diseño del capacitor:**

1. Se diseña un tiempo para filtro de rebote de 1ms.
2. Usando la constante de tiempo  $RC$ :

$$\tau = R \times C$$

3. Si  $R = 10\text{ k}\Omega$  y se desea  $\tau = 10\text{ ms}$ :

$$C = \frac{\tau}{R} = \frac{0,01}{10,000}$$

4. Se sustituye y se calcula un valor de 1 nF

- **Resistor de protección:**

1. Se aplica la ley de Ohm y de kirchhoff para obtener la formula que resulta en R:

$$R = \frac{V_{\text{out}} - V_{\text{PIC}} - (n(V_{\text{LED}}))}{I_{\text{safe}}}$$

Donde:

- $V_{\text{LED}}$ : se refiere a la tensión a la que el LED empieza a conducir (Threshold).
  - $I_{\text{safe}}$ : Corriente segura para el LED.
  - $V_{\text{out}}$ : Tensión de la fuente.
  - $V_{\text{PIC}}$ : Caída de tensión por el MCU.
2. Se sustituyen los valores en la fórmula según el LED a emplear en el circuito que se aprecian en la 6 y en la hoja del fabricante del MCU [1].

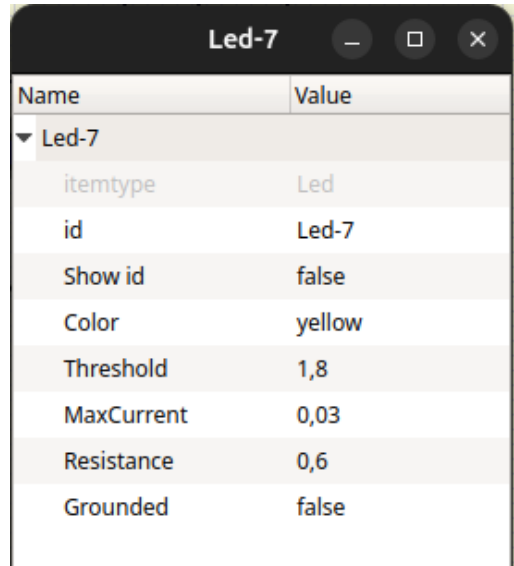
$$R = \frac{5V - 0,3V - ((1)(2,4V))}{0,02A} = 145\Omega \text{ se aproxima al valor comercial } 150\Omega$$

Donde

- $V_{\text{LED}} = 1.8\text{ V}$ .
- $I_{\text{safe}}$ : La máxima que soporta el LED es de 0.03 A, pero para el GPIO es menor, por lo que se utilizara 0.02 A por seguridad.
- $V_{\text{out}}$ : Tensión para el MCU igual a 5V.

- $V_{PIC}$ : 0.3 V.
3. Se aproxima el valor obtenido a uno comercial de 150  $\Omega$  para 1 LED
  4. Se repite el proceso para los LED que estarán en configuración paralela, modificando la corriente a su valor máximo seguro para el GPIO, ya que esta se tendrá que dividir en dos LEDs de modo que:

$$R = \frac{5V - 0,3V - ((2)(2,4V))}{0,02A} = 55\Omega ohm \text{ se aproxima al valor comercial } 56\Omega$$



Name	Value
▼ Led-7	
itemtype	Led
id	Led-7
Show id	false
Color	yellow
Threshold	1,8
MaxCurrent	0,03
Resistance	0,6
Grounded	false

Figura 6: Información de LED a utilizar

### 2.3.3. Presupuesto

Tomando en cuenta los componentes anteriores, el presupuesto para el desarrollo del proyecto se desglosa en la tabla 6. Los precios fueron consultados en distribuidores locales y sitios web especializados [11, 12, 13].

Componente	Cantidad	Precio unitario (CRC)	Subtotal (CRC)
LED rojo 5mm	7	150	1,050
Resistencia 150 $\Omega$	1	25	25
Resistencia 56 $\Omega$	3	25	75
Resistencia 10k $\Omega$	1	25	25
Capacitor cerámico 1nF	1	100	100
Botón pulsador	1	300	300
PIC12F683	1	1,500	1,500
Protoboard	1	2,500	2,500
Cables dupont	10	50	500
Fuente 5V	1	3,000	3,000
<b>Total</b>			<b>9,075</b>

