Universidad de Costa Rica
Facultad de Ingeniería
Escuela de Ingeniería Eléctrica
IE-0624 Laboratorio de Microcontroladores
I ciclo 2025

Laboratorio #1: Introducción a microcontroladores y manejo de GPIOS

Elizabeth Matamoros Bojorge C04652
Grupo 01

Profesor: MSc. Marco Villalta Fallas

25 de marzo de 2025

Índice

1. Introducción/Resumen

En este laboratorio se desarrolló un simulador de dado electrónico utilizando el microcontrolador PIC12F683. El objetivo principal fue generar, de manera pseudoaleatoria, un número entre 1-6 y reflejarlo a través de la activación de distintos LEDs. Para ello, se empleó un botón como entrada, el cual, al ser presionado, dispara el proceso de generación y despliega el valor aleatorio en el conjunto de LEDs.

El circuito se diseñó con el fin de garantizar la correcta alimentación del microcontrolador y la protección de los LEDs. En el mismo, se incluyó un resistor pull-up en la línea MCLR para estabilizar el arranque del PIC, además de resistencias en serie con cada LED para limitar la corriente y preservar la integridad tanto de los LEDs como del propio microcontrolador. Además, el código en C, se encargó de la lógica de generación de números, del control de tiempos mediante retardos y del encendido selectivo de los pines de salida que corresponden a cada cara del dado.

Como conclusiones importantes, se evidenció que la integración del software y el hardware (simulación) cumple con el objetivo de simular un dado funcional de seis caras. Asimismo, el laboratorio permitió consolidar conocimientos sobre la configuración de pines de entrada y salida, la manipulación de interrupciones de estado para la lectura de un botón y el uso de generadores pseudoaleatorios en aplicaciones de microcontroladores.

Finalmente, se puede visitar el repositorio en donde se desarrolló el laboratorio en el siguiente enlace: `https://github.com/elizamatambojor/LabMicrocontroladores.git`

2. Nota Teórica

2.1. Información general MCU

El PIC12F683 es un microcontrolador de 8 bits desarrollado por la empresa Microchip Technology. Pertenece a la familia PIC de arquitectura RISC, lo que significa que está diseñado con un conjunto reducido de instrucciones que permiten una ejecución eficiente y rápida. Este microcontrolador es ideal para proyectos donde se necesita una lógica de control sencilla, entradas/salidas digitales, conversión analógica-digital, y bajo consumo de energía [1].

Por otra parte, el PIC12F683 incorpora un núcleo RISC que ejecuta la mayoría de instrucciones en un solo ciclo de reloj. Opera con una frecuencia máxima de hasta 20 MHz y posee 2048 palabras de memoria de programa tipo Flash, lo que permite su reprogramación cuantas veces sea necesario. También cuenta con 128 bytes de memoria RAM y 256 bytes de memoria EEPROM [1].

Además, el PIC12F683 ofrece un total de 6 pines de entrada/salida (GPIOs), los cuales pueden ser configurados individualmente como entradas o salidas digitales. Esta característica resulta fundamental para la interacción con otros dispositivos como sensores, botones, LEDs, entre otros. Además, incluye temporizadores internos que permiten realizar tareas relacionadas

con conteo, generación de retardos o control de eventos temporales [1]. Es importante recalcar que el pin GP3 se utiliza únicamente como entrada.

Adicionalmente, otra ventaja importante es su eficiencia energética, ya que utiliza tecnologías como nanoWatt, optimizadas para reducir el consumo en aplicaciones alimentadas por baterías [1][2].

2.2. Diagrama de bloques y de pines

A continuación se muestra el diagrama de bloques y diagrama de pines encontrado en la hoja de datos del fabricante [1].

