Laboratorio 1: Introducción a microcontroladores y manejo de GPIOS

Ricardo HIdalgo Campos B63464

***Abstract*—Este laboratorio se realizo para desarrollar habilidades y ambientación en el área de la programación de microcontroladores y el manejo de GPIOS, específicamente en la utilización de microcontroladores PIC.**

I. Introducción

Este documento explicara paso a paso la realización del laboratorio 1 de microcontroladores. Se utilizo un microcontrolador PIC12f683 para simular el comportamiento de un dado de seis caras mediante el control de LED‘s al presionar un botón, esto sera implementado mediante la plataforma de SimulIDE en la cual se hará la configuración necesaria para la realización de la tarea. Para lograr esto, a nivel firmware se hizo el manejo de las salidas y entradas, tiempo de respuesta para visualizar el comportamiento y la generación de números aleatorios mediante el método de desplazamiento de registro de retroalimentación lineal.

El laboratorio se logro con éxito, se configuro de manera correcta la entrada del botón para que al activarse genere un numero aleatorio del 1 al 6 y se enciendan las salidas correspondientes a dicho numero. La lógica aplicada a las salidas obtuvo resultados esperados, por lo cual significa que el firmware fue generado de manera satisfactoria.

II. Nota Teórica

A continuación se describirá las características mas relevantes del microcontrolador PIC12f683, su funcionamiento y capacidades para realizar el laboratorio.

A. PIC12f683

El microcontrolador PIC12f683 pertenece a la familia de microcontroladores PIC12. Este es un microprocesador de 8 bits diseñado para aplicaciones de bajo costo y bajo consumo de energía.

B. Diagrama de pines

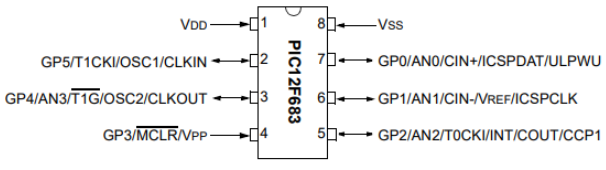

Fig. 1. Diagrama de pines del microcontrolador PIC12f683

- VDD: Entrada de alimentación.
- GP5/T1CKI/OSC1/CLKIN: Entrada/salida GPIO programable.
- GP4/T1G/OSC2/CLKOUT: Entrada/salida GPIO programable.
- GP3/MCLR/Vpp: Entrada GPIO con Pull-up interno.
- GP2/AN2/T0CKI/INT/COUT/CCP1: Entrada/salida GPIO programable.
- GP1/AN1/CIN-/Vref/ICSPCLK: Entrada/salida GPIO programable.
- GP0/AN0/CIN+/ICSPDAT/ULPWU: Entrada/salida GPIO programable.
- Vss: Conexión a tierra.

C. Diagrama de bloques

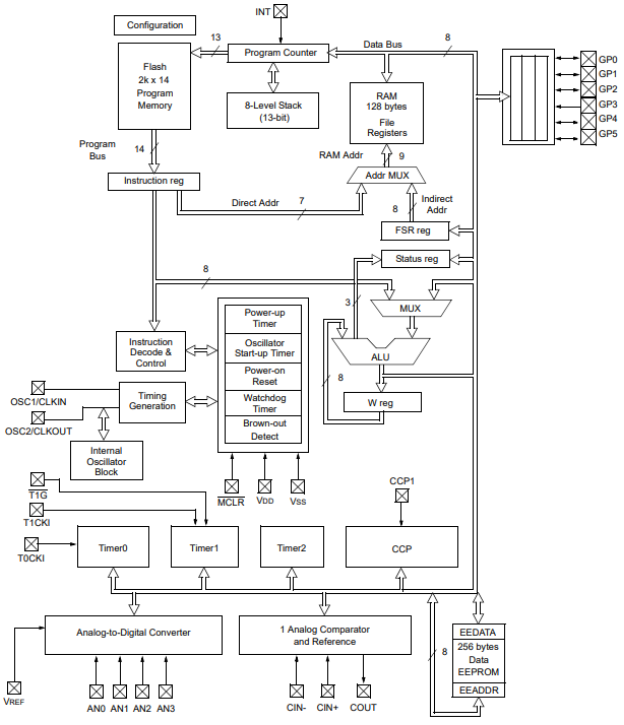

Fig. 2. Diagrama de bloques del microcontrolador PIC12f683

D. Memoria

Tiene una memoria flash la cual varia de acuerdo a la versión, puede andar entre 1kB a 3.5kB de almacenamiento. Posee memoria RAM y EEPROM para el almacenamiento de

datos temporales (RAM) y no temporales (EEPROM). Por lo general tienen una memoria RAM de 128 bytes y una EEPROM de 256 bytes.

