Universidad de Costa Rica

Facultad de Ingeniería
Escuela de Ingeniería Eléctrica

IE-0624
Laboratorio de Microcontroladores

Laboratorio #1: Introducción a microcontroladores y manejo de GPIOS

Gabriel Alberto Barahona Otoya B70896

Profesor:
Marco Villalta Fallas

II-Ciclo, 2022

Índice

1. Introducción **2**
1.1. Resumen . . . 2
1.2. Conclusiones importantes . . . 2

2. Nota teórica **2**
2.1. Información general del Microcontrolador . . . 2
2.2. Registros . . . 4
2.2.1. TRISIO . . . 4
2.2.2. GPIOs . . . 5
2.2.3. CONFIG . . . 5
2.3. Periféricos . . . 6
2.3.1. Decodificador BCD a 7 segmentos . . . 6
2.3.2. Display de 7 segmentos . . . 7
2.3.3. Compuerta NOT . . . 8
2.4. Diseño del circuito . . . 8
2.5. Lista de componentes y precios . . . 9
2.6. Conceptos/temas del laboratorio . . . 9

3. Desarrollo/Análisis de resultados **9**
3.1. Programa . . . 9
3.2. Componentes . . . 10
3.3. Demostración de funcionamiento . . . 11

4. Conclusiones y recomendaciones **13**

Referencias **13**

5. Anexos **14**
5.1. GITHUB . . . 14
5.2. Código del laboratorio . . . 14

1. Introducción

1.1. Resumen

El objetivo principal del laboratorio es brindar una introducción sobre las funcionalidades y el uso correcto de los microcontroladores, la manipulación de GPIOS (General Purpose Input Outputs), generación de números aleatorios y el flujo de desarrollo que se solicita en prácticas dirigidas. El trabajo realizado consiste en el desarrollo de un simulador de tómbola simplificada de bingo utilizando como componentes principales un microcontrolador PIC12F675, un botón que al presionarlo activará 2 displays de 7 segmentos que se encargarán de mostrar un número aleatorio entre el 00 y el 99. Al haber mostrado 16 números se muestra un parpadeo del número 99 y se reinicia el juego.

1.2. Conclusiones importantes

Debido a que el microcontrolador es un poco limitado en ciertos aspectos, como por ejemplo en la cantidad de pines de entrada y salida de información y la cantidad de memoria disponible, se utilizaron otros componentes electrónicos para lograr el correcto funcionamiento de lo solicitado en laboratorio. Cada uno de estos componentes adicionales se detallarán en el aparatado de nota teórica.

2. Nota teórica

2.1. Información general del Microcontrolador

Los fabricantes de este componente lo catalogan como un potente microcontrolador de 8 bits basado en CMOS Flash (ejecución de instrucciones en 200 nanosegundos), fácil de programar (solo 35 instrucciones de una sola palabra) y que incluye la potente arquitectura PIC® MCU de Microchip en un paquete de 8 pines y cuenta con 4 canales para el convertidor analógico-digital (A/D) de 10 bits, comparador de 1 canal y 128 bytes de memoria de datos EEPROM. Este dispositivo es adaptado con facilidad a aplicaciones industriales, de electrodomésticos y de productos básicos de consumo que requieren ser reprogramables [2]. Se programa en lenguaje C.

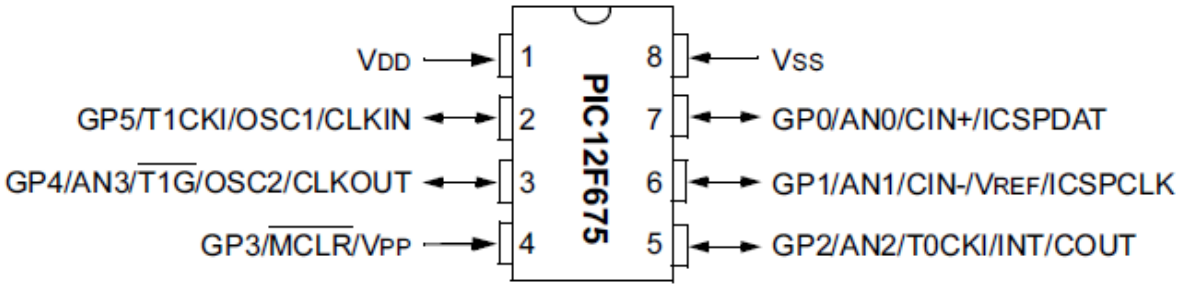


Figura 1: Diagrama de pines del PIC12F675 [2]

Información general [3]:

- Fabricante: Microchip

- Serie: PIC12F675
- Tamaño de memoria del programa: 1.75 kB
- Ancho de bus de datos: 8 bit
- Resolución del conversor de señal analógica a digital (ADC): 10 bit
- Frecuencia de reloj máxima: 20 MHz
- Número de entradas/salidas: 6 I/O
- Tamaño de RAM de datos: 64 B
- Voltaje de alimentación - Mín.: 2 V
- Voltaje de alimentación - Máx.: 5.5 V
- Temperatura de trabajo mínima: - 40 C
- Temperatura de trabajo máxima: + 85 C
- Tamaño de ROM de datos: 128 B
- Tipo de Rom de datos: EEPROM
- Cantidad de canales del conversor de señal analógica a digital (ADC): 4 Channel
- Cantidad de temporizadores/contadores: 2 Timer
- Tipo de memoria de programa: Flash
- Temporizadores de vigilancia: Watchdog Timer

Absolute Maximum Ratings†

Ambient temperature under bias	-40 to +125°C
Storage temperature	-65°C to +150°C
Voltage on V_{DD} with respect to V_{SS}	-0.3 to +6.5V
Voltage on $\overline{MCLR}$ with respect to Vss	-0.3 to +13.5V
Voltage on all other pins with respect to V_{SS}	-0.3V to (V_{DD} + 0.3V)
Total power dissipation(1)	800 mW
Maximum current out of V_{SS} pin	300 mA
Maximum current into V_{DD} pin	250 mA
Input clamp current, I_{IK} (V_I < 0 or V_I > V_{DD})	± 20 mA
Output clamp current, I_{OK} (Vo < 0 or Vo > V_{DD})	± 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 all GPIO	125 mA
Maximum current sourced all GPIO	125 mA

