Universidad de Costa Rica

Facultad de Ingeniería

Escuela de Ingeniería Eléctrica

IE-0624 - Laboratorio de Microcontroladores

II ciclo 2023

Laboratorio 1

Introducción a microcontroladores y manejo de GPIOS

Kristel Ariana Herrera Rodríguez C13769

Grupo 01

Profesor: Marco Villalta Fallas

Índice

Índice de figuras

1. Resumen

Los microcontroladores son comunmente utilizados en el ámbito de la ingeniería. Estos ofrecen una amplia gama de aplicaciones, desde dispositivos médicos hasta sistemas embebidos en automóviles. Uno de los aspectos clave para entender estos sistemas es el manejo de E/S de Propósito General (GPIOs, por sus siglas en inglés: General Purpose Input Outputs) y la simulación de eventos aleatorios, que son fundamentales en la generación de señales y en el control de dispositivos. En este contexto, el presente laboratorio pretende introducir estos conceptos a través del desarrollo de un simulador de dado utilizando LEDs, un botón y el microcontrolador PIC12F683.

La simulación a realizar busca replicar el comportamiento de un dado de seis caras, donde el número generado aleatoriamente se representa mediante la cantidad de LEDs encendidos en el circuito. Esto no solo sirve como una forma intuitiva de entender el uso de GPIOs, sino también introduce la generación de números aleatorios en un contexto práctico. Para interactuar con el sistema, el usuario debe presionar un botón que simula el acto de lanzar el dado, generando un resultado visual en los LEDs.

Los resultados indican que el sistema es capaz de generar números aleatorios de manera efectiva, representando estos números a través de patrones de LEDs. Las mediciones de voltaje y corriente confirmaron que el circuito funciona dentro de los parámetros seguros para los componentes electrónicos utilizados. Sin embargo, se observaron pequeñas discrepancias entre los valores teóricos y los medidos, aunque estas se mantuvieron dentro de los márgenes aceptables. En resumen, el laboratorio demostró ser una herramienta efectiva para aprender sobre la programación de microcontroladores, el manejo de GPIOs y la generación de números aleatorios. Se recomienda la investigación de técnicas de filtrado de señal para futuras implementaciones.

2. Nota Teorica

2.1. Microcontrolador PIC12F683

El microcontrolador PIC12F683, desarrollado por Microchip Technology Inc, es un dispositivo de 8 pines que ofrece diversas funcionalidades que se explorarán en detalle en las siguientes secciones. Entre ellas, destaca la versatilidad de sus puertos de Propósito General de Entrada/Salida (GPIO), que permiten una amplia gama de configuraciones de entrada y salida.

