Universidad de Costa Rica

Escuela de Ingeniería Eléctrica

IE-0624: Laboratorio de Microcontroladores

Reporte de laboratorio #1: Introducción a microcontroladores y manejo de GPIOS

Jose Mario Navarro Bejarano

B75398

II-2024

Contenido

Introducción

Los microcontroladores son dispositivos con muchas funcionalidades que permiten realizar muchas tareas de forma muy eficiente. Son dispositivos versátiles y de bajo costo que se pueden encargar de realizar muchas funciones dependiendo de en dónde sean implementados. En este laboratorio se realizó la implementación de un pequeño juego de Bingo, en el cual el PIC12F683 permite generar números pseudoaleatorios que serán mostrados en un par de display de 10 segmentos mediante la ayuda de lógica combinacional. Para la realización de esto fue necesario manipular los registros del microcontrolador y realizar un proceso de análisis y diseño del codigo utilizado para implementar la solución. A continuación, se muestra el proceso seguido, los elementos utilizados y el resultado obtenido. El repositorio con los archivos base se puede encontrar en: https://github.com/marionabe/Lab_micros

Nota teórica

Información General del PIC12F683

El PIC12F683 es un microcontrolador de 8 bits que utiliza instrucciones RISC y posee una frecuencia de operación de 20 MHz. Posee seis terminales GPIO y dos terminales para alimentación. En la figura 1 se muestra un diagrama de la función de cada terminal.

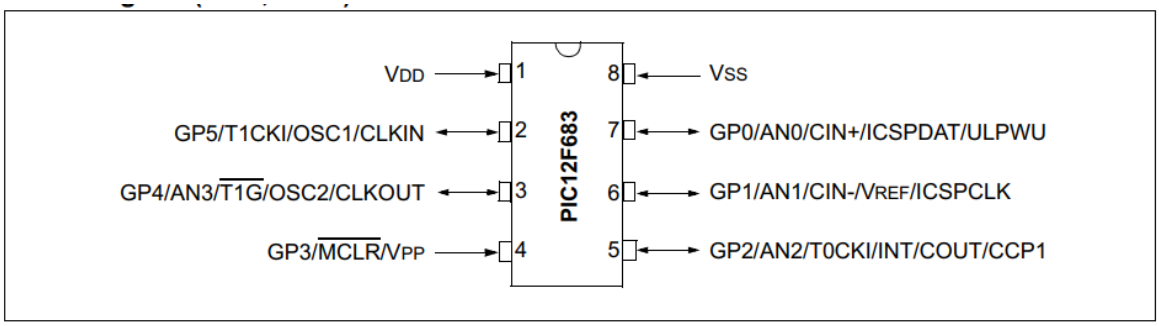

Figura 1: Diagrama de función de cada terminal. Obtenido de la hoja de datos del fabricante.

Es importante tener claras las características eléctricas de este dispositivo. En la figura 2 se muestra una tabla con las principales especificaciones para el control de energía de sus terminales.

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

Figura 2: Especificaciones eléctricas del PIC12F683. Obtenidas de la hoja de datos del fabricante.

Por otra parte, en la figura 3 se muestra un diagrama de bloques que describe, de forma general, la construcción interna de este microcontrolador.