Figura 2: Características eléctricas [2]

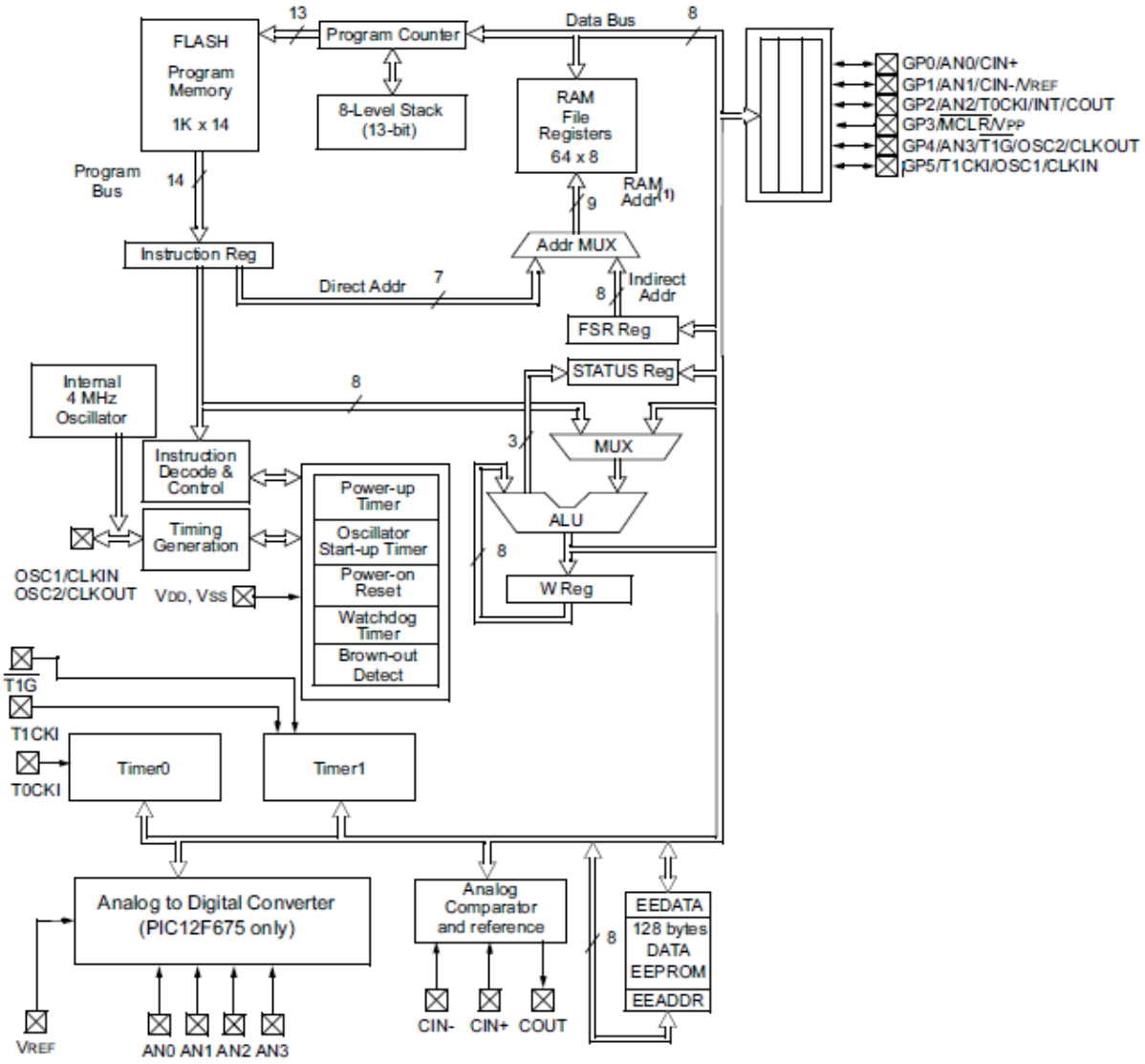


Figura 3: Diagrama de bloques del PIC12F675 [2]

2.2. Registros

El PIC12F675 consta de varios registros que se pueden operar digitalmente. A continuación, se va a explicar con detalle el funcionamiento de cada uno de los registros utilizados para la realización del laboratorio:

2.2.1. TRISIO

Es uno de los registros más importantes ya que este permite establecer el modo de operar de cada pin. Un bit en alto (1) configura al pin como una entrada y un bit en bajo (0) como una salida [1].

REGISTER 3-2: TRISIO — GPIO TRISTATE REGISTER (ADDRESS: 85h)

U-0	U-0	R/W-x	R/W-x	R-1	R/W-x	R/W-x	R/W-x
—	—	TRISIO5	TRISIO4	TRISIO3	TRISIO2	TRISIO1	TRISIO0
bit 7							bit 0

bit 7-6: **Unimplemented**: Read as '0'

bit 5-0: **TRISIO<5:0>**: General Purpose I/O Tri-State Control bit

1 = GPIO pin configured as an input (tri-stated)

0 = GPIO pin configured as an output.

Note: TRISIO<3> always reads 1.

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

Figura 4: Registro TRISIO [2]

2.2.2. GPIOs

Los GPIOs (General Purpose Input Output) corresponden a la mayoría de pines que conforman el microcontrolador. Los pines restantes del dispositivo normalmente son para la alimentación del mismo (Vdd y Vss), reset, entradas del reloj, entre otros. Estos pines tienen modos de operación similares pero normalmente poseen distintas funcionalidades por lo que el programador decide cual de todas es la que quiere usar mediante registros de configuración. Es así como con el registro GPIO se realiza una lectura del estado del pin y se escribe al latch de salida [1].

REGISTER 3-1: GPIO — GPIO REGISTER (ADDRESS: 05h)

U-0	U-0	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
—	—	GPIO5	GPIO4	GPIO3	GPIO2	GPIO1	GPIO0
bit 7							bit 0

bit 7-6: **Unimplemented**: Read as '0'

bit 5-0: **GPIO<5:0>**: General Purpose I/O pin.