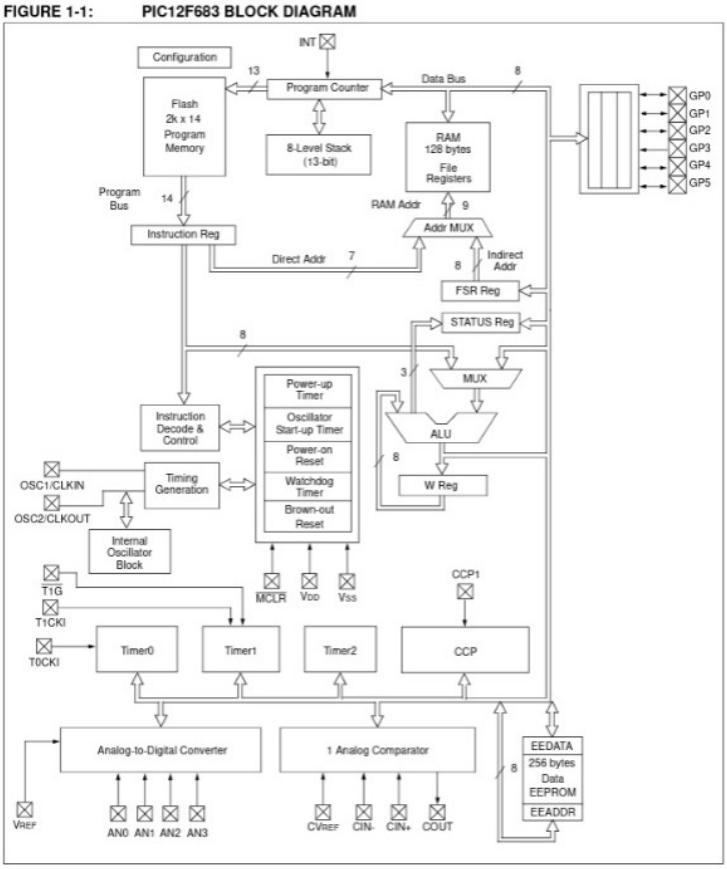

Figura 1: Diagrama de bloques del microncontrolador [1].

2.2. Organización de la Memoria

2.2.1. Organización de la Memoria de Programa

El PIC12F683 tiene un contador de programa de 13 bits capaz de abordar un espacio de memoria de programa de 8k x 14. Solo se ha implementado físicamente el primer 2k x 14 (0000h-07FFh) para el PIC12F683. Acceder a una ubicación por encima de estos límites provocará un retorno al primer espacio de 2K x 14. El vector de reinicio está en 0000h y el

vector de interrupción está en 0004h [1].

2.3. Puerto GPIO

2.3.1. Descripción General

El microcontrolador cuenta con 6 pines de propósito general disponibles, los cuales, a excepcion del GP3, se pueden configurar como entradas o salidas. El GP3 está configurado incialmente como un "Master Clear with internal pull-up", y además puede ser una entrada o un voltaje programado.

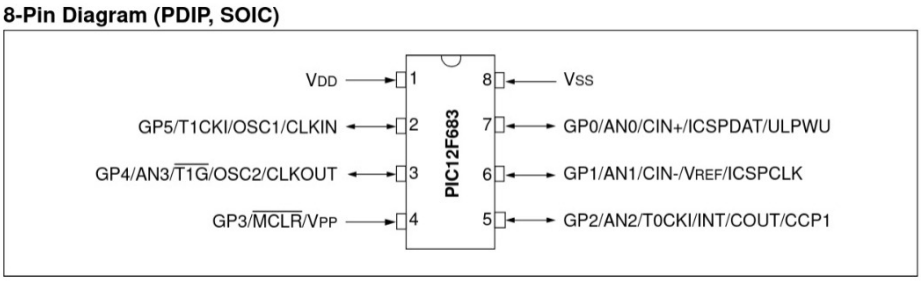

Figura 2: Diagrama de pines[1].

2.3.2. GPIO y los Registros TRISIO

GPIO es un puerto bidireccional de 6 bits de ancho. Configurar un bit en TRISIO con un valor de 1 hará que el pin GPIO correspondiente funcione como una entrada. Asignar un valor de 0 a un bit en TRISIO hará que el pin GPIO correspondiente funcione como una salida. Además, se utilizó el registro CONFIG para establecer el WDT en 0, ya que este está habilitado por defecto, lo que causaría que el software se reinicie automáticamente cada cierto tiempo.

2.4. Funciones de Pin Adicionales

2.4.1. Registro ANSEL

Configurar el bit ANSEL apropiado en alto hará que todas las lecturas digitales en el pin se lean como '0'.