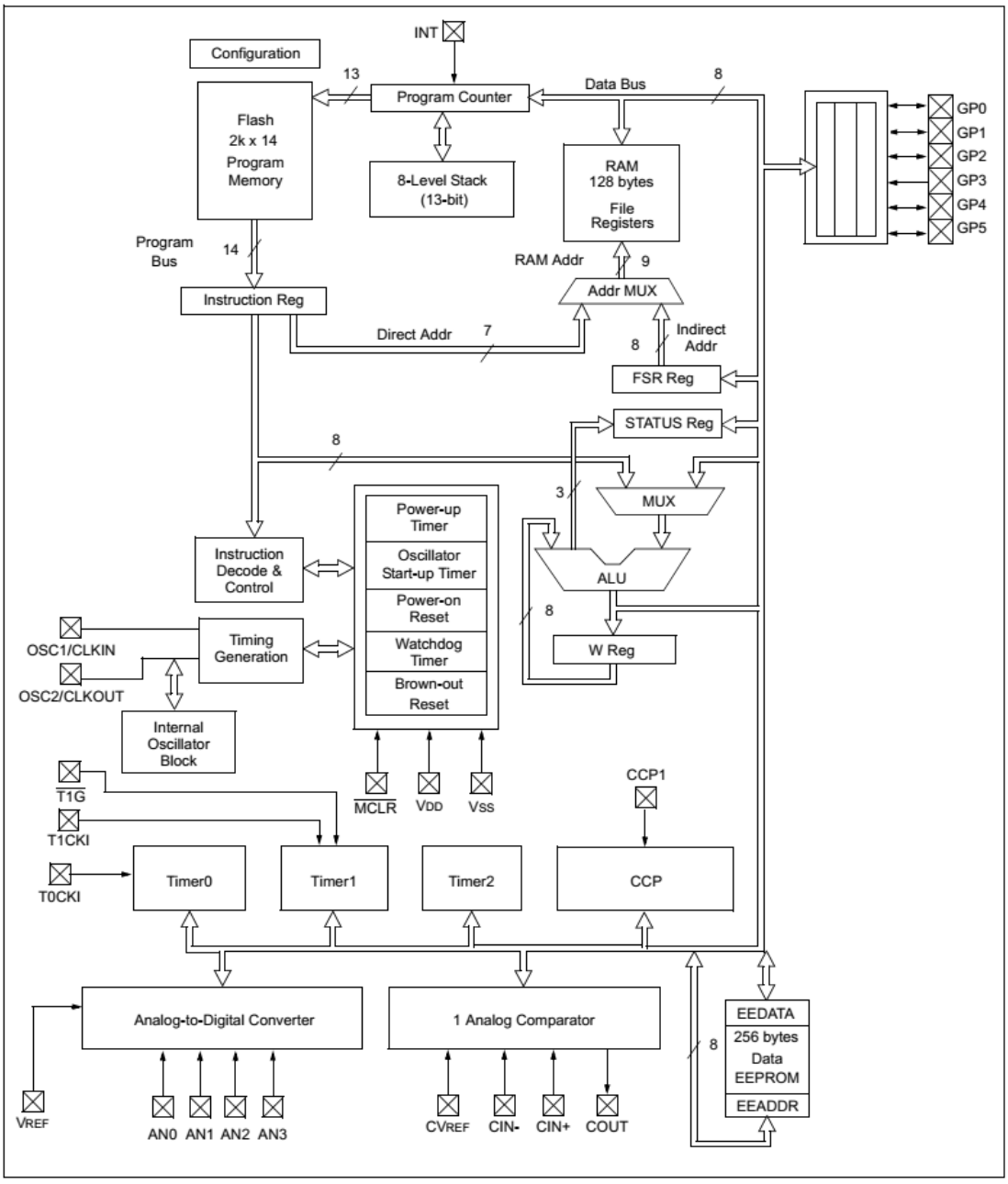

Figura 3: Diagrama de construcción del microcontrolador. Obtenida de la hoja de datos del fabricante.

Este microcontrolador posee, entre otros, varios registros que permiten configurar su modo de funcionamiento, y que además permite decidir cómo manipular cada uno de sus registros de propósito general. A continuación, se dá una explicación de los registros que fueron utilizados para realizar este laboratorio.

Registro Watchdog Timer:

Este registro posee 4 bits dedicados a configurar el tiempo que el registro WD esperará antes de aplicar un reset al programa. Posee además un bit que permite activarlo o desactivarlo mediante código durante la ejecución.

U-0	U-0	U-0	R/W-0	R/W-1	R/W-0	R/W-0	R/W-0
—	—	—	WDTPS3	WDTPS2	WDTPS1	WDTPS0	SWDTEN
bit 7							bit 0

Figura 4: Bits del registro WD Timer. Obtenido de la hoja de datos del fabricante.