1 = Port pin is >VIH

0 = Port pin is <VIL

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

Figura 5: Registro GPIO [2]

2.2.3. CONFIG

Dependiendo de la función que se le quiere dar al pin se debe configurar el registro CONFIG. Para esto, se utilizan instrucciones de preprocesador como macros, que se encuentran definidas en los encabezados del microcontrolador que utiliza el compilador. Para la realización del laboratorio se van a deshabilitar tanto WDT como MCLRE. El PIC12F675 tiene por defecto una configuración de watchdog timer habilitado (WDT) que es un tipo de temporizador que se utiliza comúnmente para poder llevar un control en la operación del microcontrolador y evitar que este se quede enciclado. El pin GPIO3 que sólo funciona como entrada, inicia con una configuración de reset

(MCLRE). Este registro pertenece al espacio de configuración de memoria que es accesado durante la programación [1].

2.3. Periféricos

2.3.1. Decodificador BCD a 7 segmentos

Los circuitos decodificadores son uno de los componentes comerciales más populares y son utilizados para disminuir la dificultad de conexión y uso de los display de 7 segmentos. Se les proporciona una entrada en formato BCD y posee la capacidad de encender y apagar cada uno de los segmentos que poseen este tipo de displays [7]. Los códigos BCD se utilizan para representan datos en valores decimales a formato binario por lo que se utilizan grupos de 4 bits para hacer la representación de cada número del 0 al 9 [7].

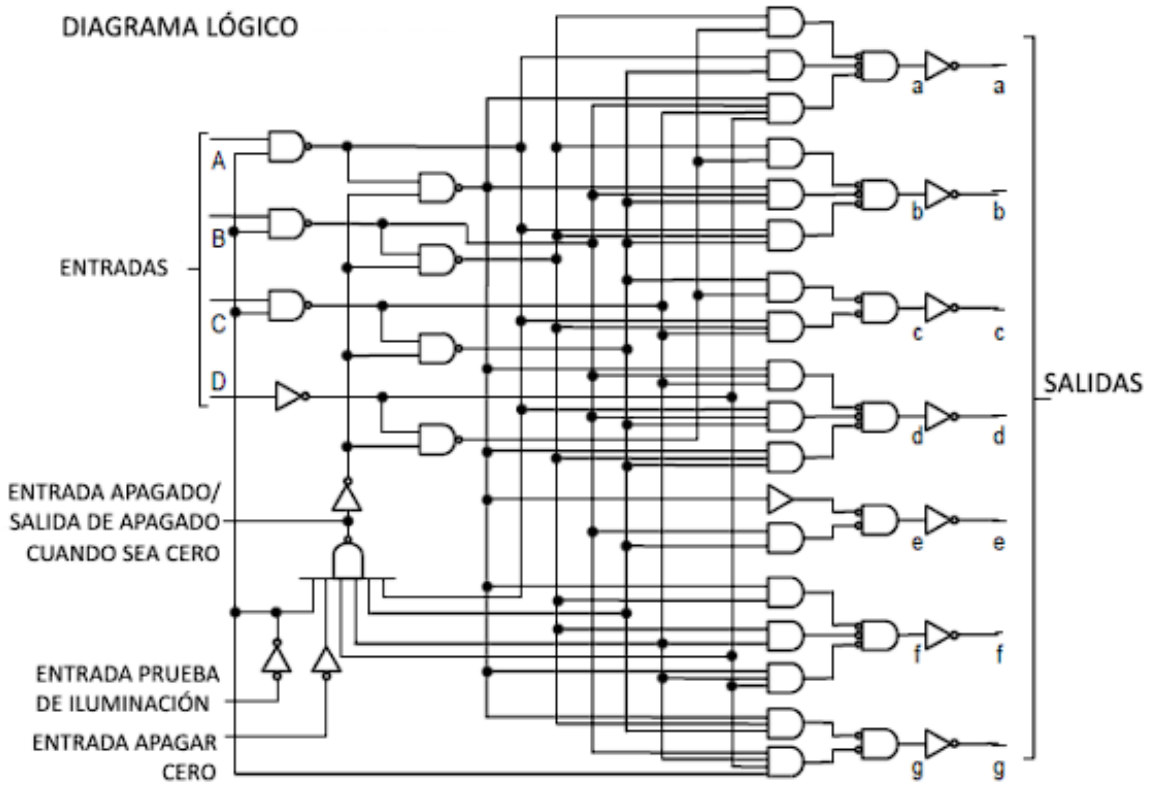
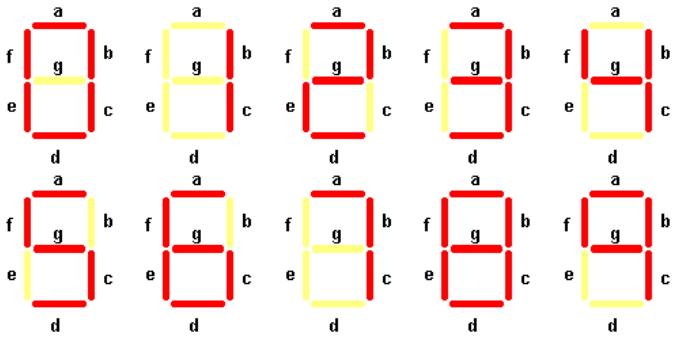


Figura 6: Circuito interno del decodificador [7]



ENTRADAS				SALIDAS							NUMERO
A3	A2	A1	A0	a	b	c	d	e	f	g	
0	0	0	0	1	1	1	1	1	1	0	0
0	0	0	1	0	1	1	0	0	0	0	1
0	0	1	0	1	1	0	1	1	0	1	2
0	0	1	1	1	1	1	1	0	0	1	3
0	1	0	0	0	1	1	0	0	1	1	4
0	1	0	1	1	0	1	1	0	1	1	5
0	1	1	0	1	0	1	1	1	1	1	6
0	1	1	1	1	1	1	0	0	0	0	7
1	0	0	0	1	1	1	1	1	1	1	8
1	0	0	1	1	1	1	0	0	1	1	9
1	0	1	0	x	x	x	x	x	x	X	x
1	0	1	1	x	x	x	x	x	x	x	x
1	1	0	0	x	x	x	x	x	x	x	x
1	1	0	1	x	x	x	x	x	x	x	x
1	1	1	0	x	x	x	x	x	x	x	x
1	1	1	1	x	x	x	x	x	x	x	x

Figura 7: Tabla de verdad del decodificador

2.3.2. Display de 7 segmentos

Es un componente que consta de un arreglo de 7 diodos LEDs rectangulares colocados en forma de 8. Cada LED o segmento esta directamente relacionado a una letra de la a a la g. También poseen un punto que se representa mediante dp [7].