Tabla 6: Presupuesto detallado del proyecto

### 3. Análisis de resultados

#### 3.1. Análisis de programación

El programa desarrollado sigue una lógica estructurada como se muestra en el diagrama de flujo de la figura 7 donde para simular el lanzamiento de un dado controlado por un botón para generar números aleatorios que simulan los resultados de un dado, con valores entre 1 y 6. Al presionar el botón, se genera un número aleatorio y se encienden LEDs en diferentes configuraciones, según el número obtenido. La función esperar introduce un retraso antes de apagar los LEDs. El LFSR (Linear-Feedback Shift Register) modifica el valor del número aleatorio a través de desplazamientos, garantizando una secuencia pseudoaleatoria. Este generador de números pseudoaleatorios es basado en un LFSR [5], cuya operación se detalla en la figura 8 donde se manipula el valor de *\*numero\_aleatorio*. Si el bit más bajo del número es 1, se desplaza a la derecha y se aplica un XOR con los bits de las posiciones 15, 14, 12 y 3 del registro. Si el bit más bajo es 0, solo se realiza el desplazamiento sin retroalimentación. El valor resultante se usa para calcular un número aleatorio entre 1 y 6, simulando el lanzamiento de un dado. Este enfoque fue seleccionado por su bajo consumo de recursos computacionales, ideal para el PIC12F683 [14].