Registro TRISIO:

El registro TRISIO se utiliza para manipular los buffers triestado que se ubican en las terminales GPIO, esto permite configurarlas como entradas o como salidas dependiendo del uso que se desee. Cada bit de este registro modifica una de las terminales GPIO.

U-0	U-0	R/W-1	R/W-1	R-1	R/W-1	R/W-1	R/W-1
—	—	TRISIO5(2,3)	TRISIO4(2)	TRISIO3(1)	TRISIO2	TRISIO1	TRISIO0
bit 7							bit 0

Figura 5: Bits del registro TRISIO. Obtenido de la hoja de datos del fabricante.

Registro GPIO:

El tercer registro que fue utilizado para la realización de este laboratorio es el registro GPIO. Este registro contiene los valores digitales detectados en cada una de sus terminales, si se configuran como entradas. O contiene el valor que se desea generar en sus salidas si estas fueron configuradas como salidas.

U-0	U-0	R/W-x	R/W-0	R-x	R/W-0	R/W-0	R/W-0
—	—	GP5	GP4	GP3	GP2	GP1	GP0
bit 7							bit 0

Figura 6: Bits del registro GPIO. Obtenido de la hoja de datos del fabricante.

Otros elementos utilizados

Para la realización de este laboratorio también fue necesario hacer uso de compuertas lógicas AND, OR y NOT. En las tablas 1, 2 y 3 se detalla el funcionamiento de cada una de estas compuertas.

A	B	AND
0	0	0
0	1	0
1	0	0
1	1	1

Tabla 1: Funcionamiento de la compuerta lógica AND con base en dos señales de entrada A y B. Fuente: Creación propia.

A	B	OR
0	0	0
0	1	1
1	0	1
1	1	1

Tabla 2: Funcionamiento de la compuerta lógica OR con base en dos señales de entrada A y B. Fuente: Creación propia.

A	NOT
0	1
1	0

Tabla 3: Funcionamiento de la compuerta lógica NOT con base en una señal de entrada A Fuente: Creación propia.

También fueron utilizados dos display de 7 segmentos. Estos dispositivos, como su nombre lo indica, poseen 7 Leds que se pueden encender o apagar independientemente para poder mostrar diferentes números. En la figura 7 se muestra una captura de pantalla de los display utilizados. Es importante mencionar que estos dispositivos, al ser formados únicamente de Leds, se les debe agregar una resistencia, para limitar la corriente que les llega y evitar dañarlos. Como se observa en la siguiente figura, cada segmento a-g posee su propia terminal de alimentación, así es como se puede controlar cada uno de forma independiente.

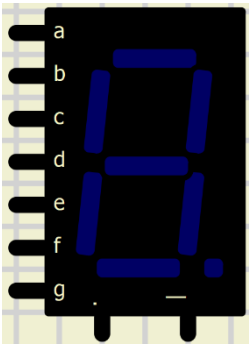

Figura 7: Display de 7 segmentos utilizados. Obtenido de SimulIDE.

Desarrollo

Diagrama de Flujo

Para la resolución de este ejercicio se comienza realizando el siguiente diagrama de flujo que permite resolver el problema. Es importante mencionar que a la hora de realizar el código se realizaron algunas cosas que no se muestran en el diagrama, como el método utilizado para generar los numero aleatorio.

