Laboratorio 1: Introducción a Microcontroladores y manejo de GPIOs

1st Ronny Granados
Escuela de Ingeniería Eléctrica
Universidad de Costa Rica.
San José, Costa Rica
C03505
ronny.granadosperez@ucr.ac.cr

I. Introducción/Resumen

El laboratorio consistió en la elaboración de un simulador de un dado utilizando LEDs, resistencias, un botón y un microcontrolador PIC12f683. Para poder lograr el funcionamiento deseado fueron necesarios 4 pines que fueron tomados como salidas del MCU y un pin que fue utilizado como entrada en donde se pudo leer el estado del botón. Mediante lenguaje de programación C fue implementada la lógica del dado y se utilizó un algoritmo de generación de números aleatorios para poder dar un tipo de aleatoriedad al dado, tal como ocurre con uno real.

II. Nota Teórica

II-A. PIC12f683

Este microcontrolador cuenta con 8 pines. Siguiendo el diagrama de la figura 1, los pines 1 y 8 son los pines de VDD y VSS respectivamente y se encargan de la alimentación del MCU; los pines GP0, GP1, GP2 y GP4 se configuran como salidas de las cuales se obtienen 5 V; el pin GP5 fue configurado como una entrada y recibe también 5 V que son alternados mediante un switch en configuración pull-down.

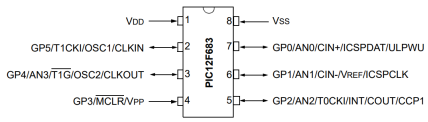

Figura 1. Diagrama de pines del MCU.

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 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: PDIS = VDD x {IDD – ∑ IOH} + ∑ {(VDD – VOH) x IOH} + ∑(VOl x IOL).

† 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 2. Características Eléctricas del PIC12f683 [2].

II-A1. Registros GPIO y TRISIO: Según la hoja de datos del MCU, el registro GPIO se utiliza para estabecer el estado de los pines I/O en donde "1" significa que el pin estará en ALTO, obteniéndose 5 V y "0" para tener el pin en BAJO. Por otra parte, el registro TRISIO es utilizado para definir si se desea establecer los pines como entradas o salidas.

Al inicio del código fuente (`*.c`) se colocan todos los pines como salidas (a excepción del GP5 que se configura como una entrada) y todos en bajo, de la siguiente forma `TRISIO = 0x20` y `GPIO = 0x00` al inicio de la función `main()`. Posteriormente este registro GPIO es modificado a conveniencia para poder iluminar los LEDs de forma que se obtenga la configuración deseada de un dado.

II-B. Diseño Electrónico

Para el diseño se van a utilizar 7 LEDs que serán controlados mediante 4 periféricos, esto pues se parte del hecho que para obtener la típica configuración de un dado real.

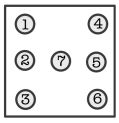

Figura 3. Enumeración de los LEDs para configurar periféricos.

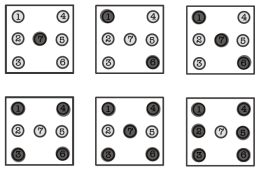

Figura 4. Combinaciones del dado y respectivos LEDs encendidos.

De la figura 4 se puede ver cómo las esquinas superior e inferior de lados opuestos siempre ocurren a la vez, al igual que los LEDs centrales en la combinación correspondiente al número 6. Por lo tanto, se deciden utilizar los pines GP0, GP1, GP2, GP4 para controlar los 7 LEDs, de la siguiente forma:

Cuadro I
PINES UTILIZADOS Y LEDS QUE CONTROLAN

Pin	LEDs Controlados
GP0	7
GP1	1 y 6
GP0	3 y 4
GP0	2 y 5

Una vez establecida este orden, se debe de establecer cómo se necesita que se comporte el registro GPIO para poner en ALTO un pin o múltiples pines a la vez y lograr así, con los LEDs, una configuración igual a la de la figura 4:

Cuadro II
ESTADO DEL REGISTRO GPIO DE ACUERDO AL NÚMERO DESEADO EN EL DADO.

Número Deseado	Reg. GPIO
1	0x01
2	0x02
3	0x03
4	0x06
4	0x07
6	0x16

Hecho este diseño, se puede implementar en hardware en el simulador, pero hay que tomar en cuenta que se debe de cuidar el estado de los LEDs protegiéndolos con una resistencia que vaya acorde con la tensión que reciben y la corriente máxima que se desea que pase sobre estos, respetando las especificaciones eléctricas del microcontrolador (figura 2). Mediante ley de ohm, si se sabe que $5\,\mathrm{V}$ son los que alimentan a los LEDs y cada uno requiere de $1{,}2\,\mathrm{V}$ para poder encenderse y que la corriente máxima que circula por estos es de $20\,\mathrm{mA}$, se calcula la resistencia del LED de la siguiente forma:

$$R_{LEDs} = \frac{(5\,\mathrm{V} - 1{,}2\,\mathrm{V} \cdot 1{,}2\,\mathrm{V})}{20\,\mathrm{mA}} = 178\,\Omega$$

$$R_{LED_7} = \frac{(5\,\mathrm{V} - 1{,}2\,\mathrm{V})}{20\,\mathrm{mA}} = 190\,\Omega$$

Por último, para la configuración pull-down del botón conectado a GP5, es recomendable utilizar una resistencia para poder así desviar la corriente que pueda ser enviada al dispositivo y dejar únicamente (de manera ideal) la tensión. Un valor de resistencia pull-down recomendado según [3] es de $4{,}7\,\mathrm{k}\Omega$. Como todo switch tiene un efecto de rebote, este se debe de amortiguar con un capacitor en paralelo con la resistencia Pull-Down, según [1], de $0{,}1\,\mu\mathrm{F}$.