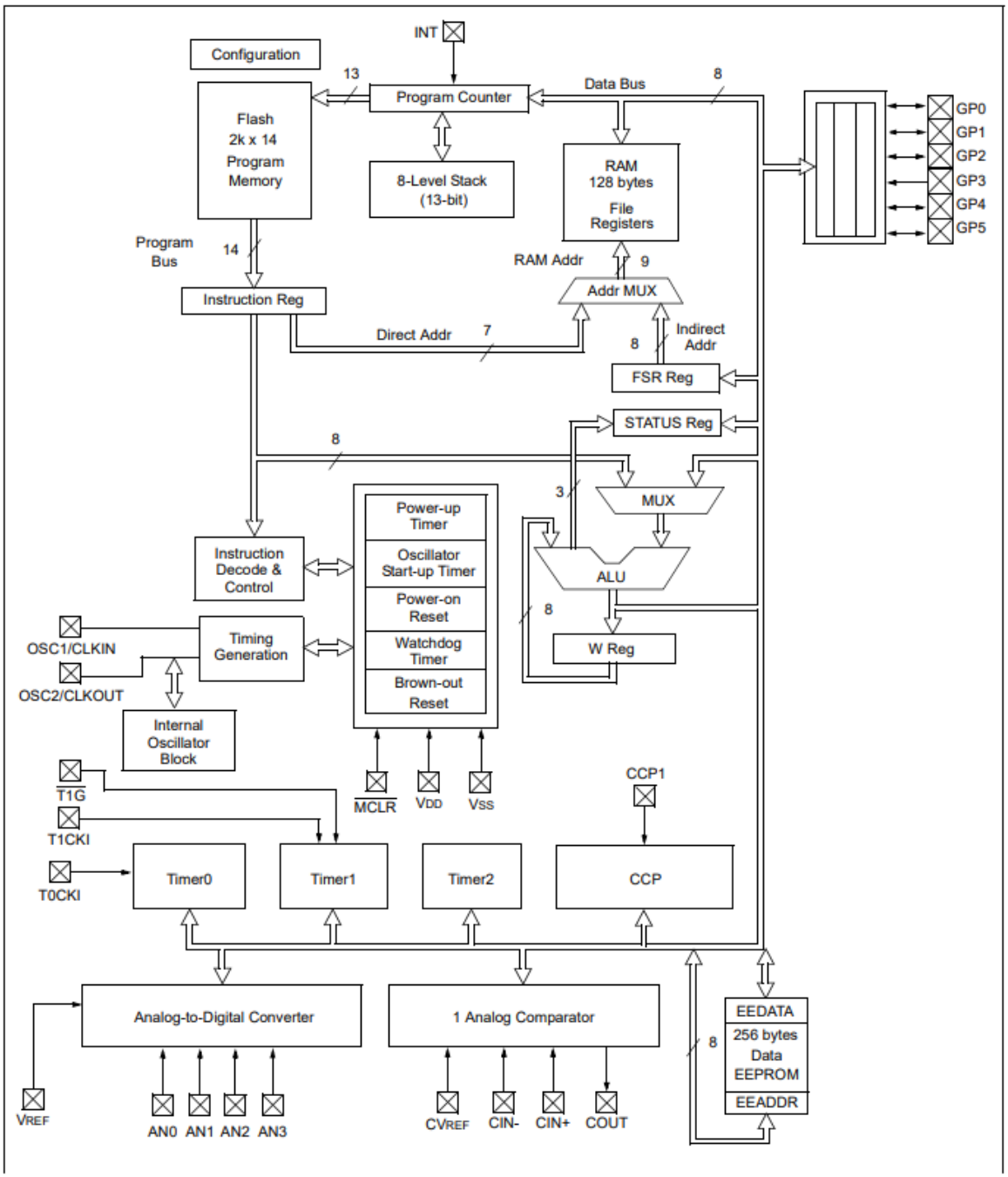

Figura 1: Diagrama de bloques del PIC12F683 [1]

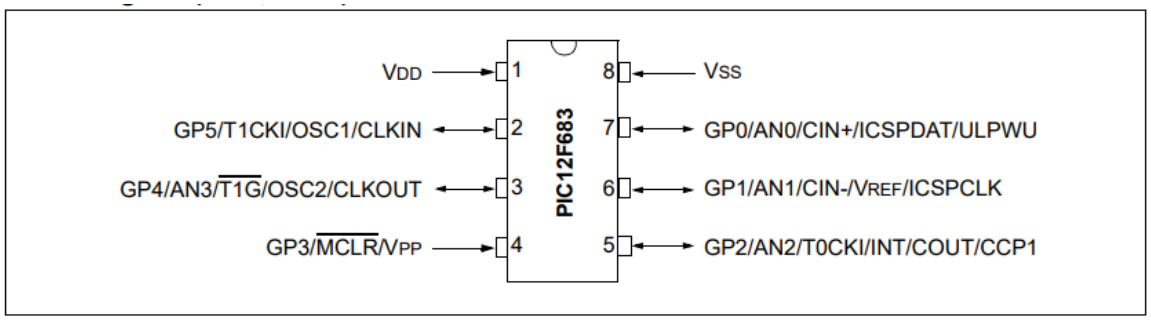

Figura 2: Diagrama de pines del PIC12F683 [1]

2.3. Periféricos

A continuación, se detallan los principales periféricos incorporados en este microcontrolador:

El PIC12F683 cuenta con un convertidor analógico-digital de 10 bits que admite hasta 4 canales. Este ADC permite la conversión precisa de señales analógicas a digitales, facilitando la interacción con sensores y otros dispositivos analógicos [1].

Por otro lado, el microcontrolador dispone de un módulo CCP que ofrece funcionalidades de captura, comparación y generación de señales PWM [1].

Además, el PIC12F683 está equipado con dos temporizadores de 8 bits y un temporizador de 16 bits. Estos temporizadores pueden operar como temporizadores o contadores, proporcionando control preciso del tiempo en diversas aplicaciones. Son fundamentales para tareas como la generación de retardos, la medición de intervalos de tiempo y la creación de eventos periódicos [1].

Asimismo, ofrece 6 pines de entrada/salida configurables individualmente como entradas o salidas digitales. Esta flexibilidad permite la interacción con una amplia gama de dispositivos externos, incluyendo sensores, actuadores y otros componentes electrónicos, facilitando la adaptación del microcontrolador a diferentes necesidades de diseño [1].

El microcontrolador integra un oscilador interno ajustable que puede configurarse en un rango de 32 kHz a 8 MHz. Esta característica elimina la necesidad de componentes externos para la generación de reloj, simplificando el diseño del circuito [1].

También, cuenta con 256 bytes de memoria EEPROM interna, que permite el almacenamiento de datos no volátiles. Esta memoria es ideal para guardar configuraciones, parámetros de usuario o cualquier información que deba preservarse entre ciclos de encendido del dispositivo [1].

2.4. Registros

En el desarrollo de aplicaciones con microcontroladores como el PIC12F683, el uso adecuado de los registros especiales es esencial para controlar el funcionamiento del hardware. Estos registros permiten configurar tanto el comportamiento de los pines de entrada/salida como la activación de periféricos y características internas. En este laboratorio, se utilizaron principalmente los registros TRISIO, GPIO, macros individuales GPx, y el registro de configuración __CONFIG.