E. Periféricos

EL microcontrolador posee cuatro convertidores analógico/digital. Ademas, viene con modulador de ancho de pulso (PWM). Viene equipado con temporizadores internos con un total de dos. Tiene una interfaz de comunicación serial (USART) permitiendo conexiones seriales a nivel software.

F. Arquitectura

Posee un CPU de 8 bits basado en arquitectura Harward lo que brinda mayor velocidad de procesamiento. Ademas, posee arquitectura RISC con pocas instrucciones haciendo simple su programación.

G. Diseño de circuito

Para realizar el laboratorio el cual consiste en emular mediante el microcontrolador PIC12f683 el comportamiento de un dado de seis caras, se realizo la construcción de un circuito conectado y controlado por el microcontrolador. Este tiene una sección que se llamara entrada donde se explicara como se definió la construcción del circuito que maneja el botón y sus parámetros, y otra sección llamada salida que describirá el comportamiento de las salidas y sus conexiones a los LED's.

1) Entrada: El PIN4 se selecciono para ser una GPIO de entrada y por la cual es donde se conectara la lógica analógica implementada para dar inicio al sistema, a su vez, se selecciono este PIN por razones prácticos y topológicos que permitieran un mejor manejo de los otros pines en posibles futuras modificaciones.

El sistema analógico creado para la entrada se trata de el control de la señal de un botón, el cual se implementaron practicas típicas ampliamente conocidas de manejo de señales de pulso tipo el botón conocida como "debouncing" y el circuito se construye unido a la fuente (VDD) que entrega 5V, siendo un valor normal para el funcionamiento del microcontrolador, por lo que fue necesaria la utilización de resistencias a la entrada GPIO.

Las resistencias que se conectan entre la entrada digital y la fuente se les conoce como "pull-up" asegurando que la entrada este desconectada cuando no hay señal de entrada.

El método por el cual nos aseguramos de que el microcontrolador reciba de manera limpia la señal que es afectada cuando se presiona el botón es el "debouncing". Esto es comúnmente utilizado a la hora de utilizar botones o interruptores para purificar la señal de interferencias provocadas al presionar el botón, las cuales son llamadas rebotes, ya que no es un proceso inmediato por lo que puede alterar su señal y comportamiento al sistema. Sabiendo lo anterior, los valores que son comunes en la industria para las resistencias pull-ups son de 120Ω los cuales tomaremos dichos valores para el laboratorio. Basándonos en la teoría de "debouncing" se buscara un retraso aproximado de 36ns el cual permita el comportamiento correcto al presionar el botón. Ademas, para la configuración se necesita un capacitor de seguridad que nos permitirá controlar el tiempo que se necesita mediante la siguiente ecuación:

$$\tau = R * C \tag{1}$$

Al despejar la anterior ecuación, se obtiene un valor de 150pF de capacitancia necesaria para que el funcionamiento sea el correcto. Por lo que el circuito fue construido de la siguiente manera:

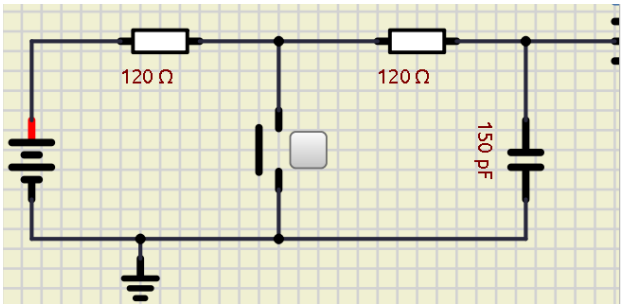

Fig. 3. Circuito para inicializar de la entrada.

2) Salidas: Para generar el resultado al presionar el botón y dar un numero del uno al seis en los LED's y solo disponer de cuatro GPIO de salida y se ocupan seis para representar cada numero se hizo necesario realizar una lógica secuencial para agrupar los LED's y luego a nivel software, de acuerdo al numero obtenido, encender el mismo numero de LED's.

Para ello se conecto a la PN0 un solo LED, a la PN1 dos LED y a la PN2 tres LED, con esta configuración es posible manejar las salidas de forma que se encienda una cantidad de LED's igual al numero obtenido.