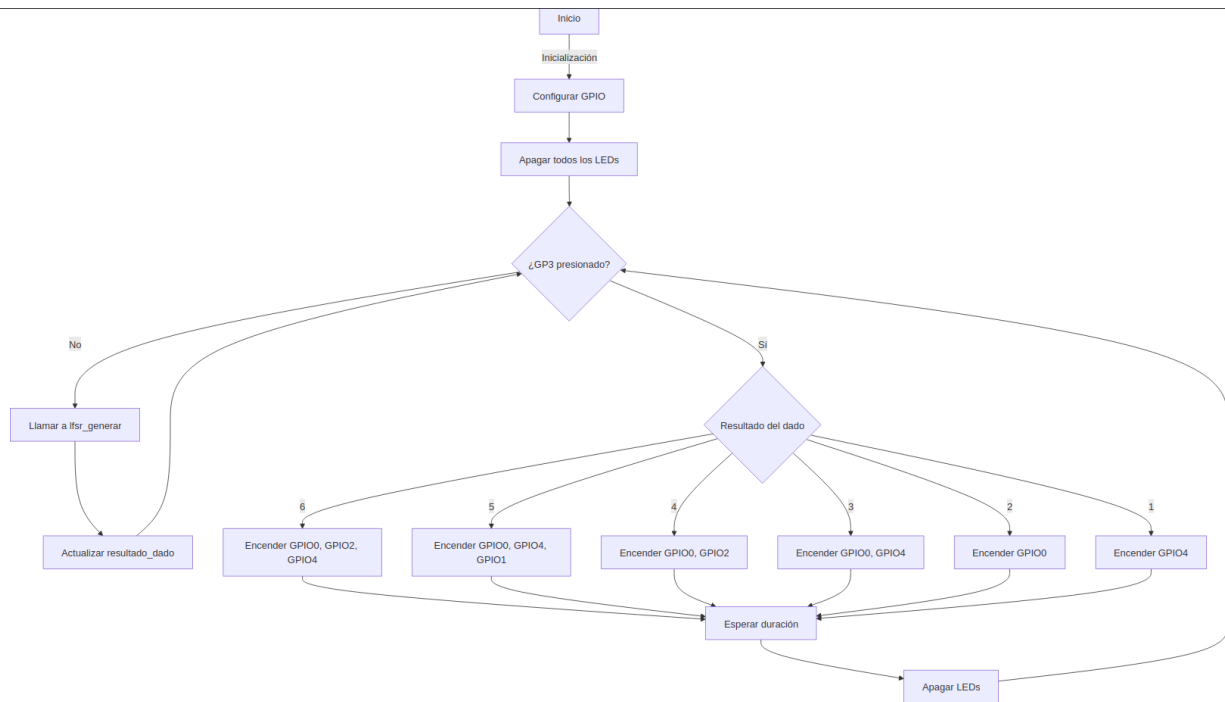


Figura 7: Lógica de programa

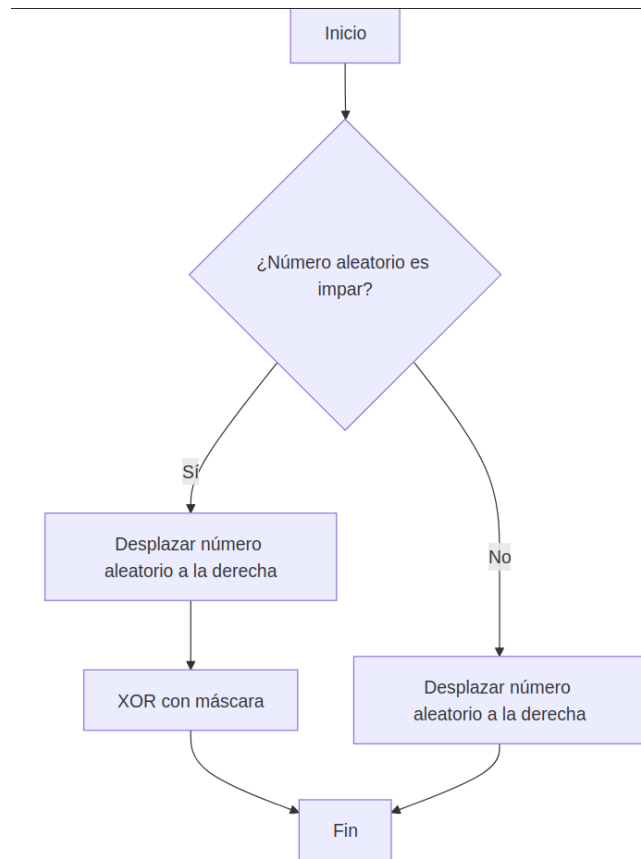


Figura 8: Lógica de LFSR (linear-feedback shift register)

La secuencia de operación del programa se verificó mediante pruebas unitarias, donde se confirmó que todas las caras del dado eran ilustradas de manera aleatoria como se aprecia en la tabla ?? con el circuito ya reacomodado para ilustrar de manera apropiada cada una de las caras previamente estudiadas para la configuración en serie de los LEDs.


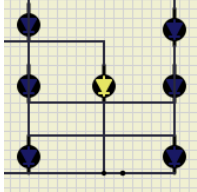

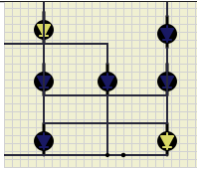

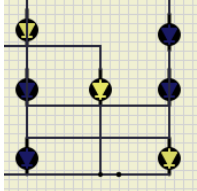

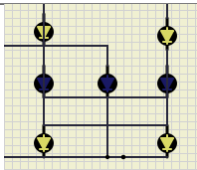

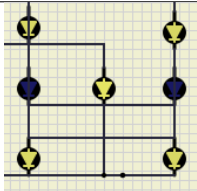

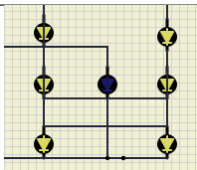
Cara del dado	Combinación de LEDs
	
	
	
	
	
	

Tabla 7: Tabla de resultados

### 3.2. Análisis electrónico

Para esta sección se realizó una serie de mediciones claves al momento de tener el programa en funcionamiento, cuyos detalles se pueden apreciar en la figura 9. En primera instancia se destaca el correcto funcionamiento del botón a 5 V al GPIO3 configurado como entrada, esto ayudo a comprobar la caída que sufre la tensión pasar por el MCU la cual el fabricante detallo ser de 0.3 V, sin embargo, con la medición a la salida de GPIO0 se calcula que el simulador maneja una caída de 0.58 muy cercano al valor teórico esperado. Por otra parte, se aprecia como el cálculo de las resistencias fue el apropiado denotando como la corriente de GPIO2 y GPIO4 no sobre pasa el límite de 0.02A, agregado a esto se confirmaron las caídas de tensiones de los LEDs aproximadas al valor teórico de 1.8 V.