Uno de los registros fundamentales en este proyecto es **TRISIO**, que determina la dirección de cada pin del puerto GPIO. Cuando un bit del registro TRISIO se configura en “1”, el pin correspondiente actúa como entrada; si se configura en “0”, actúa como salida. En el diseño propuesto, se requiere que el pin GP5 actúe como entrada para detectar el estado de un botón, mientras que los demás pines funcionan como salidas digitales para controlar LEDs. Esta configuración se establece con la instrucción:

$$TRISIO = 0b0010000;$$

lo cual configura GP5 como entrada (1) y GP0–GP4 como salidas (0). Esta asignación concuerda con lo descrito en la hoja de datos del microcontrolador, donde se especifica que TRISIO es el registro que controla la dirección del puerto de entrada/salida general (GPIO) [1].

El registro **GPIO** permite leer el estado de los pines configurados como entradas y escribir niveles lógicos en los pines configurados como salidas. En este diseño, se utiliza para controlar combinaciones de LEDs que representan valores de 1 a 6, como por ejemplo:

$$GPIO = 0b00000111;$$

Esta instrucción activa simultáneamente los pines GP0, GP1 y GP2 (enciende tres LEDs). Esta técnica permite una manipulación eficiente del puerto cuando se requiere modificar varios pines al mismo tiempo. Además, para situaciones donde se quiere actuar sobre un pin específico, también se usaron las macros individuales GP0, GP1, etc., por ejemplo:

$$GP1 = 1;$$

$$GP0 = 0;$$

Estas macros son definiciones simbólicas que permiten un acceso directo a los bits individuales del registro GPIO, mejorando la legibilidad del código cuando se trata de cambios simples o puntuales [1].

Además del control de pines, se utilizó el registro de configuración especial **__CONFIG** para desactivar el Watchdog Timer (WDT). Este temporizador interno, si está activado, reinicia automáticamente el microcontrolador si no se resetea periódicamente durante la ejecución. Para evitar reinicios no deseados se desactiva mediante:

$$word\_\_at0x2007\_\_CONFIG = (\_WDTE\_OFF);$$

Esta línea configura el bit de fuse WDTE en OFF en la dirección 0x2007, según lo indicado por la hoja de datos del PIC12F683. De esta forma, se garantiza que el microcontrolador no será reiniciado por el WDT durante la ejecución del código [1].

Aunque existen muchos otros registros estos no fueron utilizados en este laboratorio. Sin embargo, es importante mencionar que estos registros permiten ampliar las capacidades del microcontrolador en futuras implementaciones más complejas.

2.5. Diseño del circuito

El circuito diseñado tiene como objetivo simular el funcionamiento de un dado electrónico. Al presionar un botón, el sistema genera un número aleatorio entre 1 y 6, representado mediante la iluminación de una combinación específica de LEDs naranjas. El diseño es el siguiente:

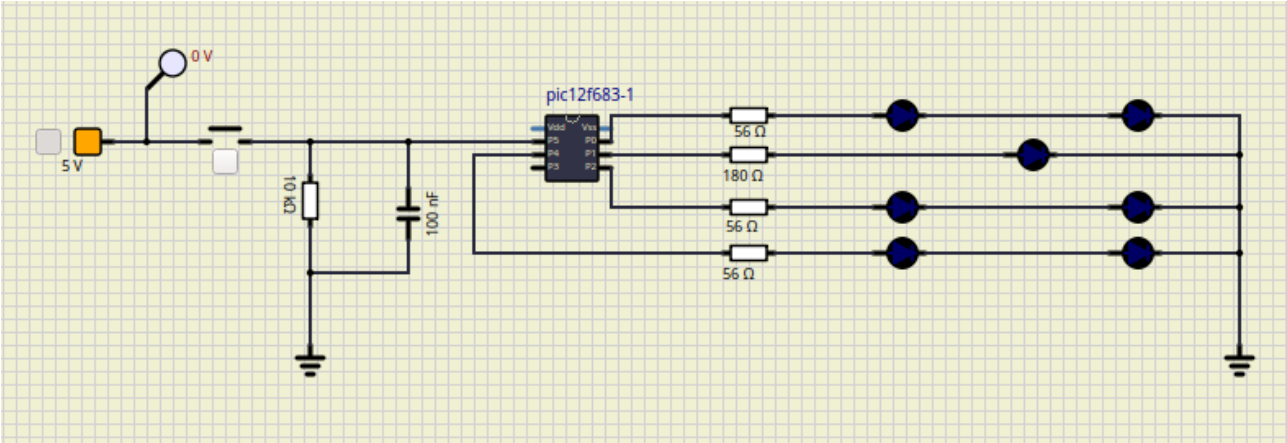

Figura 3: Diseño del circuito implementado

La visualización de los números se logró encendiendo diferentes combinaciones de LEDs, reutilizando algunos de ellos en varias configuraciones. Por ejemplo, el LED central se comparte entre las caras impares del dado, mientras que las caras pares activan pares de LEDs dispuestos simétricamente. La cara número 6 activa un tercer par de LEDs adicionales. Esta estrategia permite representar los seis valores con solo cuatro salidas digitales del microcontrolador, dejando libre una entrada para conectar el botón.

Según el datasheet del PIC12F683, cada pin puede entregar hasta 25mA, pero se estableció un límite de 20mA para proteger tanto al microcontrolador como a los LEDs [1]. Los LEDs naranjas utilizados tienen una caída de tensión típica de 2.0V, y la alimentación del circuito es de 5.0V.