Entre cada grupo de LED's y el microcontrolador debe existir una resistencia de seguridad para los dispositivos, por lo cual sabiendo que la tensión entregada por la fuente es de 5V y el máximo de corriente que puede ir a los pines del microcontrolador es de 25mA, si aplicamos la ley de Ohm podremos calcular el valor de dicha resistencia como se demuestra a continuación:

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

con este resultado se obtiene el valor de diseño de dichas resistencias, las cuales por su distribución y para cumplir los requerimientos se implemento una resistencia de 200Ω para el PN0, dos de 400Ω para el PN1 y tres de 25Ω para el PN2.

Con todo lo anterior ya planteado el circuito de salidas es el siguiente:

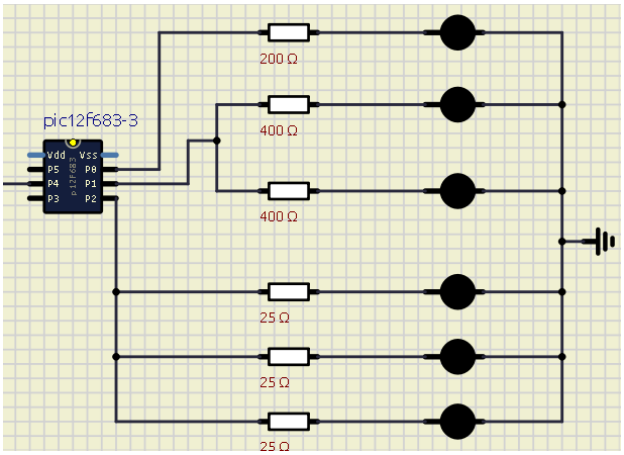

Fig. 4. Circuito para visualizar el resultado del dado.

Por lo que tenemos finalmente el circuito completo para simular el dado de seis caras con un microcontrolador PIC12f683 de la siguiente manera:

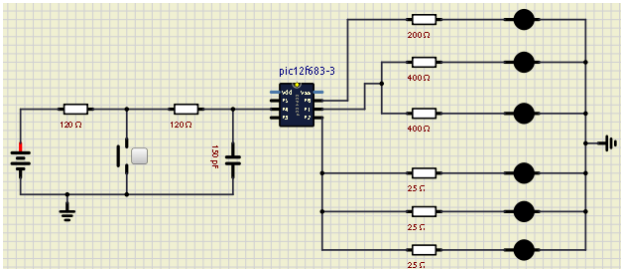

Fig. 5. Circuito final para simular el comportamiento de dado.

H. Lista de componentes

Los componentes y su valor para la realización de este laboratorio son los que se muestran a continuación:

Componente	Cantidad	Precio
PIC12f683	1	¢1200
LEDs	6	¢200
Resistencia 120Ω	2	¢100
Resistencia 200Ω	1	¢45
Resistencia 400Ω	2	¢100
Resistencia 25Ω	3	¢135
Capacitor	1	¢70
Total		**¢1850**

TABLE I
LISTA DE COMPONENTES Y PRECIOS

III. Desarrollo

Se analizara lo realizado en el laboratorio, tanto a nivel software como electrónico, y se vera paso a paso como se comporto y obtuvo cada objetivo. Para ello, se dividirá en dos secciones, una llamada análisis de programa y otra sera análisis electrónico.

1) Análisis de programa: Para programar el microcontrolador PIC12f683 se necesito descargar el compilador SDCC y las librerías que permiten al ambiente poder programar con las instrucciones del MC. Luego de realizado esto, se procedió a crear un archivo en C para darle las tareas necesarias que realizaran el comportamiento de un dado.
A continuación se mostrara partes del código y su explicación:

```
#include <pic14/pic12f683.h>

void delay(unsigned int time);
unsigned char RandomNum();
//Funcion MAIN
void main(void) {
    TRISIO = 0b00010000;
    GPIO = 0x00;

    unsigned char valor;

    while (1) {
        if (GP4 == 0) {
            valor = RandomNum();
            GPIO = 0x00;
```

Fig. 6. Librería e inicializacion de GPIO.

