Laboratorio 1: Introducción a microcontroladores y manejo de GPIOS.

Jafet Soto Arrieta
B77543

***Abstract*—Este trabajo se ha desarrollado para introducir a los estudiantes al manejo de GPIOs, generación de números aleatorios y desarrollo de firmware para el microcontroladores PIC12F683**

I. Introduction

El objetivo es crear un simulador de dado simple que utilice LEDs y un botón para representar el lanzamiento de un dado de seis caras; la señal del botón indica el inicio de la simulación, el microcontrolador debe seleccionar un número aleatorio entre 1 y 6 para que sea mostrado como resultado en los leds.

Se utiliza SimulIDE para realizar el laboratorio, este simulador permite trabajar con el microcontrolador PIC12F683 que se requiere y diferentes componentes electrónicos que nos permiten llegar a la solución final.

Primero se deben estudiar las entradas del microcontrolador, para plantear un diseño de 3 etapas, en primer lugar, el botón que indica el inicio de la simulación, seguidamente la etapa encargada de ejecutar la simulación y finalmente la salida de los datos resultantes del proceso.

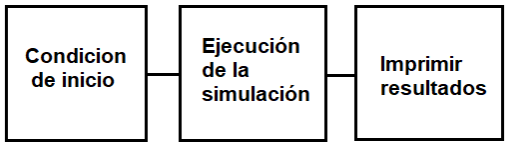

Fig. 1. Diagrama de ejecución del laboratorio 1. Elaboración propia.

II. Nota Teorica

Analizaremos el funcionamiento del PIC12F683, su diagrama de bloques, distribución de pines y funcionamiento general del microcontrolador, para definir un diseño para el simulador del dado.

A. El PIC12F683

Es un microcontrolador de 8 bits muy utilizado en aplicaciones de electrónica y sistemas embebidos debido a su bajo costo y amplia gama de características. El PIC12F683 tiene una arquitectura RISC, cuenta con una variedad de periféricos integrados como GPIOs, temporizadores, ADC, comunicación serial y generadores de PWM.

B. Caracteristicas principales

Posee una memoria Flash integrada cuyo tamaño varía según la versión, alcanzando hasta 1 KB para almacenar el programa de usuario. Además, cuenta con memoria RAM para datos temporales y EEPROM para datos no volátiles. Puede operar a frecuencias de reloj que van desde unos pocos kHz hasta varios MHz.

C. Diagrama de pines

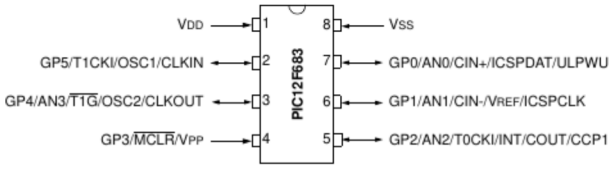

Fig. 2. Diagrama de pines del PIC12F683. [1]

- **VDD:** Alimentación positiva.
- **GP5/T1CKI/OSC1/CLKIN:** Entrada/Salida GPIO con resistencia de pull-up programable y activación de interrupción por cambio.
- **GP4/AN3/T1G/OSC2/CLKOUT:** Entrada/Salida GPIO con resistencia de pull-up programable y activación de interrupción por cambio.
- **GP3/MCLR/VPP:** Entrada GPIO con activación de interrupción por cambio y función de Master Clear con pull-up interno.
- **GP2/AN2/T0CKI/INT/COUT/CCP1:** Entrada/Salida GPIO con resistencia de pull-up programable y activación de interrupción por cambio.
- **GP1/AN1/CIN-/VREF/ICSPCLK:** Entrada/Salida GPIO con resistencia de pull-up programable y activación de interrupción por cambio.
- **GP0/AN0/CIN+/ICSPDAT/ULPWU:** Entrada/Salida GPIO con resistencia de pull-up programable y activación de interrupción por cambio.
- **VSS:** Tierra (referencia de tierra).