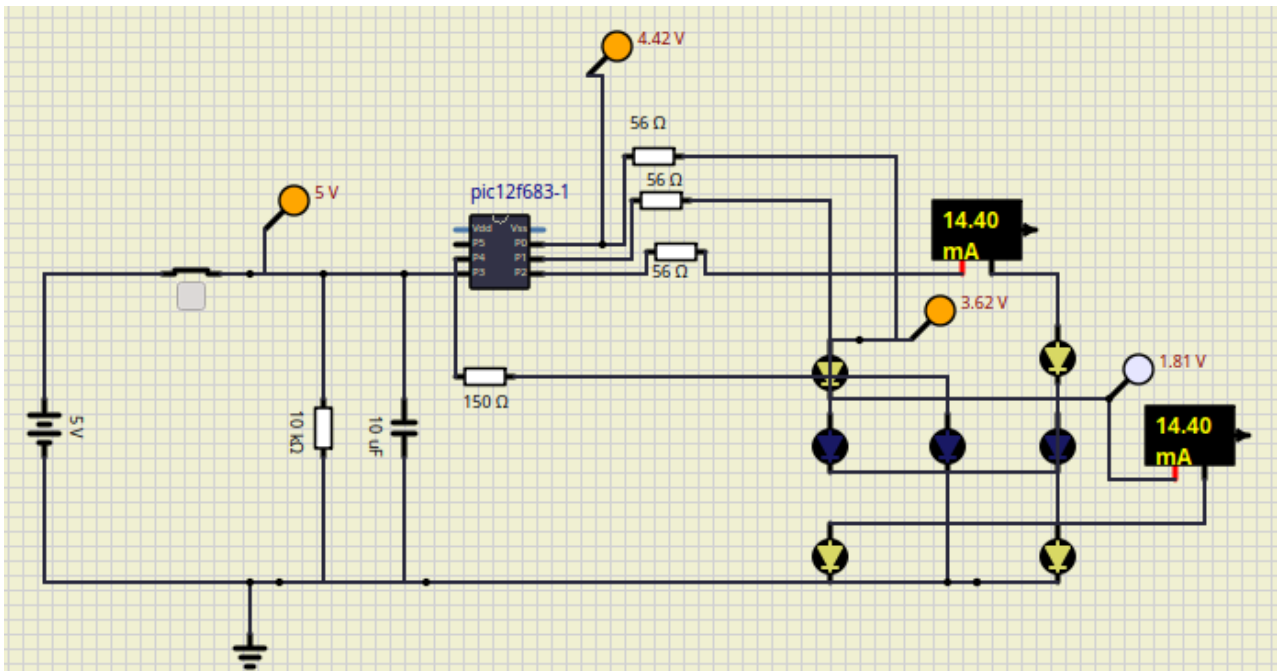


Figura 9: Mediciones del circuito

## 4. Conclusiones y recomendaciones

- El proceso de selección de componentes, como los LEDs, resistencias, y capacitores, fue clave para garantizar el funcionamiento correcto del circuito. Se validaron los valores teóricos de las resistencias y las caídas de tensión en los LEDs, comprobando que el diseño cumpliera con los requisitos establecidos para el proyecto.
- La implementación de un generador de números aleatorios mediante el uso de un LFSR (Linear-Feedback Shift Register) demostró ser una solución eficiente en términos de recursos computacionales, permitiendo la simulación de un dado con el PIC12F683. Las pruebas unitarias confirmaron que el programa generaba resultados aleatorios adecuados, activando los LEDs correspondientes según la cara del dado.
- El botón, configurado como entrada con un resistor "pull-down" y un capacitor para filtrar el rebote, funcionó correctamente. El sistema fue capaz de detectar de manera fiable las presiones del botón, evitando interferencias causadas por transiciones erráticas debido al rebote mecánico.
- El análisis de las mediciones de voltaje y corriente, junto con la validación de las caídas de tensión en los pines del MCU, confirmaron que el circuito operaba dentro de los parámetros de seguridad para los LEDs y otros componentes, sin sobrepasar los límites de corriente establecidos para el PIC12F683, garantizando la fiabilidad del diseño a largo plazo.

## Referencias

- [1] Microchip Technology Inc. Pic12f683 data sheet: 8-pin flash-based, 8-bit cmos microcontrollers with nanowatt technology. Accedido el 21 de marzo de 2025, 2007.
- [2] Microchip Technology. Pic18fxxk input configuration. Accedido el 25 de marzo de 2025, 2025.
- [3] MikroElektronika. Bit operations in mikroC for pic. Accedido el 25 de marzo de 2025, 2025.
- [4] Embedded Artists. Debouncing techniques in embedded systems. Accedido el 25 de marzo de 2025, 2025.
- [5] Maxim Integrated. Linear feedback shift register (lfsr) application note. Accedido el 25 de marzo de 2025, 2025.
- [6] Electronics Tutorials. Diode clipping circuits. Accedido el 25 de marzo de 2025, 2025.
- [7] All About Circuits. How voltage, current, and resistance relate. Accedido el 25 de marzo de 2025, 2025.
- [8] Arrow Electronics. Electronic components and products. Accedido el 25 de marzo de 2025, 2025.
- [9] Khan Academy. Capacitors. Accedido el 25 de marzo de 2025, 2025.
- [10] All About Circuits. Switch bounce: How to deal with it. Accedido el 25 de marzo de 2025, 2025.
- [11] Electrocomponentes. Venta de componentes electrónicos. Accedido el 25 de marzo de 2025, 2025.
- [12] Intelec-Tro. Distribuidor de electrónica y tecnología. Accedido el 25 de marzo de 2025, 2025.
- [13] Tienda Electrónica. Componentes y material electrónico. Accedido el 25 de marzo de 2025, 2025.
- [14] Microchip Technology. Lfsr application for pic. Accedido el 25 de marzo de 2025, 2025.