Aquí observamos como se incluye la librería de PIC12f683 y la declaración de la función delay que recibe como parámetro el valor de ”time” que se usara para generar el retraso de respuesta del circuito y así poder visualizar el valor en los LED’s y la declaración de ”RandomNUm” que nos permitirá generar el numero aleatorio mas adelante.
Se observa como se declara la función ”main” y esta contiene la inicializacion de de la entrada PN4 ya que su valor binario para declarar dicha entrada es ”0b00010000” y se ponen en bajo las GPIO para iniciar con el comportamiento esperado. Una vez declarado esto se crea la variable ”valor” la cual guardara el valor obtenido con ”RandomNum”.
Ahora, se inicializa un while donde mediante la lógica manejada mediante un ”if” con la condición de que si GP4 es 0 sera la condición de cuando se toca el botón y buscara guardar en ”valor” lo que tenga ”RandomNum” el cual dará acceso a la siguiente serie de casos:

```
switch (valor) {
    case 1:
        GP0 = 1;
        break;

    case 2:
        GP1 = 1;
        break;

    case 3:
        GP2 = 1;
        break;

    case 4:
        GP0 = 1;
        GP2 = 1;
        break;

    case 5:
        GP1 = 1;
        GP2 = 1;
        break;

    case 6:
        GP0 = 1;
        GP1 = 1;
        GP2 = 1;
        break;
}

delay(400);
GPIO = 0x00;
```

Fig. 7. Lógica de funcionamiento de los LED's.

Básicamente, de acuerdo al valor guardo por la variable "valor" se accederá al caso equivalente a dicho valor, donde encenderá las salidas que se reflejara en los LED's y en esta configuración de seis casos abarca cada resultado posible en un dado de seis caras. Finalmente, se agrega el valor de delay para observar el comportamiento y limpiar los registros utilizados.

```
void delay(unsigned int time) {
    unsigned int i;
    unsigned int j;

    // Bucle para controlar el tiempo de retardo.
    for (i = 0; i < time; i++) {
        for (j = 0; j < 1400; j++); // Tiempo para
    }
}

// Funcion generadora de números pseudoaleatorios d
unsigned char RandomNum() {
    static unsigned char lfsr = 0x07;
    unsigned char random_b;

    do {
        random_b = ((lfsr >> 0) ^ (lfsr >> 1)) & 1;
        lfsr = (lfsr >> 1) | (random_b << 2);
    } while (lfsr > 6);

    return lfsr;
```

Fig. 8. Función de manejo del delay y función lfsr.

Observamos como se definió la función delay, recibiendo como parámetro el valor "time". Se declaran dos variables "i" y "j" las cuales mediante un bucle "for" anidando otro "for" manejan el tiempo de retraso y el retraso deseado. Se entra al bucle dando el parámetro "time" como condición de salida del bucle por lo que durara el bucle el valor que se le haya dado a dicho parámetro y el siguiente bucle es el tiempo que se retrasara que sera lo que le tome a "j" llegar al valor de condición de salida dado.

La función "RandomNum" es la encargada de generar números aleatorios para cuando el botón es presionado, como no existe una instrucción especifica que genere números aleatorios, se necesita crear una función que simule esto para un rango de números predefinidos y esta función fue hecha para cumplir dicha función. El algoritmo utilizado es conocido como lfsr (desplazamiento de registro de retroalimentación lineal) el cual es utilizado para generar números pseudoaleatorios el cual consiste en guardar en una variable un valor fijo, dependiendo del numero de valores aleatorios deseados, y usando este registro utilizar álgebra booleana, extrayendo los bits menos significativo y el siguiente y haciendo una XOR para obtener un valor de un solo bit y la mascara con uno para asegurar que sea un valor valido, toda esta lógica se guarda en una variable y es lo que define el numero generado. Lo siguiente es la retroalimentación, la cual es la encargada de que siempre se modifique el valor con cada iteración. Finalmente se crea un bucle while que se asegura que no se salga de los parámetros esperados.

2) Análisis electrónico: En esta sección se mostrara los diferentes resultados del comportamiento del sistema al presionar el botón y lo que muestra. A continuación se muestra las diferentes salidas obtenidas y posibles:

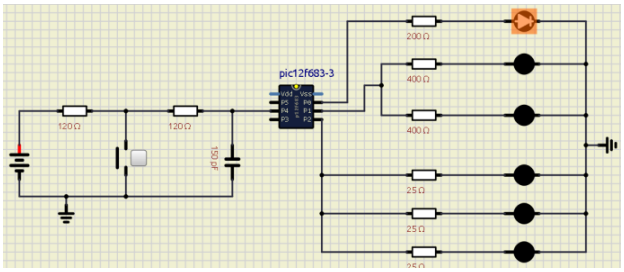

Fig. 9. Salida cuando el valor del dado es uno.