Por lo tanto, el diseño final queda de la siguiente forma:

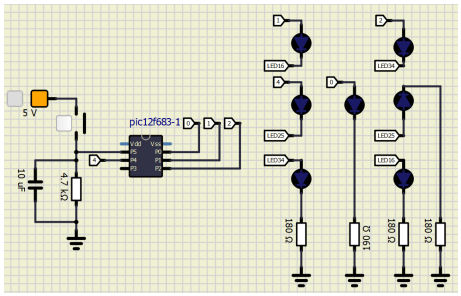

Figura 5. Diseño electrónico del dado simulado.

Los componentes utilizados se muestran en la siguiente tabla:

Cuadro III
COMPONENTES ELECTRÓNICOS NECESARIOS PARA EL DISEÑO

Componente	Cantidad	Precio (Unidad)
PIC12f683	1	$8.99 (2 pcs)
Resistencia $220\,\Omega$	4	$5.49 (100 pcs)
Resistencia $4{,}7\,\mathrm{k}\Omega$	1	$5.99 (100 pcs)
LED Rojo	7	$6.99 (100 pcs)
Capacitor $0{,}1\,\mu\mathrm{F}$	1	$5.68 (25 pcs)
Switch	1	$6.98 (12 pcs)

III. DESARROLLO/ANÁLISIS DE RESULTADOS

Para poder llevar a cabo el simulador del dado, se debió de programar en lenguaje C el correcto funcionamiento de este. Primeramente, para poder tener un mejor manejo de lo establecido en la tabla II

```
// Se definen los pines en variables
#define one 0b00000001
#define two 0b00000010
#define three 0b00000011
#define four 0b00000110
#define five 0b00000111
#define six 0b00010110

unsigned int led_matrix[] = {one, two, three,
                             four, five, six};
```

La lógica detrás del dado mostrando los distintos números aleatorios cae en el loop while dentro de la función main. Este detecta cuando el pin GP5 esta en ALTO y genera un número aleatorio mediante la función `rand()`:

```
while (1)
{
    if (GP5 == 1)
    {
```

```
        seed += rand_counter++;
        unsigned int random = rand();
        random = (random % 6);

        GPIO = led_matrix[random];
        delay(time);
        GPIO = 0x00;
    }
}
```

Por otra parte, la función `rand()` utiliza el algoritmo de Bum-Bum-Shub para generar dichos números. Para ello es necesario una semilla y dos números distintos **q** y **q**, tal que la función se ve de la siguiente manera:

```
unsigned int rand(){
    unsigned int p = 9802;
    unsigned int q = 2348;
    unsigned int random_num;

    random_num = (seed*seed) % p*q;

    return random_num;
}
```

A continuación, se muestran capturas de pantalla del funcionamiento del programa, en donde se puede ver cómo el dado muestra los números 1, 3 y 5 en los LEDs mientras el switch está cerrado:

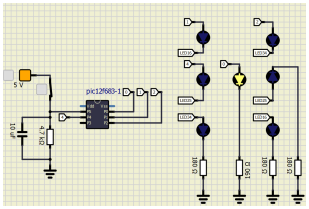

Figura 6. Dado mostrando el número 1.

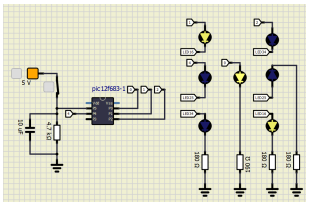

Figura 7. Dado mostrando el número 3.

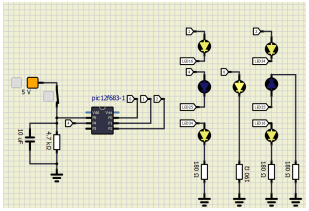

Figura 8. Dado mostrando el número 5.

IV. Conclusiones y Recomendaciones

El programa funciona de manera correcta, cumple con lo deseado y se respetan las limitaciones físicas del microprocesador, sin embargo, la única parte en la que se tuvo problemas fue en la generación de números aleatorios pues si bien la función sí genera números pseudo aleatorios, estos no cambian mucho entre una corrida y otra, pues la implementación del algoritmo carece de algún mecanismo que haga cambiar la semilla cada cierto tiempo, sin embargo, el resto del programa funciona adecuadamente y el diseño electrónico es muy limpio y ordenado.

Como recomendaciones están el plasmar primero en papel lo que se desea hacer y no comenzar a programar de manera inmediata pues esto puede generar mucha confusión, es mejor ir poco a poco con cada parte del programa, pensando qué estructuras de datos se ajustan mejor a lo que se quiere lograr, en el caso de este diseño, un array de números enteros funcionó bien, pero pudo haberse implementado una función que hiciera esto, entre otras cosas. También recomiendo utilizar alguna alternativa más analógica a la generación de números aleatorios, algún método que dependa del azar y de magnitudes físicas pues generarlos con software es mucho más complejo y mucho más cuando no se tiene acceso a librerías que hagan esto de forma automática.

REFERENCIAS

[1] Jens Christoffersen. Switch Bounce and How to Deal with It. https://www.allaboutcircuits.com/technical-articles/switch-bounce-how-to-deal-with-it/, 2015.

[2] Microchip. PIC12f683. https://ww1.microchip.com/downloads/en/devicedoc/41211d_.pdf, 2007.

[3] Naylamp Mechatronics. Resistencias Pull-Up y Pull-Down. https://naylampmechatronics.com/blog/39_resistencias-pull-up-y-pull-down.html, 2017.