TABLE 1-1: PIC12F683 PINOUT DESCRIPTION

Name	Function	Input Type	Output Type	Description
VDD	VDD	Power	—	Positive supply
GP5/T1CKI/OSC1/CLKIN	GP5	TTL	CMOS	GPIO I/O with prog. pull-up and interrupt-on-change
	T1CKI	ST	—	Timer1 clock
	OSC1	XTAL	—	Crystal/Resonator
	CLKIN	ST	—	External clock input/RC oscillator connection
GP4/AN3/$\overline{T1G}$/OSC2/CLKOUT	GP4	TTL	CMOS	GPIO I/O with prog. pull-up and interrupt-on-change
	AN3	AN	—	A/D Channel 3 input
	$\overline{T1G}$	ST	—	Timer1 gate
	OSC2	—	XTAL	Crystal/Resonator
	CLKOUT	—	CMOS	FOSC/4 output
GP3/$\overline{MCLR}$/VPP	GP3	TTL	—	GPIO input with interrupt-on-change
	$\overline{MCLR}$	ST	—	Master Clear with internal pull-up
	VPP	HV	—	Programming voltage
GP2/AN2/T0CKI/INT/COUT/CCP1	GP2	ST	CMOS	GPIO I/O with prog. pull-up and interrupt-on-change
	AN2	AN	—	A/D Channel 2 input
	T0CKI	ST	—	Timer0 clock input
	INT	ST	—	External Interrupt
	COUT	—	CMOS	Comparator 1 output
	CCP1	ST	CMOS	Capture input/Compare output/PWM output
GP1/AN1/CIN-/VREF/ICSPCLK	GP1	TTL	CMOS	GPIO I/O with prog. pull-up and interrupt-on-change
	AN1	AN	—	A/D Channel 1 input
	CIN-	AN	—	Comparator 1 input
	VREF	AN	—	External Voltage Reference for A/D
	ICSPCLK	ST	—	Serial Programming Clock
GP0/AN0/CIN+/ICSPDAT/ULPWU	GP0	TTL	CMOS	GPIO I/O with prog. pull-up and interrupt-on-change
	AN0	AN	—	A/D Channel 0 input
	CIN+	AN	—	Comparator 1 input
	ICSPDAT	ST	CMOS	Serial Programming Data I/O
	ULPWU	AN	—	Ultra Low-Power Wake-up input
VSS	VSS	Power	—	Ground reference

Legend: AN = Analog input or output; TTL = TTL compatible input; HV = High Voltage; CMOS = CMOS compatible input or output; ST = Schmitt Trigger input with CMOS levels; XTAL = Crystal

Figura 3: Descripcion de pines[1].

2.5. Especificaciones electricas

15.0 ELECTRICAL SPECIFICATIONS

Absolute Maximum Ratings(†)

Ambient temperature under bias ..... -40° to +125°C
Storage temperature ..... -65°C to +150°C
Voltage on VDD with respect to VSS ..... -0.3V to +6.5V
Voltage on $\overline{MCLR}$ with respect to VSS ..... -0.3V to +13.5V
Voltage on all other pins with respect to VSS ..... -0.3V to (VDD + 0.3V)
Total power dissipation(1) ..... 800 mW
Maximum current out of VSS pin ..... 95 mA
Maximum current into VDD pin ..... 95 mA
Input clamp current, IIK (VI < 0 or VI > VDD) ..... ± 20 mA
Output clamp current, IOK (Vo < 0 or Vo >VDD) ..... ± 20 mA
Maximum output current sunk by any I/O pin ..... 25 mA
Maximum output current sourced by any I/O pin ..... 25 mA
Maximum current sunk by GPIO ..... 90 mA
Maximum current sourced GPIO ..... 90 mA

Note 1: Power dissipation is calculated as follows: $P_{DIS} = V_{DD} \times \{I_{DD} - \sum I_{OH}\} + \sum \{(V_{DD} - V_{OH}) \times I_{OH}\} + \sum (V_{OL} \times I_{OL})$.

† NOTICE: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at those or any other conditions above those indicated in the operation listings of this specification is not implied. Exposure above maximum rating conditions for extended periods may affect device reliability.

Figura 4: Especificaciones electricas[1].

2.6. Conexión de Botones a Arduino

2.6.1. Consideraciones Generales

Al utilizar un boton en un microcontrolador se deben tomar en cuenta varios aspectos para asegurar su correcto funcionamiento. Entre estos está la configuración en la que se conecta, puede ser con resistencias pull-up o pull-down y consideraciones mecánicas como el rebote del botón. Para este laboratorio se eligió un pulsador normalmente abierto.