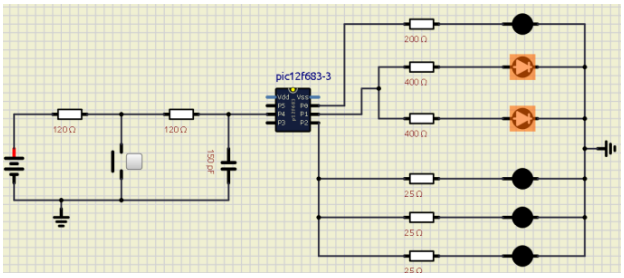

Fig. 10. Salida cuando el valor del dado es dos.

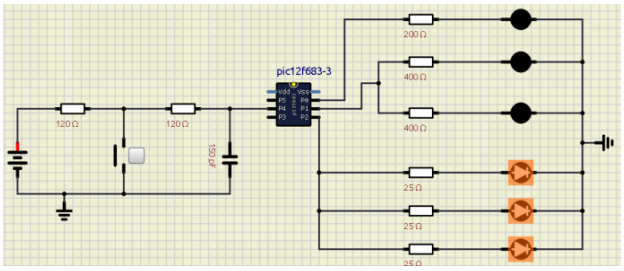

Fig. 11. Salida cuando el valor del dado es tres.

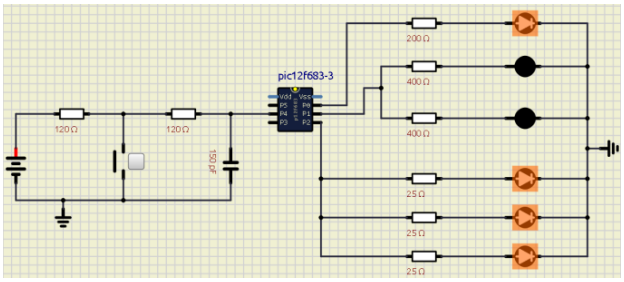

Fig. 12. Salida cuando el valor del dado es cuatro.

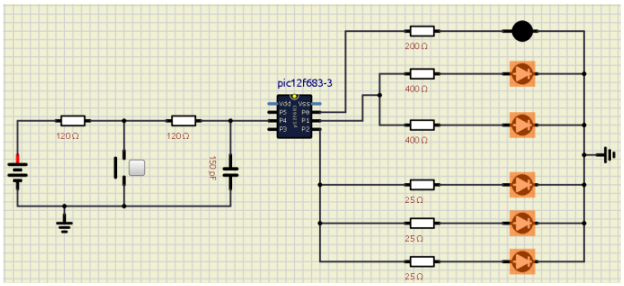

Fig. 13. Salida cuando el valor del dado es cinco.

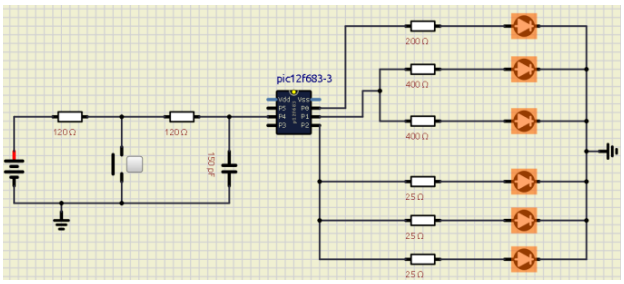

Fig. 14. Salida cuando el valor del dado es seis.

IV. Conclusiones

El laboratorio de introducción a microcontroladores y manejo de GPIO mediante la implementación del MCU PIC12f683 para simular el comportamiendo de un dado de seis caras al ser lanzado abarco temas de instalación y manejo de paquetes y librerías para el funcionamiento de PIC, ademas de la generación de firmware implementando las herramientas vistas en clase, así como creando código para la lógica deseada del comportamiento y manejo del MCU y también el uso de plataformas de simulación de microcontroladores como lo es SimulIDE que facilita las pruebas de sistemas antes de implementarlo de manera física.

Se obtuvieron los resultados esperados en el comportamiento del MCU PIC12f683, las técnicas utilizadas en el programa para la generación de números aleatorios, así como el razonamiento usado para la entras y salidas se comporto como se presupuestaba, logrando los resultados deseados.

La utilización de la plataforma SimulIDE y la construcción del circuito funciono bien para los propósitos solicitados, con los valores adecuados de diseño hizo lo que se buscaba lograr. Finalmente, como recomendación para futuras mejoras, se puede mejorar aun la lógica detrás de la generación de números aleatorios, aunque funciona, puede ser aun mejorado para ser mas preciso a la hora de manifestar su aleatoriedad.

References

[1] Microchip Technology Inc., "PIC12f683 Data Sheet" Microchip Technology Inc., 2024. https://www.microchip.com/wwwproducts/en/PIC12f683