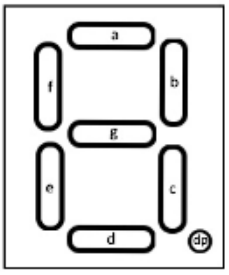


Figura 8: Display de 7 segmentos [7]

Para la realización del laboratorio se utilizará un display doble de 7 segmentos de cátodo común, es decir, todos los cátodos de los diodos mencionados se encuentran conectados entre si [7].

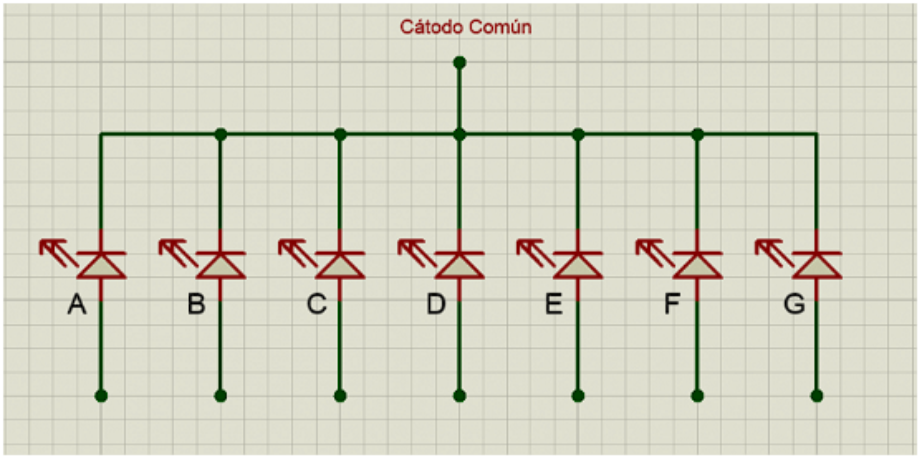


Figura 9: Display de 7 segmentos con cátodo común [7]

2.3.3. Compuerta NOT

Es parte de la familia de compuertas lógicas. Implementa la negación lógica es decir, cada vez que su entrada tiene un valor de 0 o en bajo, su salida estará en 1 o en alto [6].

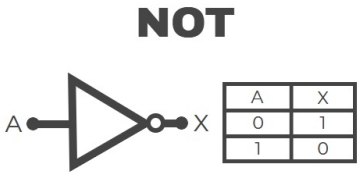


Figura 10: Compuerta NOT

2.4. Diseño del circuito

A continuación se muestra el diseño del circuito. Su funcionamiento se explica más a detalle en el apartado de análisis de resultados.

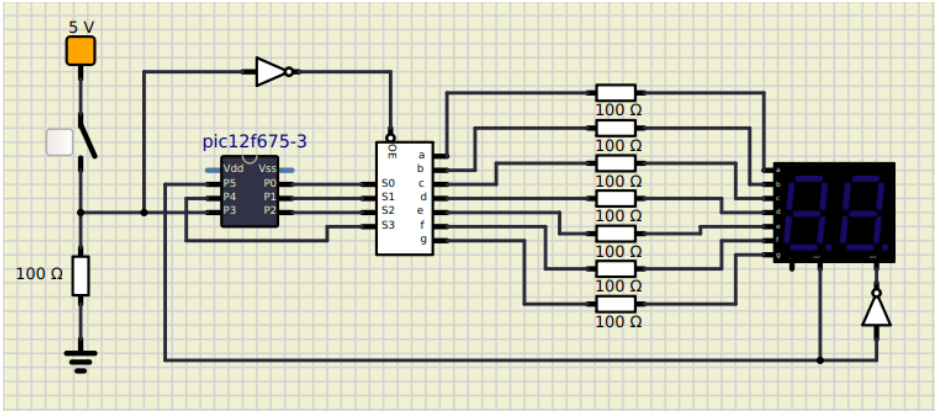


Figura 11: Circuito en SimulIDE

2.5. Lista de componentes y precios

Los precios fueron extraídos de la página web de Microchip [2], Texas Instruments [4] y de la tienda Teltron [5] ubicada en San José.

- Componente: PIC12F675 / Precio: $1.31 (1-24)
- Componente: Decodificador BCD a 7 segmentos / Precio: ¢1.509,00 (1)
- Componente: Display de 7 segmentos doble / Precio: ¢5.934,00 (1)
- Componente: 2 Compuertas NOT / Precio: $0.092 (1—99)
- Componente: Fuente de 5 V
- Componente: Switch / Precio: ¢599,00 (1)
- Componente: 8 Resistencias de 100 Ohms / Precio: ¢299,00 (10)

2.6. Conceptos/temas del laboratorio

Pseudo-aleatorio

3. Desarrollo/Análisis de resultados

3.1. Programa

El código del laboratorio se puede encontrar en el apartado de Anexos. En esta sección se va a explicar el funcionamiento de este y cada una de las partes que lo conforman.

Primero, todo programa debe iniciar con la inclusión de la librería del microcontrolador que se va a utilizar en el laboratorio, en este caso se utiliza pic14/pic12f675.h. Se reutilizó una de las funciones vistas en el ejemplo de clase la cual genera un delay o retardo en la señal. Luego, se modifica el registro CONFIG para deshabilitar el watchdog timer (WDT) y el MCLRE (reset) porque el GP3 solamente se va a utilizar como pin de entrada. Posteriormente se inicializan 4 variables globales, la primera es n que determina a cual de todos los case del switch se va ejecutar. La variable count realiza un conteo de las veces que se presiona el botón para mostrar un número. Las variables num1 y num2 guardan el primer y segundo dígito del número del número a mostrar.