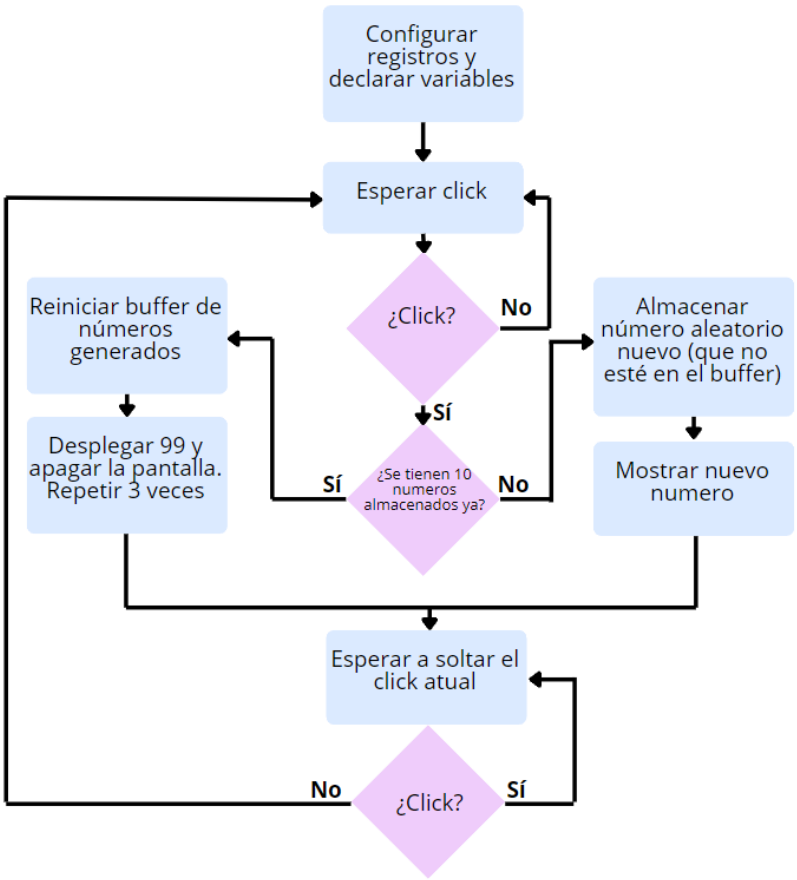

Figura 8: Diagrama de flujo utilizado para la realización del laboratorio 1. Fuente: Creación propia.

Código utilizado

A continuación, se muestran algunas capturas de pantalla del código utilizado para mostrar la forma en la que se implementó el diagrama de flujo anterior y la forma en la que fue resuelto el problema propuesto por el laboratorio. Cabe mencionar que solo se muestran las secciones del codigo que se consideran más importantes, para evitar saturar este reporte.

```
17      TRISIO = 0b00100000;
18      GPIO=0b000000;
19      WDTPS0=0b1;
20      WDTPS1=0b0;
21      WDTPS2=0b1;
22      WDTPS3=0b1;
```

Figura 9: Sección del codigo utilizado para configurar los registros.

En la imagen anterior se muestra la sección del codigo utilizada para configurar los registros necesarios. En la linea 17 se configuran los buffer triestado como salidas y el pin 5 como entrada para recibir la señal del clic. En las líneas 19-22 se configuran los bits del WD timer para aumentar la cantidad de ciclos de espera y evitar que se reinicie durante la ejecución de algún ciclo.

```
26      //Esperar al click
27      while (click_dtc == 0){
28          //Este contador se aumenta cada click y se reinicia cuando llega a 10
29          //Se utiliza como numero pseudo-aleatorio
30          counter++;
31          if (counter==10){
32              counter=0;
33          }
34          click_dtc = GP5;
35      }
```

Figura 10: Sección del codigo utilizado para esperar a un click.

En la figura anterior se muestra el código que se utilizó para esperar al click. Esto se realiza igualando una variable al bit 5 del registro GPIO. En las líneas 30-33 se muestra la parte del codigo que mantiene un contador aumentando hasta 10 y reiniciándose a cero, esto se utiliza como un generador pseudoaleatorio. Aunque lo ideal sería utilizar la función “rand()” de las librerías estándar de C, se decidió realizarlo de esta forma debido a que se estuvieron generando problemas a la hora de acceder a estas librerías, por lo que, para evitar mayores retrasos, se decidió implementar la función de esta forma.

```
37      if (indice == 10){  //Entrar si ya se tienen 10 numero almacenados
38          indice = 0;
39          for (i=0; i<10; i++){
40              reg_num_gen[i]=70;
41          }
42          GPIO = 0b011111;
43          delay(time);
44          mostrar_numero(99);
45          delay(time);
46          GPIO = 0b011111;
47          delay(time);
48          mostrar_numero(99);
49          delay(time);
50          GPIO = 0b011111;
51          delay(time);
52          mostrar_numero(99);
53          delay(time);
54          GPIO = 0b011111;
```

Figura 11: Sección del codigo utilizada para reiniciar el buffer de números almacenados.