Aplicando la Ley de Kirchhoff, se calcularon las resistencias necesarias para limitar la corriente:

Para dos LEDs en serie:

$$R = \frac{V_{DD} - 2 \cdot V_{LED}}{I_{Max}} = \frac{5 - 2 \cdot 2{,}0}{20\,\text{mA}} = 50\,\Omega \tag{1}$$

Para un solo LED:

$$R = \frac{V_{DD} - V_{LED}}{I_{Max}} = \frac{5 - 2{,}0}{20\,\text{mA}} = 150\,\Omega \tag{2}$$

Por lo tanto, se utilizaron resistencias de 56Ω para las salidas que controlan dos LEDs en serie, y resistencias de 180Ω para las que controlan un solo LED. Estos valores fueron seleccionados por su disponibilidad comercial y porque permiten mantener una corriente cercana a los 20mA, asegurando así el encendido visible y seguro de los LEDs. Estos valores de resistencias se escogen por conveniencia y accesibilidad del mercado.

El botón fue conectado entre una fuente de +5V y el pin GP5 del microcontrolador, que fue configurado como entrada digital. Para asegurar una lógica definida cuando el botón no se encuentra presionado, se utilizó una resistencia de pull-down de 10kΩ conectada entre GP5 y tierra. Además, se colocó un capacitor de 100nF en paralelo con la resistencia, formando un

filtro RC pasivo. Esta configuración ayuda a reducir el ruido y los rebotes mecánicos generados al presionar el botón, asegurando que el microcontrolador reciba una transición limpia de señal digital. Entonces:

$$\tau = R \cdot C = 10k\Omega \cdot 10^{-9} = 1ms$$

Este valor de τ indica que el filtro tarda aproximadamente un 1ms en responder, los típicos rebotes de botones tardan entre 1 y 10 ms, entonces al tener un valor tan bajo se logra eliminar la mayoría de esos picos.

2.6. Lista de componentes y precios

Los precios estimados de los componentes fueron obtenidos consultando sitios web de proveedores locales, como Electrónica CR [3], Micro JPM [4], y Centroniks[5].

Componente	**Precio estimado (CRC)**	**Proveedor**
PIC12F683	1010	Micro JPM / Electrónica CR
Resistencia de 56 Ω	32	Micro JPM
Resistencia de 180 Ω	155	Micro JPM / Electrónica CR
Botón	150	Electrónica CR
Resistencia de 10 kΩ	120	Centroniks
Capacitor de 100 nF	325	Electrónica CR
LED naranja	310	Centroniks

Tabla 1: Precio estimado de los componentes utilizados y sus proveedores

2.7. Conceptos del laboratorio

Los GPIOs (General Purpose Input/Output) son pines configurables de propósito general que permiten realizar operaciones de entrada o salida digital. En el PIC12F683, la mayoría de los pines son GPIOs, agrupados en un único puerto denominado GPIO, y su funcionalidad se configura mediante registros internos [6].

El registro TRISIO se utiliza para definir la dirección de cada pin. Un bit en alto (1) configura el pin correspondiente como entrada digital, mientras que un bit en bajo (0) lo configura como salida. Una vez definido el modo de operación, se emplea el registro GPIO para leer el estado lógico de los pines configurados como entradas, o para escribir niveles lógicos en los pines configurados como salidas [6].

Los pines digitales pueden escribirse o modificarse utilizando diferentes métodos en lenguaje C: mediante sobrescritura directa del registro completo, mediante enmascaramiento de bits para preservar configuraciones previas, o utilizando macros individuales como GP0 para modificar bits específicos del registro GPIO. Se recomienda el uso de enmascaramiento o macros, ya que permiten mayor control sobre los cambios sin afectar el estado del resto de los pines [6].

En cuanto a las entradas digitales, un pin puede estar en estado alto, bajo o flotante. Para evitar estados indeterminados, se recomienda el uso de resistencias de pull-up o pull-down. En

una configuración pull-down, el pin se mantiene en bajo (0) hasta que ocurre un evento que lo pone en alto (1). Esta configuración se conoce como entrada con lógica positiva [6].

Los botones mecánicos presentan un comportamiento no ideal al cambiar de estado, generando oscilaciones eléctricas conocidas como rebotes. Para mitigar este efecto, se puede utilizar un filtro pasivo RC, formado por una resistencia y un capacitor en paralelo, que ayuda a estabilizar la señal digital proveniente del botón y evitar múltiples transiciones falsas en una lectura [6].

En cuanto a las salidas digitales, existen dos formas de manejar corriente: current sourcing y current sinking. En el modo sourcing, el pin entrega corriente desde VDD hacia la carga; en el modo sinking, el pin permite el paso de corriente desde la carga hacia GND. En ambos casos, los pines están diseñados para manejar corrientes pequeñas, y se recomienda no superar los 20mA por pin para evitar daños al microcontrolador [6].

El registro CONFIG permite definir parámetros de arranque del microcontrolador, como la activación o desactivación del Watchdog Timer (WDT). Este registro no se encuentra en el espacio de memoria de programa, sino que se configura en tiempo de programación mediante macros definidas al inicio del código fuente.

3. Desarrollo/Análisis de resultados

3.1. Análisis programa

El programa implementado para este laboratorio está estructurado de forma modular, facilitando su comprensión, mantenimiento y reutilización. A continuación se analiza su funcionamiento en términos de inicialización, flujo principal, y comportamiento de funciones auxiliares.

3.1.1. Inicialización

En la parte superior del código se incluye la configuración del registro de fuses __CONFIG, donde se desactiva el Watchdog Timer (WDT) usando la macro _WDTE_OFF. Esto es importante para evitar reinicios automáticos del microcontrolador durante la ejecución del programa.

Luego, dentro de la función main(), se configura el registro TRISIO para definir la dirección de los pines: GP5 como entrada digital, y el resto como salidas. También se inicializa el puerto de salida con GPIO = 0x00, dejando todos los LEDs apagados.