Luego de esto, se entra a la función principal void main, se asignan los tiempos de delay que se van a utilizar. Se configura el registro TRISIO con 0x08 para determinar que el GP3 va a funcionar como pin de entrada y los demás como pin de salida. Ahora lo que hace el programa es preguntarse si GP3 se encuentra en un estado de bajo (0) o alto (1). Si el estado está en bajo, el programa va a estar ejecutando el switch y cada uno de los cases que lo conforman. Cada case es un número del 00 al 80 y la razón por la cual se llegó hasta ese número es porque al implementarlo hasta 99, la ejecución del código mostraba error. Si n es menor o igual a 80, se le suma 1 a n para cambiar de case, pero si n es mayor a 80, reinicia el valor de n a 0 para volver a pasar por todos los números. Por otro lado, si el estado de GP3 es 1 y count es menor o igual a 16, se configura el registro GPIO para que muestre num1 y num2 con un tiempo bastante rápido para lograr observar dos números que pueden ser distintos y se le suma 1 a count para llevar un registro de cuantos números han salido hasta el momento. Si count es mayor a 16, lo que hace el programa es mostrar el número 99

y reinicia a count con un valor de 0. Esto último se encuentra programado en el código sin embargo al presionar más de 16 veces el switch, no se despliega el 99.

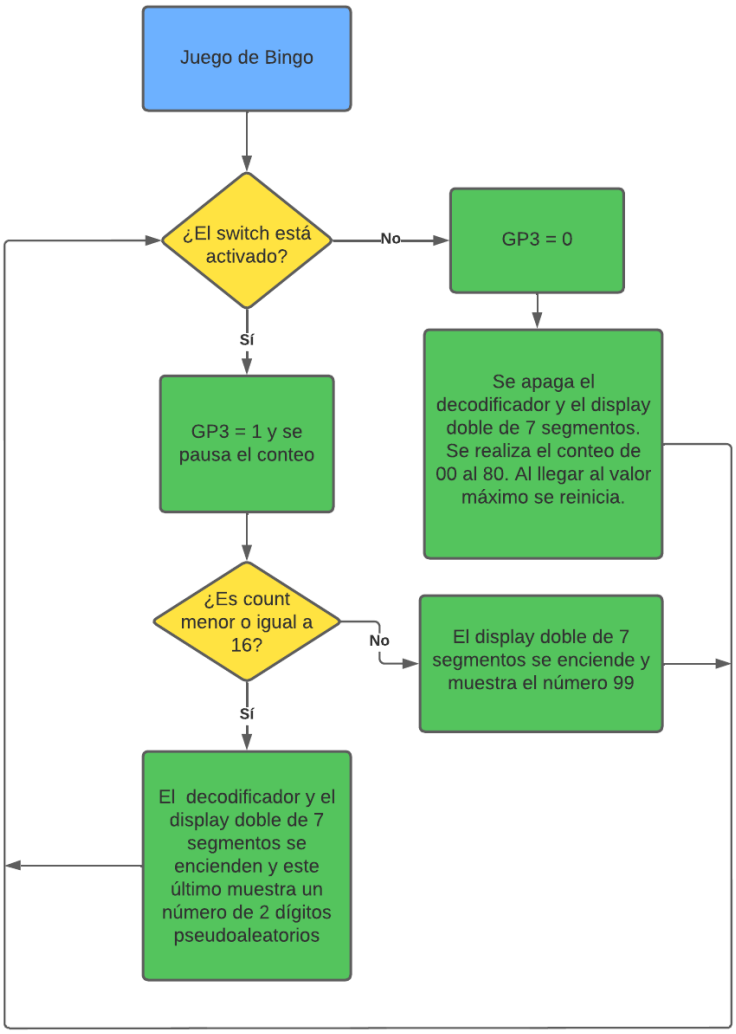


Figura 12: Diagrama de flujo

3.2. Componentes

En esta sección se explicará a detalle la razón por la cual se están utilizando cada uno de los componentes.

En la parte izquierda se puede observar un switch que funciona con una resistencia de pull down. En esta configuración, el estado del pin GP3 es en bajo hasta que ocurre un evento que lo pone en alto, lo cual sucede al presionar el switch que se encuentra conectado a una fuente de 5V. A esta etapa también se le conecta una compuerta NOT para encender o apagar el decodificador BCD a 7 segmentos. Esto se hace para mantener apagado el display y que los jugadores no puedan observar el cambio en los números en el funcionamiento interno del PIC. Al presionar el switch y cerrar el contacto del circuito, se enciende el decodificador y el display. Al deshabilitar el switch, se apagan ambos componentes mencionados pero el PIC continua su funcionamiento cambiando los números.

Continuando con otra parte del circuito, se utilizaron los pines GP0, GP1, GP2 y GP4 del PIC12F675 para conectarlos al decodificador BCD a 7 segmentos. Utilizando solamente 4 pines de salida de información y utilizando este decodificador se pueden conseguir los números del 0 al 9 en el display con su respectiva configuración del registro GPIO. A la salida del decodificador se utilizaron varias resistencia de 100 Ohms cuya finalidad es la protección, ya que el display en su circuito interno posee un diodo LED para iluminar cada segmento.

Finalmente, el funcionamiento del pin GP5 en conjunto con la compuerta NOT conectados al cátodo común de cada display de 7 segmentos consiste en encender y apagar este último elemento a una velocidad tan rápida que no se puede llegar a observar. Esta técnica se hace para obtener 2 números distintos en ambos display ya que estos se encuentran conectados en paralelo y ambos no pueden estar encendidos a la vez porque mostrarían el mismo número.

3.3. Demostración de funcionamiento

Para iniciar con la simulación se presiona el botón de Power Circuit. Al activar el switch se va a mostrar un número del 00 al 80 y se va a mantener en el display. Al desactivar el switch, el programa retoma el conteo de los números de manera interna y se apaga el display.