2.6.2. Importancia de las Resistencias Pull-Up y Pull-Down

Las resistencias pull-up o pull-down son fundamentales para mantener un estado lógico predefinido en el pin cuando el botón no está siendo presionado. Esto evita que el pin entre en un estado indeterminado que podría interpretarse de manera errónea por el microcontrolador. Las resistencias pull-up mantienen un estado lógico alto en el pin, activándose al presionar el botón y llevando el estado a bajo. Por otro lado, las resistencias pull-down mantienen un estado lógico bajo, cambiando a alto cuando se acciona el botón. Comúnmente, se utilizan resistencias con valores que oscilan entre 1 y 10 KOhms [2]. En este caso se eligió una resistencia de 5,1kΩ. A pesar de que el microcontrolador ofrece la opción de habilitar resistencias pull-up internas, se utilizará una resistencia pull-up externa en el circuito. Esta decisión se tomó para tener un mayor control sobre el valor de la resistencia y para proteger el sistema contra posibles ruidos eléctricos.

2.6.3. Manejo de rebotes

Los rebotes son fluctuaciones en la señal eléctrica que ocurren cuando un botón se presiona o se libera. Estas fluctuaciones pueden ser interpretadas erróneamente por el microcontrolador como múltiples pulsaciones en lugar de una única acción del botón.

2.6.4. Métodos para Manejar Rebotes

Por Software

El método más común para manejar los rebotes en software es a través de un retardo o *debounce delay*. Después de detectar un cambio en el estado del botón, el programa espera un período corto de tiempo y luego verifica el estado del botón nuevamente[3].

Por Hardware

Se puede utilizar un capacitor en paralelo con el botón para suavizar las fluctuaciones de la señal, aunque esto no siempre es práctico en todos los diseños de circuitos[3].

2.7. Cálculo de la Resistencia para el LED Púrpura

Para calcular la resistencia de protección adecuada para el LED púrpura, se utiliza la Ley de Ohm:

$$R = \frac{(V_{DD} - V_{LED})}{I}$$

Donde:

- R es la resistencia que se busca.
- V_{DD} es el voltaje de alimentación del microcontrolador, que en este caso es de $5V$.
- V_{LED} es el voltaje nominal del LED, que es $3{,}5V$.
- I es la corriente máxima que el LED puede manejar, $0{,}03A$.

Se sustituyen los valores en la fórmula:

$$R = \frac{(5V - 3{,}5V)}{0{,}03A}$$
$$R = \frac{1{,}5V}{0{,}03A}$$
$$R = 50\Omega$$

Una resistencia de 50Ω sería la necesaria según los cálculos, pero para darle un margen de seguridad, es recomendable redondear al siguiente valor de resistencia comercial disponible que sea mayor. En este caso, se opta por una resistencia de 56Ω.

2.8. Diseño del circuito

Se procedió a construir el siguiente circuito en el simulador

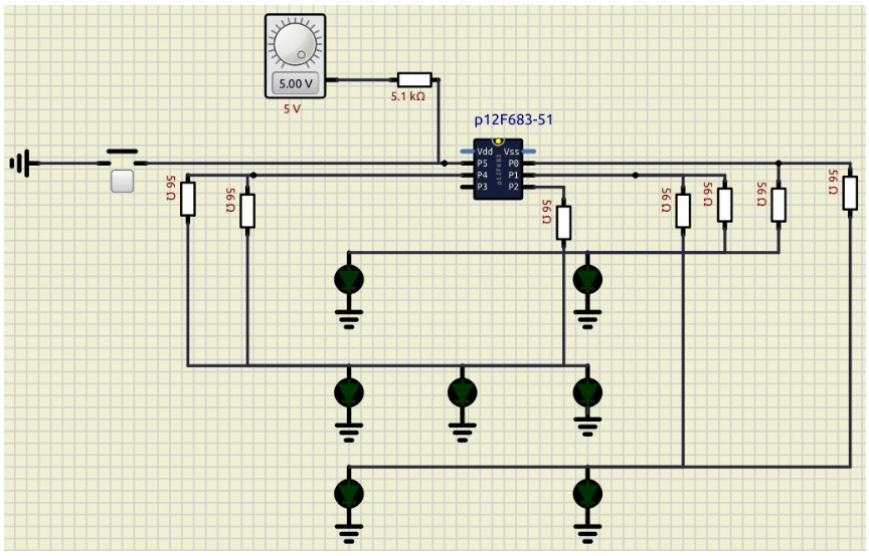

Figura 5: Circuito en el simulador.