3.1.2. Bucle principal

El programa entra en un bucle infinito (while(1)), donde verifica constantemente el estado del botón conectado al pin GP5. Si el botón está presionado (if (GP5)), se realiza el "lanzamiento" del dado:

1. Se genera un número aleatorio entre 1 y 6 con generateDiceNumber()

2. Se activa la combinación de LEDs correspondiente mediante handleLEDOutput().

3. Se introduce un retardo a través de customDelayAndSeed().

En caso contrario, es decir, si el botón no está presionado, se asegura que los LEDs permanezcan apagados.

3.1.3. Generación de número aleatorio

El sistema utiliza un generador pseudoaleatorio implementado por software basado en una versión modificada del algoritmo Xorshift, el cual trabaja sobre un arreglo de estado de 2 bytes (state[0], state[1]).

La función generateRandomByte() aplica rotaciones y operaciones XOR sobre el estado interno, retornando como salida la suma de ambos valores temporales. La función generateDiceNumber() filtra esta salida para asegurar que el valor esté en el rango [1, 6], repitiendo el intento si cae fuera de ese rango. Esta estrategia garantiza un comportamiento similar al de un dado físico, descartando valores inválidos sin necesidad de escalamiento ni truncamiento.

3.1.4. Encendido de LEDs

La función handleLEDOutput() recibe como parámetro un número entre 1 y 6, y enciende una combinación específica de LEDs utilizando macros (GP0, GP1) o escritura directa al registro GPIO. Por ejemplo:

- Para el valor 1: GP1 = 1;
- Para el valor 3: GPIO = 0b00000011;
- Para el valor 6: GPIO = 0b00010101;

Después de encender los LEDs, la función aplica un retardo y apaga todas las salidas, lo cual produce un parpadeo breve y controlado que simula la visualización de la cara del dado.

3.1.5. Retardo y aleatoriedad adicional

La función customDelayAndSeed() cumple un doble propósito: introduce un retardo por software, y a la vez modifica el estado interno del generador aleatorio. Durante el retardo, los valores i y j se combinan mediante operaciones XOR con el arreglo state, afectando así las próximas salidas del generador. Esto simula una "semilla" variable basada en el tiempo, lo cual evita que el dado genere siempre la misma secuencia de números al encenderse.

3.2. Análisis electrónico

El análisis electrónico se realizó utilizando el simulador SimulIDE, el cual permitió verificar el comportamiento del circuito físico diseñado para emular el funcionamiento de un dado digital. El sistema se basa en la activación de combinaciones específicas de LEDs que representan las seis posibles caras de un dado convencional.

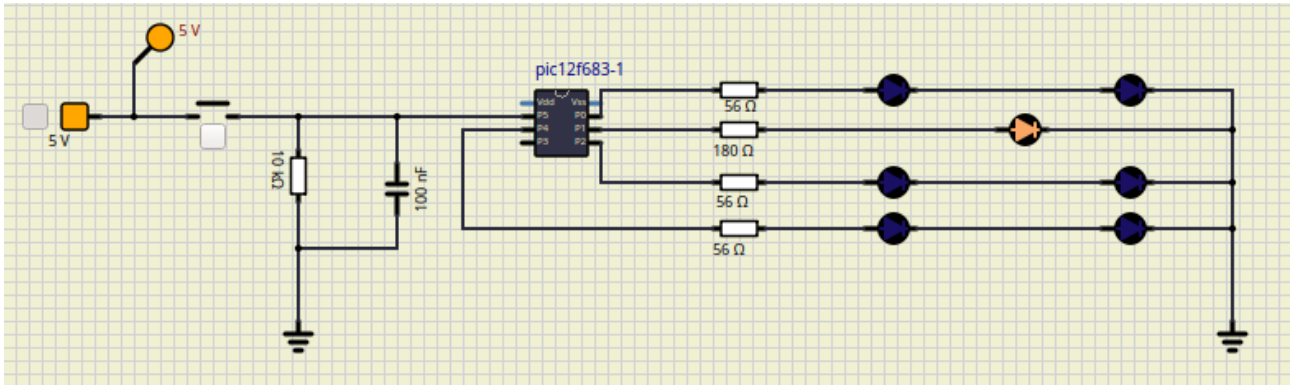

Figura 4: Cara #1 del dado

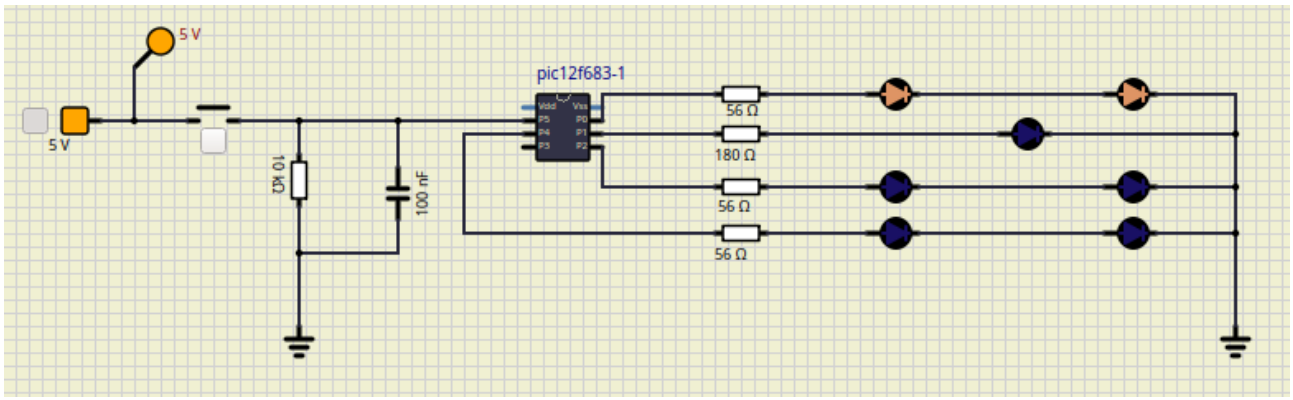

Figura 5: Cara #2 del dado

A continuación se presentan las capturas correspondientes a cada una de las seis caras, en las que se observan los LEDs encendidos y se indica el número generado. Cada imagen fue obtenida presionando el botón de entrada en el simulador, lo cual activa la lógica del programa y genera un nuevo valor pseudoaleatorio.