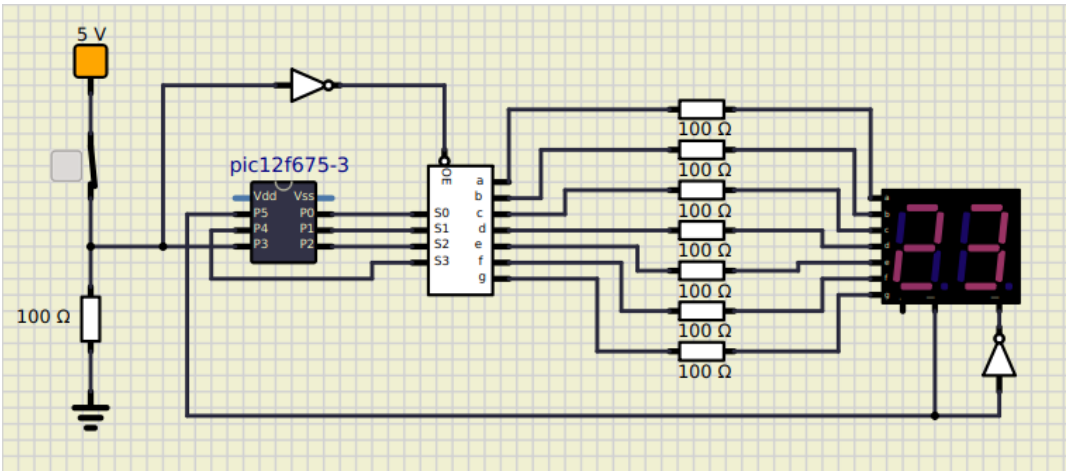


Figura 13: Primer número mostrado

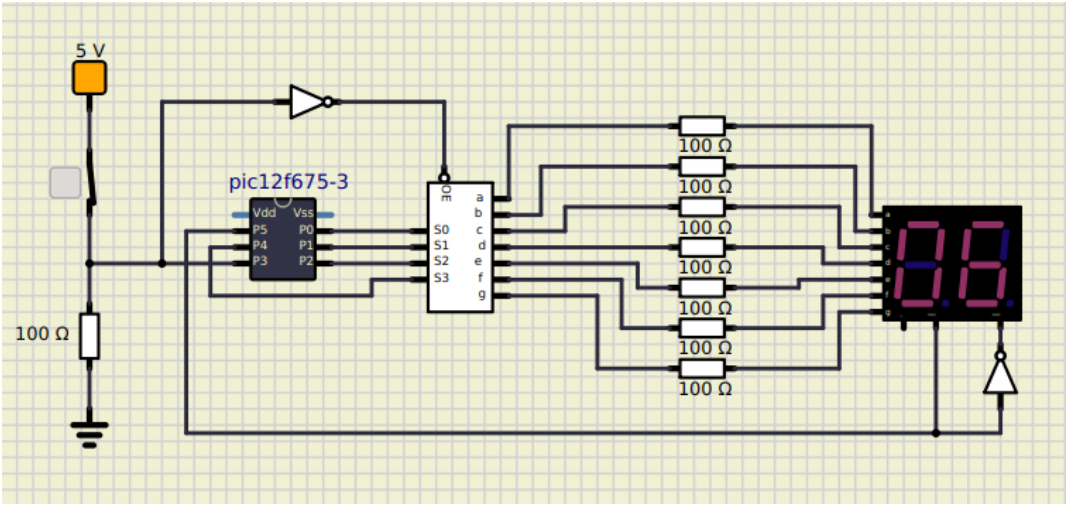


Figura 14: Segundo número mostrado

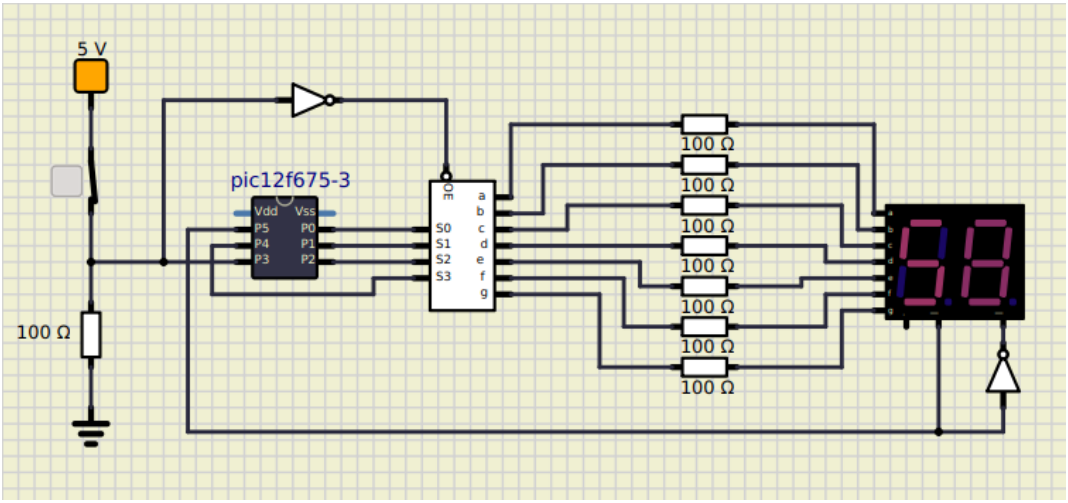


Figura 15: Tercer número mostrado

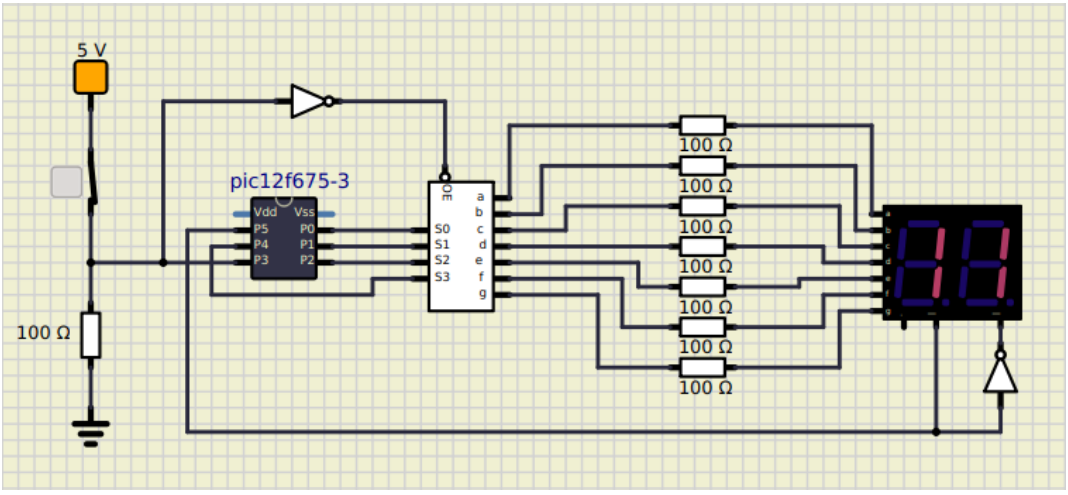


Figura 16: Cuarto número mostrado

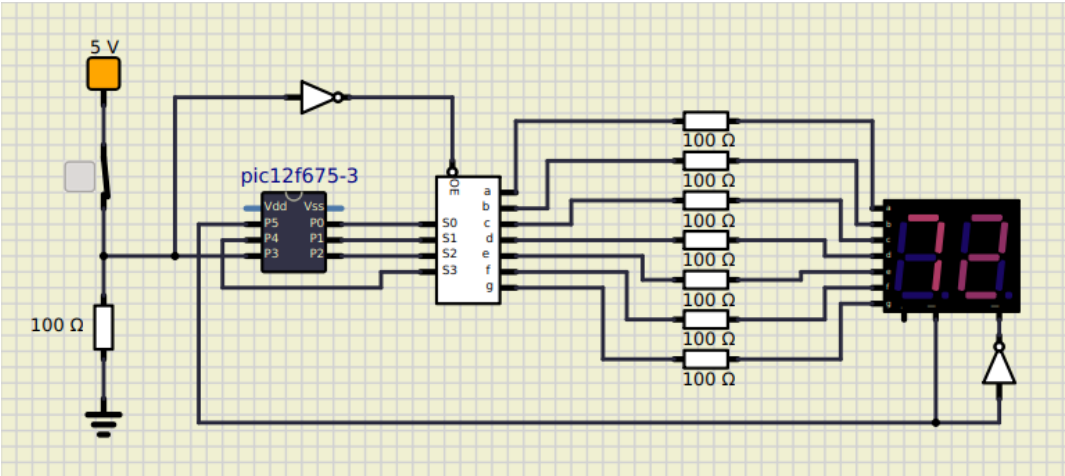


Figura 17: Quinto número mostrado

4. Conclusiones y recomendaciones

- Utilizando otro modelo de PIC con mayor número de GPIOs se pueden controlar 2 display por separado lo cual simplificaría el proceso para obtener 2 números distintos en cada display.
- Se le puede adaptar al circuito otro modelo de PIC que cuente con mayor memoria. Con esto se podrían hacer funciones como la de guardar los números que van saliendo de la tómbola para llevar un registro y que no se repitan.
- Para obtener los números del 0 al 80 se utilizó una estructura de código a base de switch y cases pero puede haber otra manera menos extensa de poder implementarlo para alcanzar los 99 números y que el PIC no muestre un error.
- Se puede disminuir aún más los tiempos de delay para que el recorrido del 00 al 80 se realice de forma más rápida.

Referencias