Se utilizaron siete resistencias de 56Ω para la protección de los LEDs. Además, se empleó la resistencia de pull-up de 5,1kΩ con el fin de garantizar una mejor lectura del estado del botón, donde para esto también se colocó una fuente de 5V. Se incorporaron siete LEDs púrpura, cada uno con una tensión nominal de 3,5V y una corriente máxima de 0.03A y un pulsador para simular el momento en que se lanzan los dados. Se requerían 7 LEDs para poder simular la forma del dado, ya que había un maximo de 5 salidas, se optó por conectar dos LEDs en paralelo en tres de los pines, mientras que un cuarto pin fue dedicado a un solo LED.

2.9. Lista de Componentes y Precios

A continuación se presenta la lista de componentes electrónicos utilizados, junto con su respectivo precio por unidad.

Componente	**Cantidad**	**Precio**
Microcontrolador PIC12F683	1	₡1 701
Resistor de 56Ω	7	₡25,01
Resistor de $5,1k\Omega$ (pull-up)	1	₡25,01
LED Púrpura	7	₡99,01
Pulsador	1	₡99,01

Tabla 1: Lista de componentes y precios.

3. Análisis de Resultados

3.1. Funcionalidad del programa

A continuación se presenta el diagrama de flujo del codigo

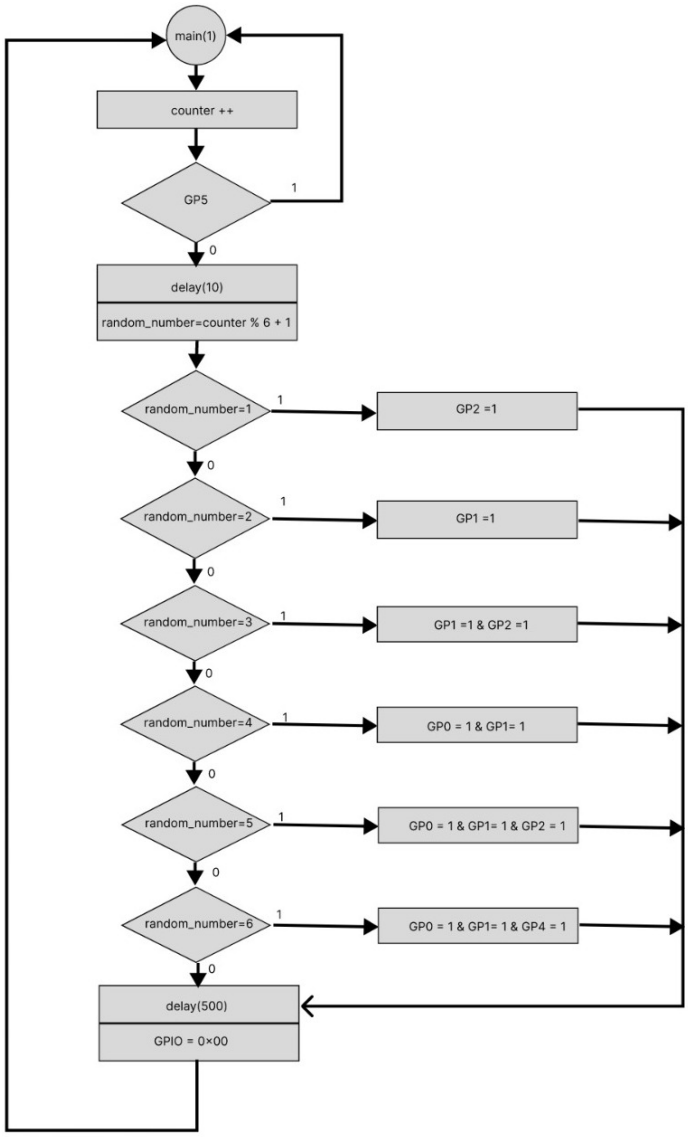

Figura 6: Diagram de flujo.