Las combinaciones observadas en las simulaciones coinciden con la lógica implementada en el programa fuente. Se confirma que el microcontrolador activa correctamente los pines configurados como salidas, y que los LEDs responden de forma inmediata al cambio de estado.

El botón digital simulado en GP5 permite activar el evento de “lanzamiento”, y se observa que cuando no está presionado, los LEDs se mantienen apagados. Esto concuerda con la lógica if (GP5) implementada en el código.

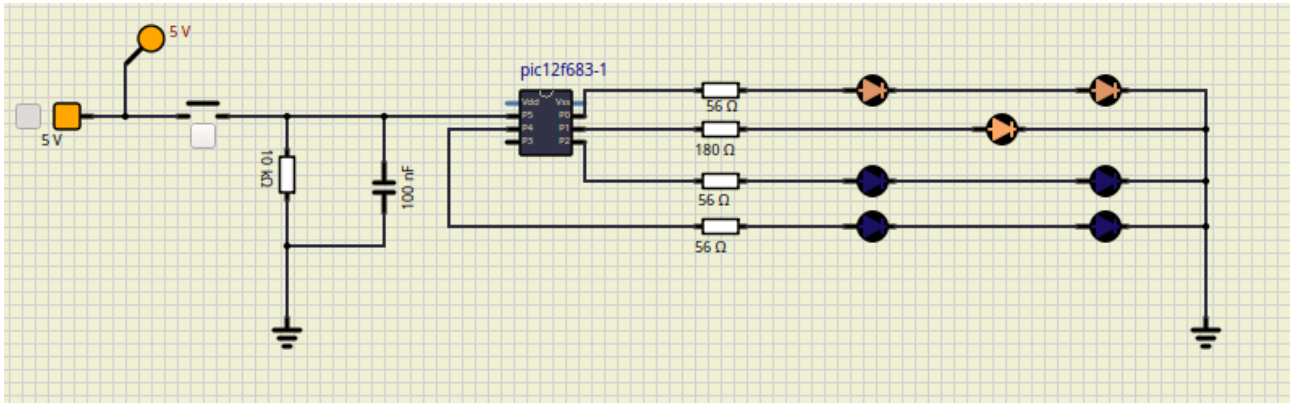

Figura 6: Cara #3 del dado

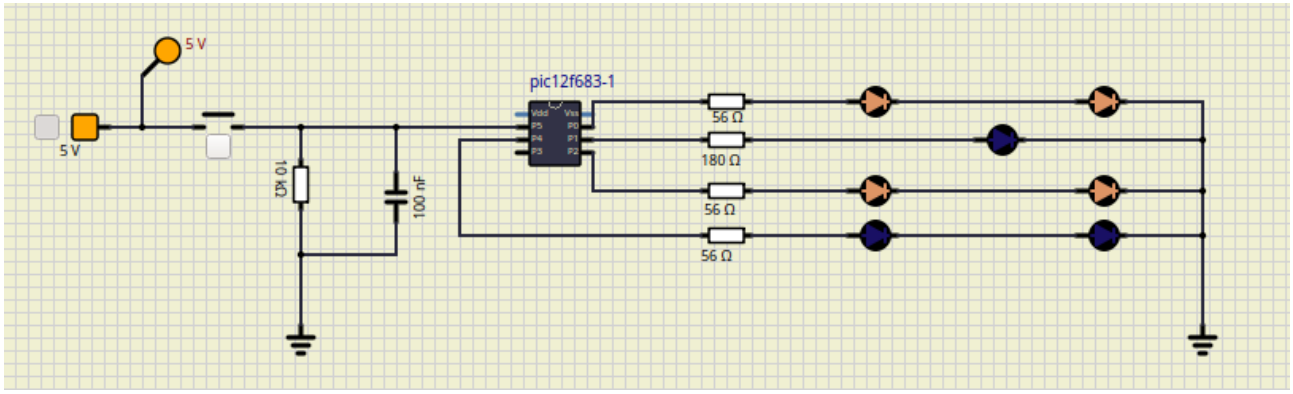

Figura 7: Cara #4 del dado

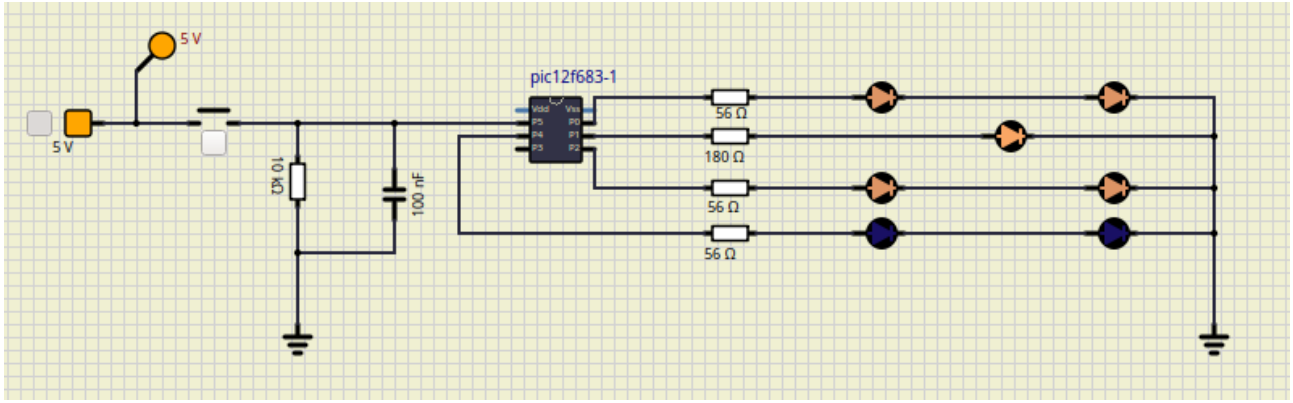

Figura 8: Cara #5 del dado