En la figura anterior se muestra la sección del codigo que se ejecuta cuando se hace click pero ya se cuentan con los diez números aleatorios generados. Como se observa en el codigo lo que se hace es reiniciar el índice utilizado, reiniciar el buffer, y apagar y encender la pantalla para mostrar el numero 99 3 veces repetidas. Para apagar la pantalla se realiza un NAND entre las 4 terminales GPIO utilizadas del microcontrolador, por lo que cuando todas las terminales se ponen en 1, se corta el suministro a los segmentos de los display.

```
55      }else{ //Si no hay 10 numeros generados, guardar el nuevo y comprobar que no se tiene almacenado ya
56          do{
57              num_inv=0;
58              for (i=0; i<indice+1; i++){
59                  if (counter==reg_num_gen[i]){
60                      num_inv=1;
61                  }
62              }
63              //Si ya se tiene almacenado, aumentar en 1 y volver a comprobar que no se tiene almacenado ese otro numero
64              if (num_inv==1){
65                  counter++;
66              }
67              if (counter==10){
68                  counter=0;
69              }
70          }while (num_inv==1);
71          mostrar_numero(counter);
72          reg_num_gen[indice] = counter;
73          indice++;
```

Figura 12: Sección del codigo que se ejecuta para almacenar un numero aleatorio nuevo.

En la figura anterior se muestra la sección del codigo que se ejecuta cuando se recibe un click y se desea almacenar un nuevo número aleatorio. En este caso en las líneas 58-61 se recorre el buffer para comprobar si el número que se desea agregar no existe ya almacenado. En caso de que ya se encuentre almacenado se pone en 1 la variable num_inv indicando que el número es invalido, por lo que se debe repetir el ciclo para comprobar si el siguiente nuevo número sí es válido. En las líneas 64-66 se aumenta el contador en 1 en caso de que se haya detectado

el numero invalido. En las líneas 67 se reinicia el contador si ha llegado a un número mayor a 9, lo cual se sale de los requerimientos del problema planteado. Finalmente, en las líneas 71-73 se llama a la función mostrar_numero para activar los display con el numero que se agregó al buffer, tambien se almacena este en el buffer y se aumenta el índice, para apuntar al siguiente espacio dentro del buffer para el siguiente número generado.

```
76          while (click_dtc == 1){
77              click_dtc = GP5;
78          }
```

Figura 13: Bucle para esperar a soltar el click

En la figura anterior se muestra un pequeño bucle que se implementó para esperar a que se suelte el botón. Este bucle se implementó porque se encontró el problema que un click podría durar más tiempo que el tiempo que le toma al microcontrolador realizar todas las instrucciones, por lo que se detectaba como 2 click, lo cual es indeseable. Esto permite que el flujo se continúe hasta que se haya levantado el botón.

```
82  void mostrar_numero(short int num){
83      switch (num){
84          case 0: GPIO = 0b000000; break;
85          case 1: GPIO = 0b000001; break;
86          case 2: GPIO = 0b000010; break;
87          case 3: GPIO = 0b000011; break;
88          case 4: GPIO = 0b000100; break;
89          case 5: GPIO = 0b000101; break;
90          case 6: GPIO = 0b000110; break;
91          case 7: GPIO = 0b000111; break;
92          case 8: GPIO = 0b010000; break;
93          case 9: GPIO = 0b010001; break;
94          case 99: GPIO = 0b010010; break;
95          default: GPIO = 0b000000;
96      }
97  }
```

Figura 14: Función utilizada para desplegar los números.

En la figura anterior se muestra la función mostrar_numero. Como su nombre lo indica, esta función se utiliza para controlar cada una de las salidas del microcontrolador con base en los numero que se desea mostrar. Cada número se despliega con base en la tabla de verdad mostrada en la tabla 4. Más adelante se explicará la lógica utilizada para poder generar cada número con base en estas salidas.