[1] Villalta, M. (2022). **GPIOs y el PIC12f629**. IE-0624 Laboratorio de Microcontroladores. Escuela de Ingeniería Eléctrica.

[2] Microchip. PIC12F675. Página web: `https://www.microchip.com/en-us/product/PIC12F675#buy-from-store`

[3] Mouser. PIC12F675-I/P. Página web: `https://www.mouser.co.cr/ProductDetail/Microchip-Technology-Atmel/PIC12F675-I-P?qs=g6BBfX6YTklkFbwYwZUX2A%3D%3D`

[4] Texas Instruments. Compuerta NOT SN74LVC1G04DBVR. `https://www.ti.com/product/SN74LVC1G04/part-details/SN74LVC1G04DBVR`

[5] Teltron (Costa Rica), S.A. Catálogo: `https://teltroncr.com/index.php?page=products.php#close`

[6] Unit Electronics. 74LS04 Compuerta NOT SN74LS04N. Página web: https://uelectronics.com/producto/74ls04-compuerta-not-sn74ls04n/#:~:text=La%20compuerta%20NOT%20es%20una,l%C3%B3gico%20alto%20y%20l%C3%B3gico%20bajo.

[7] Villa, J. (2020). Decodificadores de BCD a 7 segmentos. Página web: https://wp.7robot.net/decodificadores-de-bcd-a-7-segmentos/.

5. Anexos

5.1. GITHUB

El control de versiones o avances del programa del laboratorio se encuentra en el siguiente repositorio público de GITHUB:

- HTTPS: https://github.com/Gabo4x/Laboratorio-Microcontroladores.git
- SSH: git@github.com:Gabo4x/Laboratorio-Microcontroladores.git

