Universidad de Costa Rica
Escuela de Ingeniería Eléctrica
IE0624 - Laboratorio de Microcontroladores

Laboratorio #1
Introducción a microcontroladores y manejo de GPIOS

Jafet David Gutiérrez Guevara, B73558

4 de septiembre del 2022

1. Introducción/Resumen

Para esta práctica de laboratorio se desarrolló una especie de juego de bingo electrónico, el cual por medio de un microcontrolador PIC12f675 genera un números pseudo aleatorio entre 0 y 99. Para construir el circuito solicitado, aparte del microcontrolador, también se utilizó un pulsador, un decodificador BCD, dos displays de 7 segmentos y un filtro pasivo hecho con una resistencia y un capacitor. Dicho filtro se agregó con el fin de eliminar el factor del "bouncing" del botón. Cuando un usuario presiona el pulsador se simula que se saca un número del bingo, por lo que el microcontrolador genera un número en formato BCD y lo pasa al decodificador a través de sus salidas. El decodificador inmediatamente enciende los segmentos de los displays correspondientes al valor BCD recibido, desplegando un número de dos dígitos.

Este microcontrolador es muy básico y cuenta con poca memoria, por lo que no fue posible implementar un algoritmo muy complejo para generar los números aleatorios y tampoco ofrece la posibilidad de guardar en memoria los dígitos que produce. Por otra parte, se tuvo que resolver la limitación de hardware que representaba tener únicamente 6 GPIOs disponibles para el pulsador y para pasar dos números decimales desde el microcontrolador hasta los dos displays de 7 segmentos. No obstante, se encontró una solución a este problema implementando la técnica de multiplexación en las salidas de los GPIOs.

2. Nota teórica

2.1. PIC12f675

El PIC12f675 es un sencillo, pero potente, microcontrolador que cuenta con [1]:

- Flash de 8 bits contenido en 8 pines.
- 4 canales para un conversor análogo-digital de 10-bits
- Un comparador análogo
- Memoria EEPROM de 128 Bytes
- Reloj interno de hasta $4\,MHz$
- Procesador de tipo RISC (Reduced Instruction Set Computer)
- Consumo de corriente de $100\,\mu A$

Aparte de esto el PIC12f675 cuenta con 6 pines que pueden operar como entradas/salidas programables (GPIOs), sin embargo cabe aclarar que el GPIO3 se puede utilizar únicamente como entrada. En la figura 1 se muestra un esquema con las 8 entradas y salidas que posee el PIC12F675 [1].

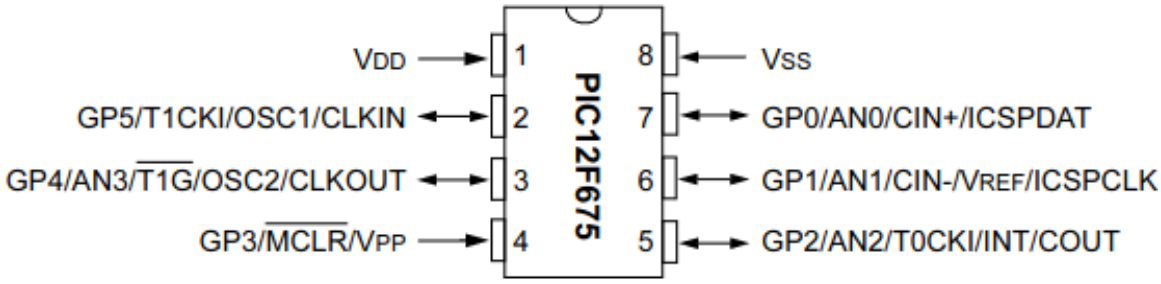

Figura 1: Esquema general del PIC12F675 [1]

Por otro lado, la memoria de este microcontrolador se encuentra partida en dos bancos, de los cuales una partición contiene los registros de propósito general y la otra posee los registros especiales. Además, el PIC12f675 tiene un contador de programa (PC) de 13 bits [1].

En cuanto a las características eléctricas del PIC12f675, se puede mencionar que cada uno de sus pines es capaz de entregar $25\,mA$. Sin embargo, todos los pines juntos tiene la capacidad de entregar un total de $125\,mA$ [1].

2.1.1. Registro TRISIO

Este registro sirve para establecer la configuración de entradas/salidas del microcontrolador. En palabras simples, a través de este registro se configura cuales periféricos van a operar como entradas y cuales van a servir como salidas. Como se puede observar en la figura 2, TRISIO es un registro de 8 bits. Sin embargo, cabe mencionar que solo los 6 bits menos significativos de dicho registro son útiles para el programador [1].

TRISIO	—	—	TRISIO5	TRISIO4	TRISIO3	TRISIO2	TRISIO1	TRISIO0	--11 1111

Figura 2: Registro TRISIO [1]

A continuación se muestra un ejemplo de la configuración del registro TRISIO:

```
TRISIO = 0b00101000; // Se configura al GPIO3 y al GPIO5 como entradas y el resto como sali
```

2.1.2. Registro GPIO

Este registro se utiliza para configurar el estado predeterminado de los GPIOs, el cual puede ser alto ($5\,V$) o bajo ($0\,V$). De la misma manera que el registro TRISIO, el GPIO posee 8 bits, pero solo los 6 menos significativos son útiles para el programador, como se puede observar en la figura 3 [1].