El diagrama de flujo se describe acontinuación:

1. **Inicio:** Al iniciar la simulación se comienza con la función main(1).
2. **Incrementar Contador:** Dentro del bucle (mientras el simulador está encendido), un contador se incrementa continuamente.
3. **Comprobación de Botón:** Se comprueba si el botón conectado a GP5 está presionado.

- Si no está presionado, se vuelve al inicio.
- Si está presionado, se avanza al siguiente paso.

4. **Manejo del Rebote del Botón:** Se espera hasta que el botón se libere y luego se introduce otro pequeño retraso.
5. **Generación de Número Aleatorio:** Utilizando el valor del contador, se genera un número aleatorio entre 1 y 6.
6. **Decisión de LEDs a Encender:** Aquí se hace una serie de comprobaciones para decidir qué LEDs se deben encender, dependiendo del número aleatorio generado.
7. **Retraso Grande:** Se introduce un retraso más largo para que el usuario pueda ver el resultado en los LEDs.
8. **Limpiar LEDs:** Se apagan todos los LEDs.
9. **Volver al Bucle:** Se vuelve al inicio para reiniciar todo el proceso.

3.2. Funcionalidad electrónica

En las siguientes imagenes se muestra que los LEDs se encienden en configuraciones diferentes similares a las de un dado al presionar el botón

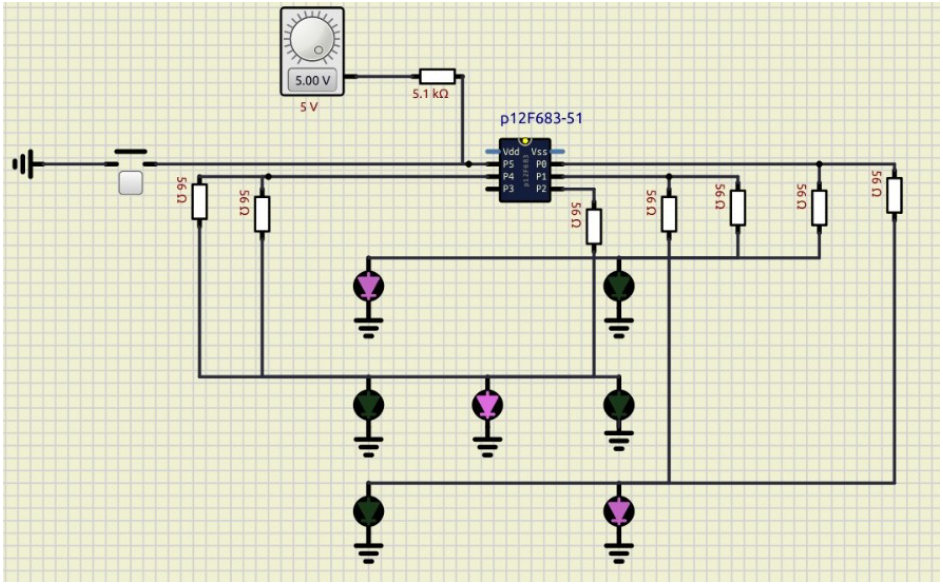

Figura 7: Dado 3.