GP4	GP3	GP2	GP1	GP0	Numero mostrado
0	0	0	0	0	0
0	0	0	0	1	1
0	0	0	1	0	2
0	0	0	1	1	3
0	0	1	0	0	4
0	0	1	0	1	5
0	0	1	1	0	6
0	0	1	1	1	7
1	0	0	0	0	8
1	0	0	0	1	9
1	0	0	1	0	99
1	0	1	1	1	Apagar display

Tabla 4: Tabla de verdad de cada número mostrado en función de las salidas GPIO.

Circuito implementado:

En la figura 15 se muestra el circuito resultante implementado. Más adelante se dará una explicación del funcionamiento de cada sección marcada.

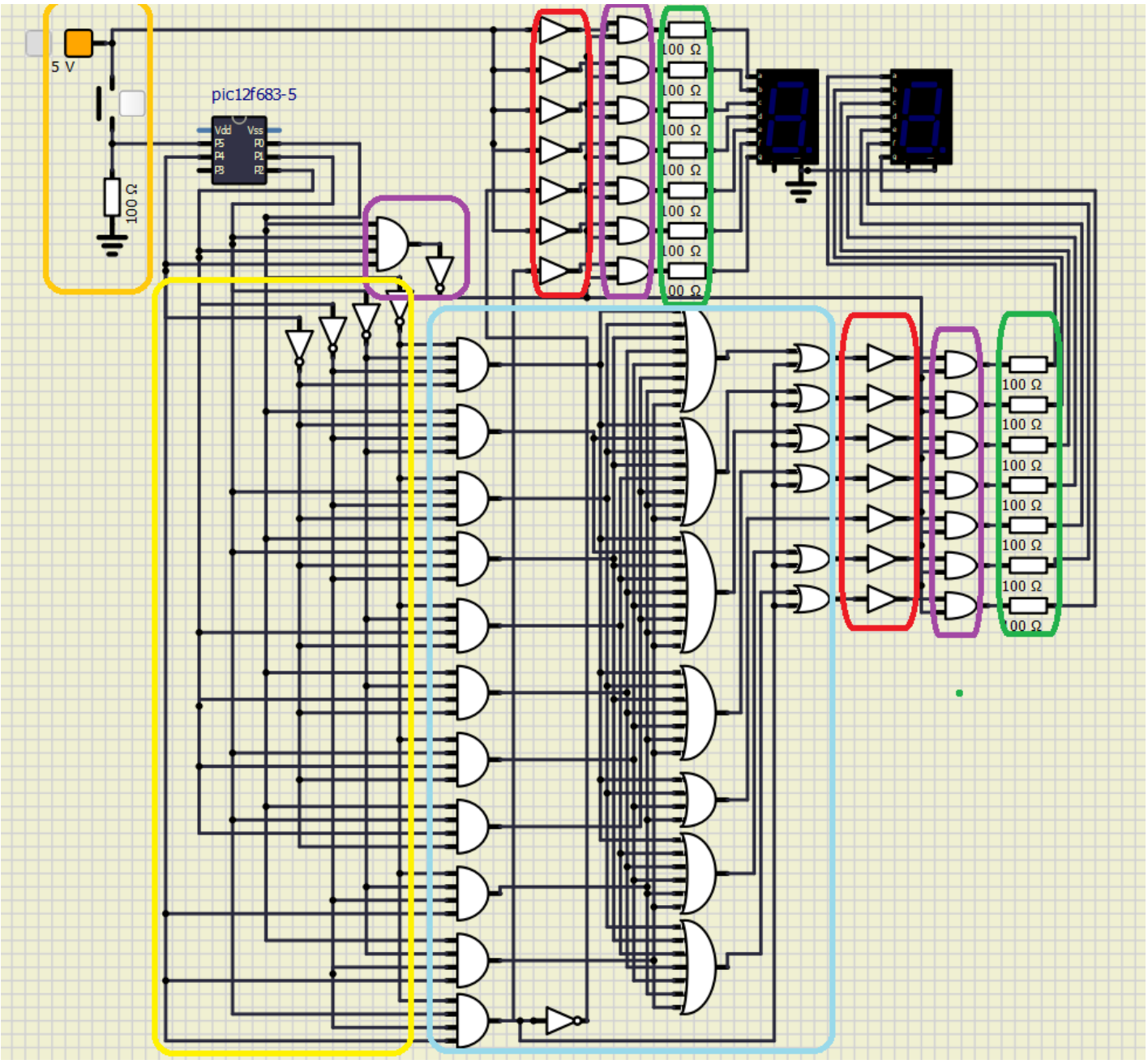

Figura 15: Circuito implementado. Fuente: Creación propia

Descripción de cada sección señalada:

- Sección verde: Estas dos secciones corresponden a un grupo de 14 resistencias de 100 Ohm utilizadas para regular la corriente que entra a cada led.

- Sección roja: Esta corresponde a un grupo de buffers utilizados para regenerar la señal y permitir una alimentación estable para cada led.