D. Diagrama de bloques

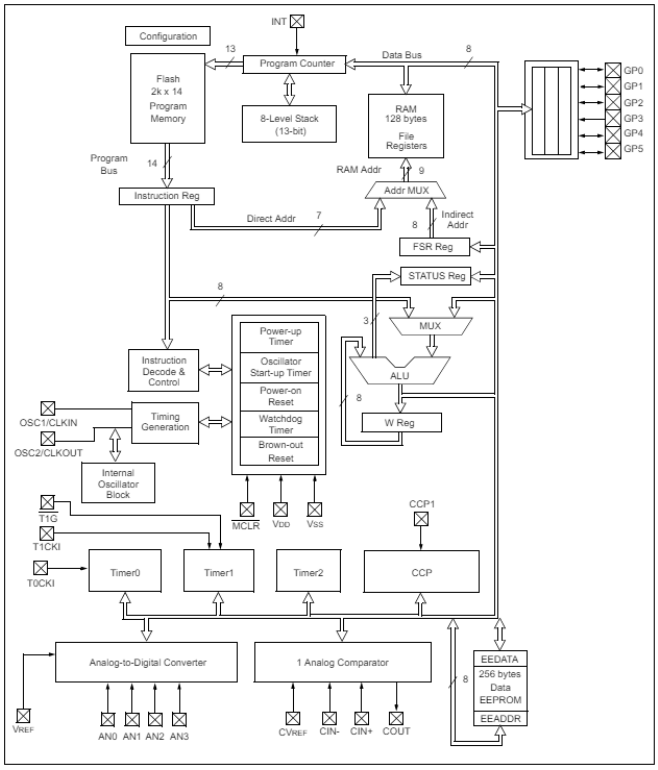

Fig. 3. Diagrama de bloques del PIC12F683. [1]

- **CPU:** Es el núcleo del microcontrolador y ejecuta las instrucciones del programa almacenado en la memoria Flash.
- **Memoria de Programa (Flash):** Almacena el código de programa que será ejecutado por la CPU. El tamaño de esta memoria varía según la versión específica del microcontrolador.
- **Memoria de Datos (RAM y EEPROM):** La RAM se utiliza para almacenar datos temporales durante la ejecución del programa, mientras que la EEPROM proporciona una memoria no volátil para almacenar datos que necesitan ser conservados incluso cuando la alimentación se corta.
- **Puertos de Entrada/Salida (GPIOs):** Estos son los pines que permiten al microcontrolador interactuar con dispositivos externos. Pueden ser configurados como entradas o salidas digitales según sea necesario.
- **Periféricos Integrados:** El PIC12F683 cuenta con una variedad de periféricos integrados como temporizadores, comparadores analógicos, módulos de comunicación serial y generadores de PWM. Estos periféricos proporcionan funcionalidades adicionales que pueden ser utilizadas en diferentes aplicaciones.
- **Oscilador Interno:** Proporciona la señal de reloj necesaria para el funcionamiento del microcontrolador. Puede ser configurado internamente o puede ser alimentado por un oscilador externo.

III. Desarrollo

Para nuestro simulador de dado de 6 caras, la condición que indica el inicio de la simulación, está dada por la activación de un botón.

A. Condicion de inicio

Este botón actuará como una entrada digital que será monitoreada por el microcontrolador. Cuando se presiona el botón, la señal correspondiente se enviará al pin GPIO, y el microcontrolador puede detectar este evento mediante una interrupción o mediante un bucle de lectura de entrada; se conectará al PIN5 del PIC12F683.

El debouncing es un proceso crucial en el diseño de circuitos electrónicos que involucran botones o interruptores. Este proceso se utiliza para filtrar las fluctuaciones no deseadas en la señal de entrada que pueden ocurrir cuando un botón se presiona o se suelta. Estas fluctuaciones, conocidas como rebotes, pueden causar múltiples cambios de estado en un corto período de tiempo, lo que puede llevar a un comportamiento errático del sistema.

Las resistencias de pull-up son resistencias que se conectan entre la entrada digital de un microcontrolador y la fuente de alimentación (VDD), generalmente en un sistema de lógica digital donde la entrada del dispositivo puede estar desconectada (flotante) en ausencia de una señal de entrada.

El valor típico de 120 ohmios para las resistencias de pull-up es una elección común que proporciona una corriente suficiente para establecer un nivel alto en la entrada del microcontrolador, pero al mismo tiempo limita la corriente para evitar consumos excesivos de energía y protege al dispositivo.