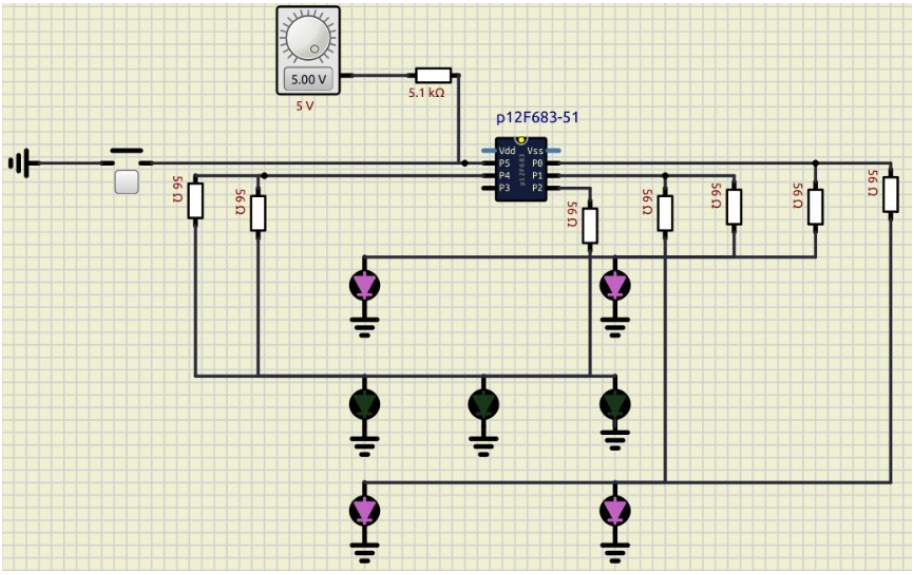

Figura 8: Dado 4.

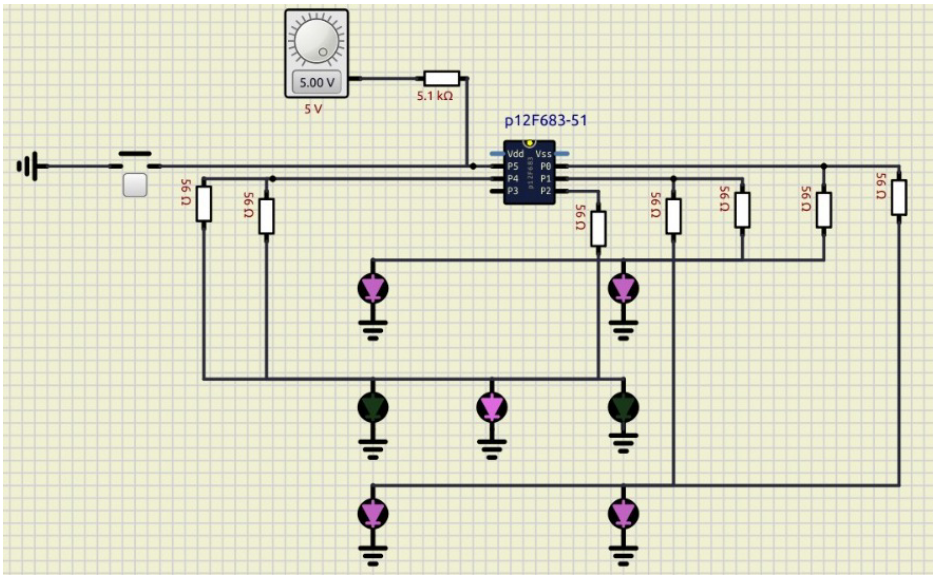

Figura 9: Dado 5.

Se procede a hacer un analisis de los voltajes y corrientes del circuito por medio de midiciones con amperimetros y multimetros