- Sección amarilla: Esta sección contiene las líneas de las señales de cada GPIO, incluyendo las señales invertidas.
- Sección morada: Esta sección corresponde a una compuerta AND y un NOR que se encarga de activarse cuando se activan todas las salidas GPIO. y esta señal invertida se pasa por un AND con la señal de cada segmente. Este funcionamiento permite apagar los leds cuando se activen todas las señales GPIO.
- Sección anaranjada: Esta sección permite leer los click desde el botón.
- Sección celeste: Esta sección contiene la lógica que se utiliza para controlar los segmentos de los display. Fue creada de forma experimental, es decir, realizando conexiones hasta lograr el comportamiento deseado para controlar cada segmento.

A continuación, se van a mostrar unas imágenes que permiten observar los display en funcionamiento:

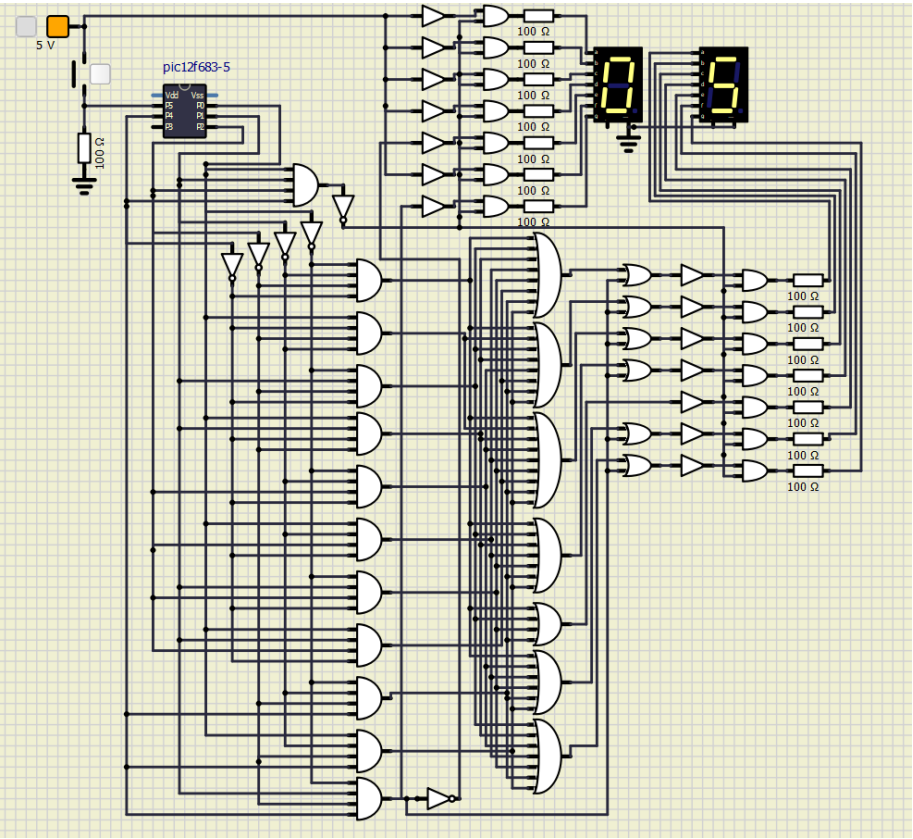

Figura 16: Display mostrando el número 05.