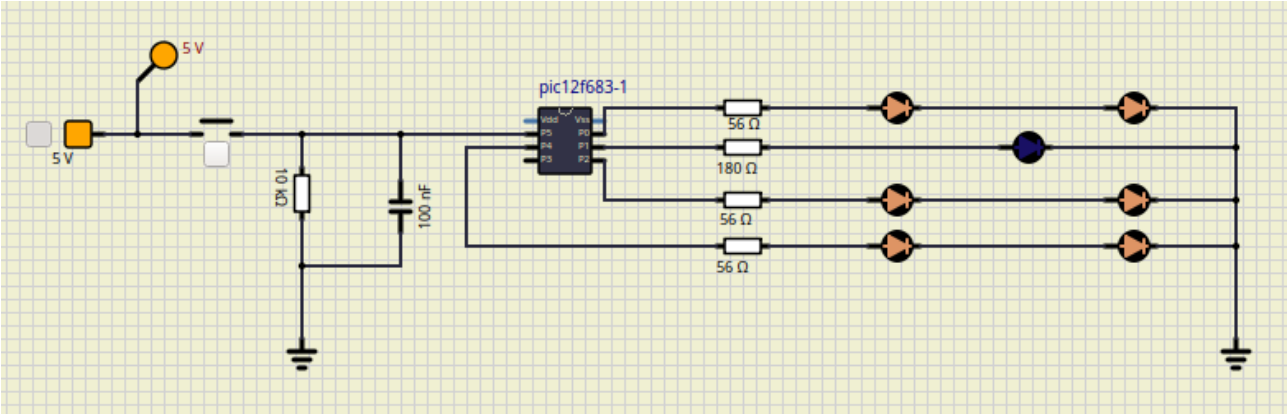

Figura 9: Cara #6 del dado

Cara	Pines activos	Descripción visual
1	GP1	LED central
2	GP0	Par superior
3	GP0, GP1	Par superior + central
4	GP0, GP2	Par superior e inferior
5	GP0, GP1, GP2	Superior, central e inferior
6	GP0, GP2, GP3	Tres pares de LEDs (superior, medio, inferior)

Tabla 2: Relación entre la cara del dado y los LEDs encendidos

3.3. Mediciones de tensión en los pines de salida

Para verificar el comportamiento eléctrico del circuito, se realizaron mediciones de tensión en los nodos de salida del microcontrolador durante la activación de los LEDs. Las pruebas se realizaron utilizando herramientas de medición disponibles en el simulador SimulIDE.