Queremos generar un retardo de 36ns, este valor de tiempo de retardo es suficiente para filtrar los rebotes del botón y garantizar una respuesta rápida del sistema.

$$\tau = RC \tag{1}$$

Con los valores definidos obtenemos una capacitancia de valor $150pF$, y el circuito debouncing es el siguiente:

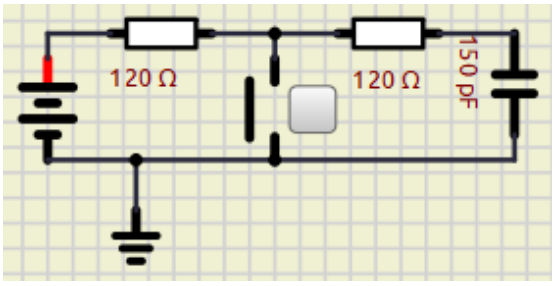

Fig. 4. Circuito de detección de inicio.

B. Imprimir resultados

Para facilitar la impresión definimos grupos de 1, 2 y 3 leds, para indicar el resultado de la simulación esta combinación de leds nos permite reducir las señales necesarias para mostrar la salida. Cuando el resultado sea mayor a 3 se utilizarán

combinaciones de estos grupos de leds para imprimir el resultado.

Para determinar las resistencias de protección que se deben de agregar a cada led utilizamos la ley de Ohm, conociendo que para una entrada VDD de 5V la corriente máxima de los pines del microcontrolador será de 25mA, por lo tanto:

$$R = \frac{5V}{25mA} = 200\Omega \tag{2}$$

Conectaremos el grupo de un led al pin0, el grupo de 2 al pin1 y el grupo de 3 al pin2, el esquema resultante es el siguiente:

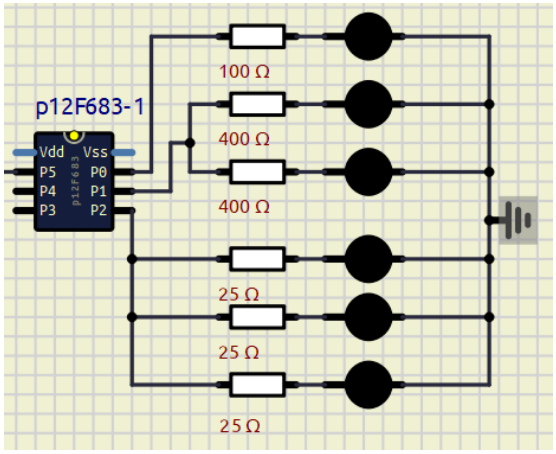

Fig. 5. Circuito de salida del simulador.

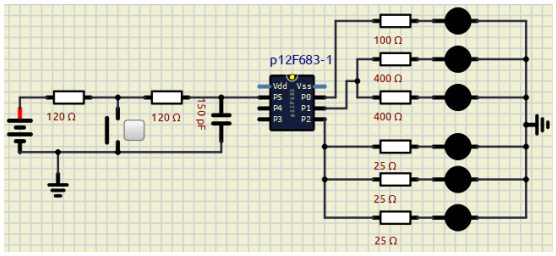

Fig. 6. Circuito simulador de un dado de 6 caras.

C. Lista de componentes

A continuación se muestra la lista de componentes necesarios para implementar el simulador:

Componente	Valor	Cantidad
Resistencia de 120 ohms	$5.99	1
Resistencia de 200 ohms	$5.49	1
Resistencias de 400 ohms	$5.49	2
Resistencias de 25 ohms	$5.79	3
PIC12F683	$2.91	1
Paquete de leds	$7.56	1
Total	**$33.23**	

TABLE I
COMPONENTES ELECTRONICOS NECESARIOS.

D. Ejecución de la simulación

Una vez definidas los componentes electrónicos del simulador podemos definir el firmware que ejecutará la tarea de seleccionar la cara resultante.

El firmware controla la salida de varios pines GPIO de acuerdo con un valor pseudoaleatorio generado por un algoritmo de desplazamiento de registro de retroalimentación lineal (LFSR). Resumen del funcionamiento del código:

- Se incluyen los encabezados necesarios para el microcontrolador PIC12F675 y se declara una función de retardo (`delay`) y una función para generar un número pseudoaleatorio (`RandomNum`).
- En la función `main`, se configura el pin `GP5` como entrada y se inicializan todas las GPIO en bajo. Luego, se entra en un bucle infinito (`while(1)`).
- Dentro del bucle infinito, se verifica si el pin `GP5` está en bajo. Si es así, se genera un número pseudoaleatorio utilizando la función `RandomNum`.
- Dependiendo del valor generado, se establecen diferentes combinaciones de pines GPIO en alto, de acuerdo con un patrón predefinido.
- Después de establecer los pines GPIO, se llama a la función `delay` para mantener esa configuración durante un tiempo determinado (300 ms en este caso).
- Finalmente, se restablecen todas las GPIO en bajo y se repite el proceso.
- La función `delay` se utiliza para introducir un retraso en milisegundos aproximadamente igual al valor pasado como argumento. Dentro de esta función, hay dos bucles anidados que generan el retraso.
- La función `RandomNum` utiliza un algoritmo de desplazamiento de registro de retroalimentación lineal (LFSR) para generar un valor pseudoaleatorio entre 1 y 6, asegurando que el valor esté dentro de ese rango antes de ser devuelto.

E. Resultados

Una vez implementado el código que podemos encontrar en el repositorio GitHub podemos simular el dado, los diferentes estados se muestran a continuación.

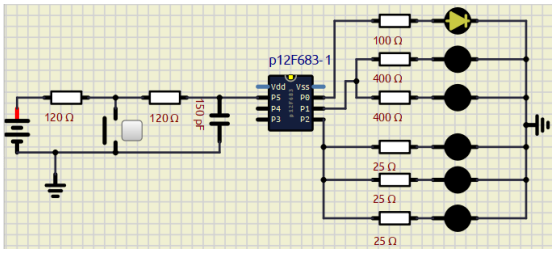

Fig. 7. Salida 1 para el dado simulado.

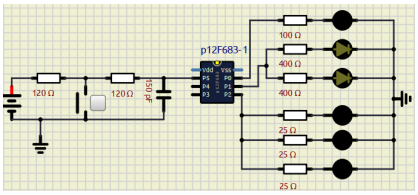

Fig. 8. Salida 2 para el dado simulado.

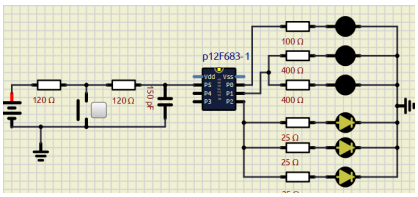

Fig. 9. Salida 3 para el dado simulado.

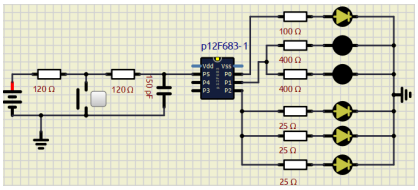

Fig. 10. Salida 4 para el dado simulado.

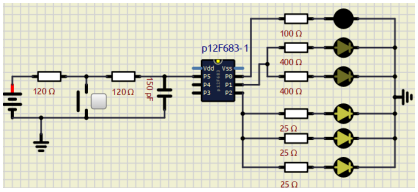

Fig. 11. Salida 5 para el dado simulado.

IV. Conclusiones

En este proyecto, se ha desarrollado un simulador de dado de 6 caras utilizando un microcontrolador PIC12F675. El simulador implementado permite generar resultados aleatorios que simulan el lanzamiento de un dado convencional.

Durante el desarrollo del proyecto, se logró crear un simulador funcional que cumple con los requisitos establecidos. Sin embargo, se identificó que la función de generación de números aleatorios puede ser mejorada. Actualmente, se utiliza un algoritmo de desplazamiento de registro de retroalimentación lineal (LFSR) para generar los valores pseudoaleatorios. Se reconoce que este método podría no proporcionar una distribución de números completamente uniforme y aleatoria.

Para futuras iteraciones del proyecto, se recomienda explorar y evaluar diferentes técnicas de generación de números aleatorios que puedan mejorar la aleatoriedad y uniformidad de los resultados obtenidos. Esto garantizará que el simulador de dado proporcione una experiencia de usuario más auténtica y precisa.

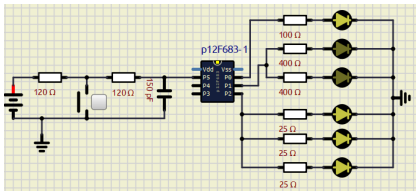

Fig. 12. Salida 6 para el dado simulado.

References

[1] Microchip Technology Inc., "PIC12F683 Data Sheet," *Microchip Technology Inc.*, 2024. https://www.microchip.com/wwwproducts/en/PIC12F683