5.2. Código del laboratorio

```
//Trabajo realizado por: Gabriel Barahona Otoya B70896
//Laboratorio de Microcontroladores IE-0624
//Laboratorio #1

#include <pic14/pic12f675.h>

void delay (unsigned int tiempo);

typedef unsigned int word;
word __at 0x2007 __CONFIG = (_WDT_OFF & _MCLRE_OFF);

int n;
int count;
int num1;
int num2;

void main(void)
{
        unsigned int time = 10;
        unsigned int time2 = 25;

    TRISIO = 0x08;

        if(GP3 == 0)
        {
                switch (n)
                {
                case 0:
                        num1 = 0x00;
                        num2 = 0x20;
                        break;
                case 1:
```

```
            num1 = 0x00;
            num2 = 0x21;
            break;
        case 2:
            num1 = 0x00;
            num2 = 0x22;
            break;
        case 3:
            num1 = 0x00;
            num2 = 0x23;
            break;
        case 4:
            num1 = 0x00;
            num2 = 0x24;
            break;
        case 5:
            num1 = 0x00;
            num2 = 0x25;
            break;
        case 6:
            num1 = 0x00;
            num2 = 0x26;
            break;
        case 7:
            num1 = 0x00;
            num2 = 0x27;
            break;
        case 8:
            num1 = 0x00;
            num2 = 0x30;
            break;
        case 9:
            num1 = 0x00;
            num2 = 0x31;
            break;
        case 10:
            num1 = 0x01;
            num2 = 0x20;
            break;
        case 11:
            num1 = 0x01;
            num2 = 0x21;
            break;
        case 12:
            num1 = 0x01;
            num2 = 0x22;
            break;
        case 13:
            num1 = 0x01;
            num2 = 0x23;
            break;
        case 14:
            num1 = 0x01;
            num2 = 0x24;
            break;
        case 15:
```

```
            num1 = 0x01;
            num2 = 0x25;
            break;
        case 16:
            num1 = 0x01;
            num2 = 0x26;
            break;
        case 17:
            num1 = 0x01;
            num2 = 0x27;
            break;
        case 18:
            num1 = 0x01;
            num2 = 0x30;
            break;
        case 19:
            num1 = 0x01;
            num2 = 0x31;
            break;
        case 20:
            num1 = 0x02;
            num2 = 0x20;
            break;
        case 21:
            num1 = 0x02;
            num2 = 0x21;
            break;
        case 22:
            num1 = 0x02;
            num2 = 0x22;
            break;
        case 23:
            num1 = 0x02;
            num2 = 0x23;
            break;
        case 24:
            num1 = 0x02;
            num2 = 0x24;
            break;
        case 25:
            num1 = 0x02;
            num2 = 0x25;
            break;
        case 26:
            num1 = 0x02;
            num2 = 0x26;
            break;
        case 27:
            num1 = 0x02;
            num2 = 0x27;
            break;
        case 28:
            num1 = 0x02;
            num2 = 0x30;
            break;
        case 29:
```

```
            num1 = 0x02;
            num2 = 0x31;
            break;
        case 30:
            num1 = 0x03;
            num2 = 0x20;
            break;
        case 31:
            num1 = 0x03;
            num2 = 0x21;
            break;
        case 32:
            num1 = 0x03;
            num2 = 0x22;
            break;
        case 33:
            num1 = 0x03;
            num2 = 0x23;
            break;
        case 34:
            num1 = 0x03;
            num2 = 0x24;
            break;
        case 35:
            num1 = 0x03;
            num2 = 0x25;
            break;
        case 36:
            num1 = 0x03;
            num2 = 0x26;
            break;
        case 37:
            num1 = 0x03;
            num2 = 0x27;
            break;
        case 38:
            num1 = 0x03;
            num2 = 0x30;
            break;
        case 39:
            num1 = 0x03;
            num2 = 0x31;
            break;
        case 40:
            num1 = 0x04;
            num2 = 0x20;
            break;
        case 41:
            num1 = 0x04;
            num2 = 0x21;
            break;
        case 42:
            num1 = 0x04;
            num2 = 0x22;
            break;
        case 43:
```

```
            num1 = 0x04;
            num2 = 0x23;
            break;
        case 44:
            num1 = 0x04;
            num2 = 0x24;
            break;
        case 45:
            num1 = 0x04;
            num2 = 0x25;
            break;
        case 46:
            num1 = 0x04;
            num2 = 0x26;
            break;
        case 47:
            num1 = 0x04;
            num2 = 0x27;
            break;
        case 48:
            num1 = 0x04;
            num2 = 0x30;
            break;
        case 49:
            num1 = 0x04;
            num2 = 0x31;
            break;
        case 50:
            num1 = 0x05;
            num2 = 0x20;
            break;
        case 51:
            num1 = 0x05;
            num2 = 0x21;
            break;
        case 52:
            num1 = 0x05;
            num2 = 0x22;
            break;
        case 53:
            num1 = 0x05;
            num2 = 0x23;
            break;
        case 54:
            num1 = 0x05;
            num2 = 0x24;
            break;
        case 55:
            num1 = 0x05;
            num2 = 0x25;
            break;
        case 56:
            num1 = 0x05;
            num2 = 0x26;
            break;
        case 57:
```

```
            num1 = 0x05;
            num2 = 0x27;
            break;
        case 58:
            num1 = 0x05;
            num2 = 0x30;
            break;
        case 59:
            num1 = 0x05;
            num2 = 0x31;
            break;
        case 60:
            num1 = 0x06;
            num2 = 0x20;
            break;
        case 61:
            num1 = 0x06;
            num2 = 0x21;
            break;
        case 62:
            num1 = 0x06;
            num2 = 0x22;
            break;
        case 63:
            num1 = 0x06;
            num2 = 0x23;
            break;
        case 64:
            num1 = 0x06;
            num2 = 0x24;
            break;
        case 65:
            num1 = 0x06;
            num2 = 0x25;
            break;
        case 66:
            num1 = 0x06;
            num2 = 0x26;
            break;
        case 67:
            num1 = 0x06;
            num2 = 0x27;
            break;
        case 68:
            num1 = 0x06;
            num2 = 0x30;
            break;
        case 69:
            num1 = 0x06;
            num2 = 0x31;
            break;
        case 70:
            num1 = 0x07;
            num2 = 0x20;
            break;
        case 71:
```

```
                num1 = 0x07;
                num2 = 0x21;
                break;
        case 72:
                num1 = 0x07;
                num2 = 0x22;
                break;
        case 73:
                num1 = 0x07;
                num2 = 0x23;
                break;
        case 74:
                num1 = 0x07;
                num2 = 0x24;
                break;
        case 75:
                num1 = 0x07;
                num2 = 0x25;
                break;
        case 76:
                num1 = 0x07;
                num2 = 0x26;
                break;
        case 77:
                num1 = 0x07;
                num2 = 0x27;
                break;
        case 78:
                num1 = 0x07;
                num2 = 0x30;
                break;
        case 79:
                num1 = 0x07;
                num2 = 0x31;
                break;
        case 80:
                num1 = 0x10;
                num2 = 0x20;
                break;
        default:
                n = 0;
        }
        if(n>80)
        {
                n = 0;
                delay(time2);
        }else if(n<=80)
        {
                n = n+1;
                delay(time2);
        }
}
if(GP3 == 1)
{
        if(count<=16)
        {
```

```
                        GPIO = num1;
                        delay(time);
                        GPIO = num2;
                        delay(time);
                }
                if(count>16)
                {
                        GPIO = 0x11;
                        delay(time);
                        GPIO = 0x31;
                        delay(time);
                        count = 0;
                }
        }
        count = count++;
}

void delay(unsigned int tiempo)
{
        unsigned int i;
        unsigned int j;
        for(i=0;i<tiempo;i++)
          for(j=0;j<1275;j++);
}
```