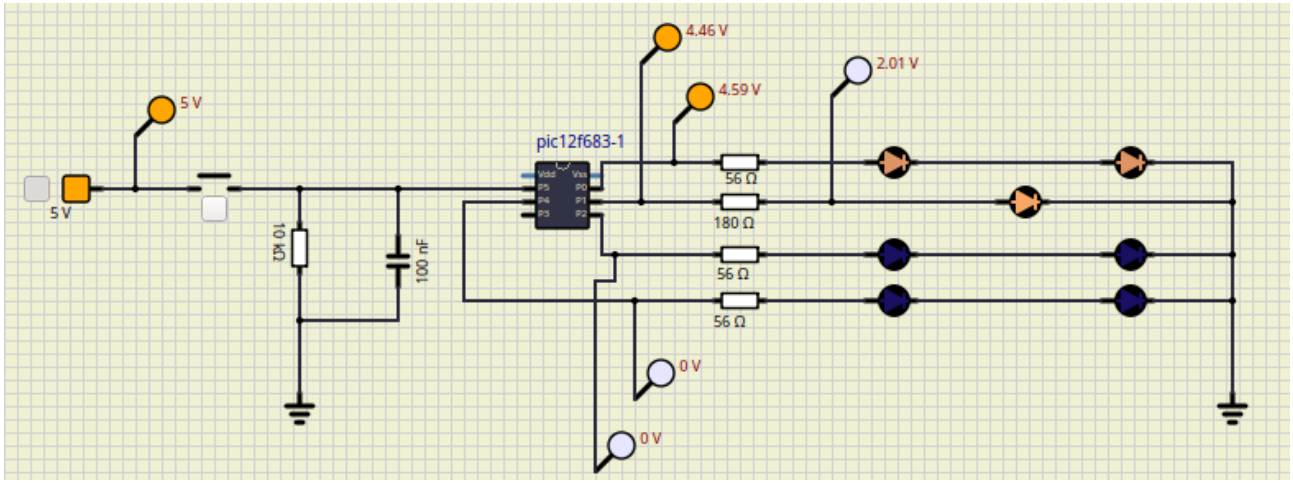

Figura 10: Prueba de medición de tensión 1

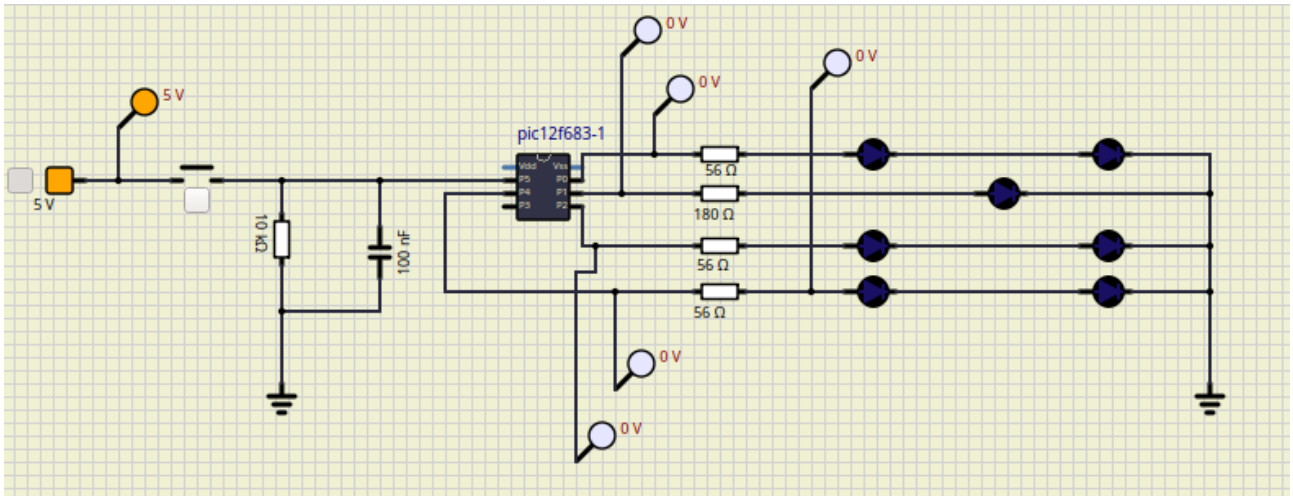

Figura 11: Prueba de medición de tensión 2

En la Figura 10 se observa que, durante el encendido de un par de LEDs, la tensión medida a la salida del pin del microcontrolador es de aproximadamente 4.59V, mientras que la caída de tensión a través de un LED encendido es de 2.01V. Esto confirma que la caída de tensión por LED es consistente con el valor esperado para un LED naranja, y que el microcontrolador entrega un nivel lógico alto adecuado para suministrar corriente en configuración current sourcing.

En la Figura 11 se muestra un estado en el cual las salidas se encuentran en nivel bajo (0V), con todos los LEDs apagados. Esto permite confirmar que el microcontrolador controla correctamente el encendido y apagado de las salidas digitales.

El diseño del circuito fue realizado para que la corriente que circula por cada LED no exceda los 20mA, utilizando resistencias limitadoras calculadas mediante la Ley de Ohm. En el caso de dos LEDs en serie con una resistencia de 56Ω:

$$I = \frac{V_{DD} - 2V_{LED}}{R} = \frac{5V - 4V}{56\Omega} \approx 17{,}86mA \tag{1}$$

Y para un solo LED con resistencia de 180 Ω:

$$I = \frac{5V - 2V}{180\Omega} \approx 16{,}67mA \tag{2}$$

Ambos valores están por debajo del límite de 25mA indicado por el fabricante, garantizando así la seguridad del componente y del microcontrolador. Las tensiones medidas en la simulación respaldan los valores teóricos, confirmando el correcto dimensionamiento de las resistencias y la validez del diseño.

4. Conclusiones y recomendaciones

4.0.1. Conclusiones

- Se logró implementar un sistema digital funcional que simula el lanzamiento de un dado utilizando el microcontrolador PIC12F683 y un conjunto de LEDs naranjas como salida visual.
- La configuración adecuada de los registros TRISIO, GPIO y __CONFIG, junto con una correcta manipulación de los pines de entrada y salida, permitió controlar eficazmente el comportamiento del sistema desde el software.
- El generador pseudoaleatorio implementado por software, basado en una variante de Xorshift, produjo una secuencia válida de números del 1 al 6, que se visualizaron mediante distintas combinaciones de LEDs.

4.0.2. Recomendaciones

- En una aplicación de hardware físico, se recomienda usar un osciloscopio o analizador lógico para verificar la calidad de las señales digitales y la respuesta del sistema ante variaciones en la alimentación.
- En caso de utilizar este sistema como base para otras aplicaciones, se debe considerar el consumo acumulado de corriente al encender múltiples LEDs simultáneamente, y verificar que no se excedan los límites totales permitidos por el microcontrolador.
- El valor del dado se muestra solo durante un retardo fijo. Se podría mejorar la interacción manteniendo los LEDs encendidos hasta que se presione nuevamente el botón, facilitando la lectura del resultado.

Referencias

[1] Microchip Technology Inc., "Pic12f683 data sheet: 8-pin flash-based 8-bit cmos microcontroller with nanowatt technology," https://ww1.microchip.com/downloads/en/devicedoc/41211d_.pdf, 2007.

[2] ——, "Pic12f683 - 8-bit pic microcontrollers," https://www.microchip.com/en-us/product/pic12f683, Último acceso: marzo 2025.

[3] Electrónica Costa Rica, "Catálogo de componentes electrónicos," https://electronica.cr/, 2025.

[4] Micro JPM, "Resistencias y componentes electrónicos," https://www.microjpm.com/, 2025.

[5] Centroniks, "Tienda en línea de componentes electrónicos," https://centroniks.com/, 2025.

[6] M. Villalta, "Gpios y el pic12f683 - ie-0624 laboratorio de microcontroladores," Presentación, Escuela de Ingeniería Eléctrica, Universidad de Costa Rica, 2025.