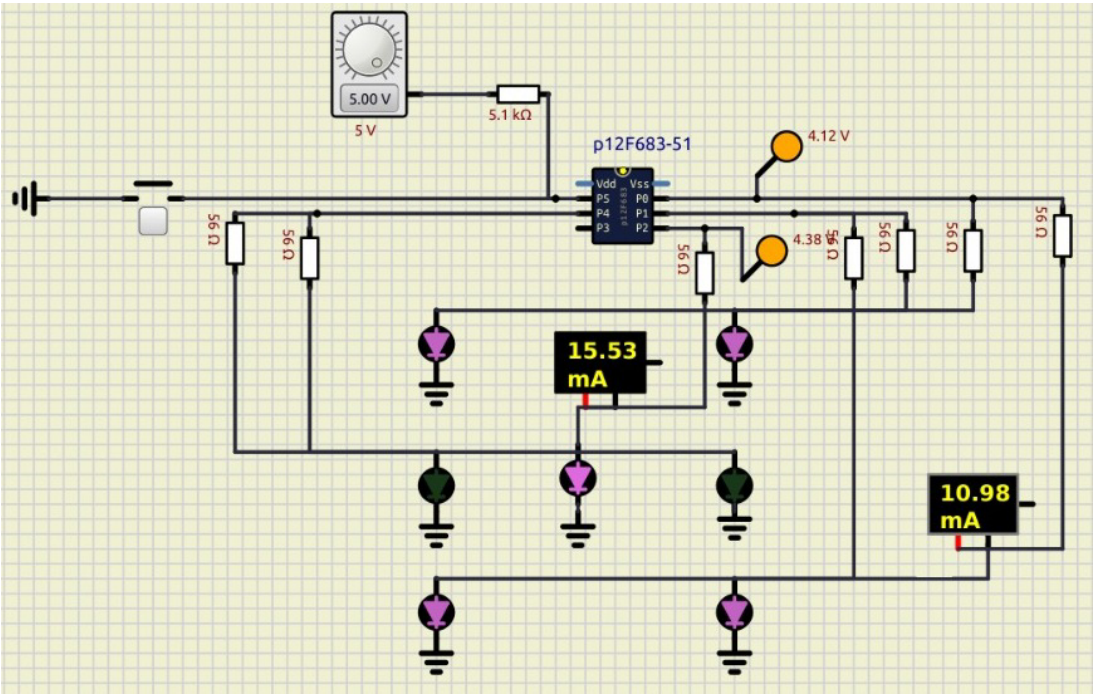

Figura 10: Mediciones.

Los cambios en corrientes y voltajes están relacionados a la carga que tiene conectada cada pin, no obstante, estas variaciones son mínimas. Ya que el voltaje en cada pin es diferente al esperado en la nota teorica, las corrientes que pasan por los LEDs son diferentes a las esperadas. Sin embargo, es importante señalar que dichas corrientes se mantienen dentro de los márgenes aceptables para el funcionamiento seguro de los LEDs.

4. Conclusiones y Recomendaciones

4.1. Conclusiones

1. El código desarrollado para simular el lanzamiento de un dado se ejecutó con éxito, tal como se verifica en el diagrama de flujo. El contador y el botón de entrada funcionan como se esperaba, generando números aleatorios y mostrando los resultados a través de LEDs.

2. Las mediciones con amperímetros y voltímetros validaron la funcionalidad electrónica del circuito. Los LEDs se encienden en configuraciones que simulan las caras de un dado, demostrando que tanto el hardware como el software trabajan en sincronía.

3. La funcionalidad del sistema es la correcta a pesar de las pequeñas desviaciones observadas en los voltajes y corrientes en comparación con los valores teóricos. Estas discrepancias fueron mínimas y se mantuvieron dentro de los límites seguros para los LEDs.

4.2. Recomendaciones

1. Para mayor precisión, se recomienda calibrar el circuito para alinear los valores teóricos y medidos de corriente y voltaje, teniendo en cuenta las especificaciones del LED y otros componentes.

2. Dado que se observaron pequeñas variaciones en los voltajes y corrientes, sería prudente investigar técnicas de filtrado de señal para reducir el ruido electrónico.

Referencias

[1] Microchip Technology Inc., *PIC12F683 Data Sheet: 8-Pin Flash-Based, 8-Bit CMOS Microcontrollers with nanoWatt Technology*, 2007, dS41211D. [Online]. Available: http://www.example.com/your-link-to-the-datasheet-if-available

[2] G. Factory. (2018, Jun) Botón o pulsador con arduino. Accedido el 25 de agosto del 2023. [Online]. Available: https://www.geekfactory.mx/tutoriales-arduino/boton-o-pulsador-con-arduino/

[3] J. Christoffersen. (2015, 9) Switch bounce: How to deal with it. [Online]. Available: https://www.allaboutcircuits.com/technical-articles/switch-bounce-how-to-deal-with-it/