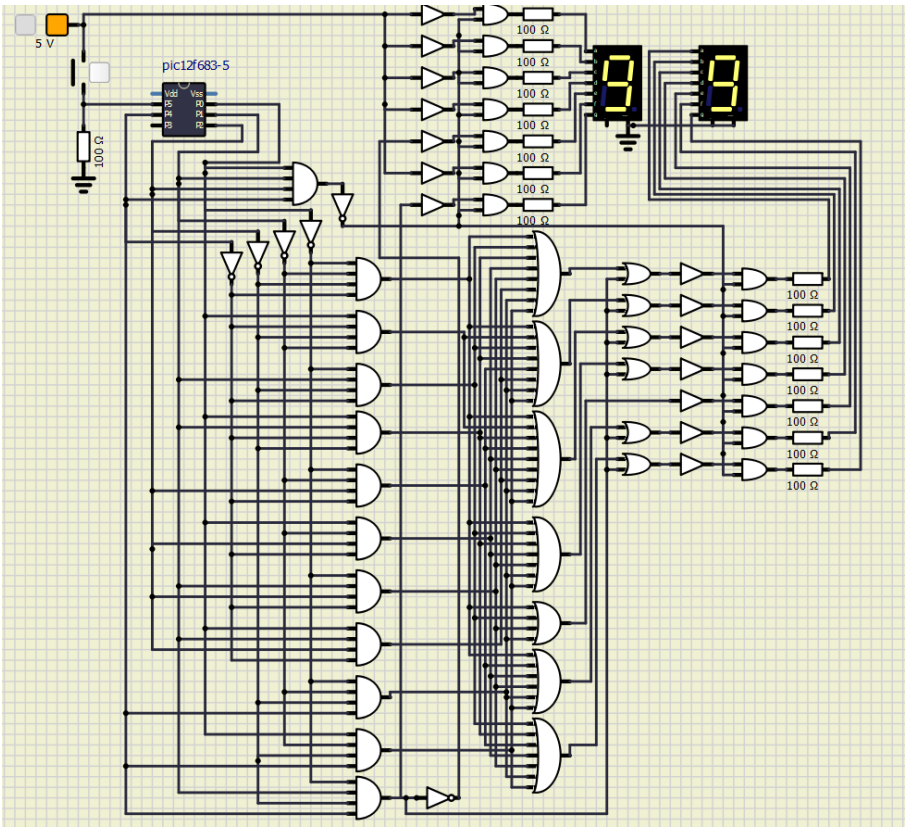

Figura 17: Cuando se generan los 10 números en total, la pantalla muestra el número 99 y se apaga, repitiendo esta secuencia 3 veces.

Conclusiones y recomendaciones

- Se logró manipular los puertos GPIO para mostrar valores en 0 o en 1 lógicos, mediante la manipulación de los registros correspondientes.
- Se logró controlar de forma exitosa los display para mostrar el número generado.
- Se logró crear un programa y ejecutar en el microprocesador PIC12F683 de manera exitosa.

A modo de recomendaciones:

- Sería recomendable buscar una forma más optima en términos de tamaño para controlar los display, ya que el circuito implementado pareciera ser muy complejo.
- Sumado a lo anterior, sería bueno volver a realizar el circuito que controla los leds, pero desde un punto de vista de análisis y optimización, para evitar el uso de compuertas innecesarias, esto en lugar de la forma experimental en la que fue realizado para este laboratorio.

Bibliografía

- Microchip. “PIC12F683 Data Sheet”. 2007. Obtenido de: https://ww1.microchip.com/downloads/en/devicedoc/41211d_.pdf