## 5. Anexo

### 5.1. Avences git

:<https://github.com/AlisonR21/Microcontroladores-Labs/commits/main/>

### 5.2. Código para leds

```
1 #include <pic14/pic12f683.h>
2
3 unsigned int __at 0x2007 __CONFIG = (_MCLRE_OFF&_WDTE_OFF);
4
5 void esperar(unsigned int tiempo_espera);
6 void lfsr_generar(unsigned short *numero_aleatorio);
7
8 void main(void) {
9
10     unsigned short valor_aleatorio = 1;
11     unsigned int duracion = 3000;
12     unsigned short resultado_dado = 0;
13     unsigned short *puntero_aleatorio = &valor_aleatorio;
14
15     TRISIO = 0b00001000; // GPIO0, GPIO1, GPIO2 y GPIO4 como salidas;
16     //GPIO3 como entrada
17     GPIO = 0b00000000; // Apagar todos los LEDs al inicio
18
19     while (1) {
20         if (GP3) {
21             switch (resultado_dado) {
22                 case 6: // Se encienden 3 pares de LEDs (3x2=6)
23                     GPIO = 0b00001101; // GPIO0, GPIO2 y GPIO4 encendidos
24                     // Apagar los LEDs:
25                     esperar(duracion);
26                     GPIO = 0b00000000;
27                     break;
28                 case 5: // Se encienden 2 pares de LEDs y el
29                     \\central (2x2+1=5)
30                     GPIO = 0b00010101; // Pares GPIO0 y GPIO2 encendidos
31                     \\y central GPIO4
32                     // Apagar los LEDs:
33                     esperar(duracion);
34                     GPIO = 0b00000000;
35                     break;
36                 case 4: // Se encienden 2 pares de LEDs (2x2=4)
37                     GPIO = 0b00000101; // Pares GPIO0 y GPIO2 encendidos
38                     // Apagar los LEDs:
39                     esperar(duracion);
40                     GPIO = 0b00000000;
41                     break;
42                 case 3: // Se encienden 1 par de LEDs y el central (2+1=3)
43                     GPIO = 0b00010001; // Par GPIO0 y central GPIO4 encendidos
44                     // Apagar los LEDs:
45                     esperar(duracion);
```

```

46         GPIO = 0b00000000;
47         break;
48     case 2: // Se enciende 1 par de LEDs (2)
49         GPIO = 0b00000001; // Par GPIO0 encendido
50         // Apagar los LEDs:
51         esperar(duracion);
52         GPIO = 0b00000000;
53         break;
54     case 1: // Se enciende el central (1)
55         GPIO = 0b00010000; // central GPIO4 encendido
56         // Apagar los LEDs:
57         esperar(duracion);
58         GPIO = 0b00000000;
59         break;
60     default:
61         break;
62     }
63     } else { //llamada a funcion (LFSR):
64         GPIO = 0b00000000;
65         lfsr_generar(puntero_aleatorio);
66         resultado_dado = 1 + (valor_aleatorio % 6); //Recalcula el valor
67     }
68 }
69 }
70
71 void esperar(unsigned int tiempo_espera) { // funcion de espera
72     unsigned int i, j;
73     for (i = 0; i < tiempo_espera; i++) {
74         for (j = 0; j < 256; j++);
75     }
76 }
77
78 void lfsr_generar(unsigned short *numero_aleatorio) { //Metodo LFSR
79     if ((*numero_aleatorio) & 1) {
80         (*numero_aleatorio) >>= 1;
81         (*numero_aleatorio) ^= (1<<15) + (1<<14) + (1<<12) + (1<<3);
82     } else {
83         (*numero_aleatorio) >>= 1;
84     }
85 }

```

### 5.3. hojas del fabricante:

width=!,height=!,pages=1-15