GPIO	—	—	GPIO5	GPIO4	GPIO3	GPIO2	GPIO1	GPIO0

Figura 3: Registro GPIO [1]

A continuación se muestra un ejemplo de la configuración del registro GPIO:

```
GPIO = 0; // Se configura el GPIO0 como salida en estado bajo por defecto
GPI1 = 0; // Se configura el GPIO1 como salida en estado bajo por defecto
GPI2 = 0; // Se configura el GPIO2 como salida en estado bajo por defecto
GPI3 = 0; // Se configura el GPIO3 como salida en estado bajo por defecto
GPI4 = 1; // Se configura el GPIO4 como salida en estado alto por defecto
GPI5 = 0; // Se configura el GPIO5 como salida en estado bajo por defecto
```

Sin embargo, la forma más común para realizar esta configuración es la siguiente, ya que por lo general se preestablecen todos estos periféricos para que funcionen virtualmente como tierras:

```
GPIO = 0; // Se preestablecen todos los pines como salidas en estado bajo
```

2.1.3. Registro ANSEL

Con este registro se puede controlar el conversor analógico/digital del microcontrolador, sin embargo no se empleó para este laboratorio. Así que se dejó apagado (en cero).

ANSEL	—	ADCS2	ADCS1	ADCS0	ANS3	ANS2	ANS1	ANS0

Figura 4: Registro ANSEL [1]

2.1.4. Registro CMCON

Este registro sirve para controlar el comparador que se encuentra dentro del microcontrolador. No obstante, al igual que el registro ANSEL este no se empleó en la solución del laboratorio.

CMCON	—	COUT	—	CINV	CIS	CM2	CM1	CM0

Figura 5: Registro CMCON [1]

2.2. Dimensionamiento de las resistencias de protección de los displays

Los displays utilizados para este laboratorio contienen una serie de LEDs y, por lo tanto, es necesario conectar a cada una de sus terminales una resistencia que proteja dichos LEDs de sobrecorrientes. Ya se sabe que la corriente máxima que ofrece cada perifério del PIC12f675 es de $25\,mA$, pero en este caso para el cálculo de las resistencias de protección se considerará que cada pin entregará menos de $20\,mA$ en todo momento. Además, tomando en cuenta que un 1 lógico en un GPIO es equivalente a una salida de $5\,V$ se tiene que:

$$R_{proteccion} = \frac{V}{I} = \frac{5\,V}{20\,mA} \approx 250\,\Omega \tag{1}$$

2.3. Rebote del pulsador

Cuando se presiona un pulsador, la salida del mismo no es una señal totalmente limpia, sino que presenta rebotes. Esto se debe a que los contactos del pulsador suelen presentar un pequeño rebote entre ellos. Esto representa un inconveniente ya que los rebotes pueden introducir falsas lecturas al sistema. Para contrarrestar este efecto en el circuito diseñado, se agregó el siguiente filtro pasivo:

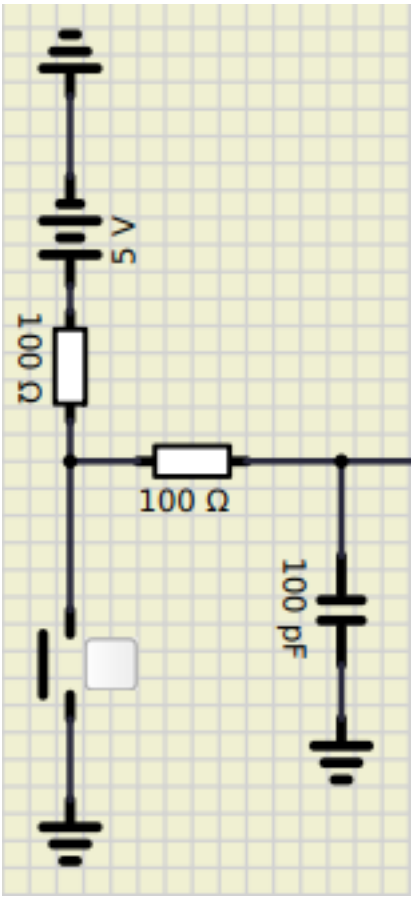

Figura 6: Filtro diseñado para eliminar el efecto del rebote del pulsador

Tomando en cuenta que la entrada del circuito es DC y, por lo tanto, no es necesario diseñar el filtro para atenuar un rango de frecuencias, se escogieron los valores de los elementos que componen dicho filtro arbitrariamente. Así que tomando un capacitor de $100\,pF$ junto con dos resistencias, cada una de $100\,\Omega$, se tiene que la constante de carga de este filtro sería:

$$\tau = R \cdot C = 200\,\Omega \cdot 100\,pF = 20\,ns \tag{2}$$

El periodo de tiempo obtenido es lo suficientemente grande como para eliminar cualquier distorsión en la señal de entrada, producido por el rebote, pero también es lo bastante pequeño como para que su efecto sea imperceptible por el usuario.

2.4. Diseño del bingo electrónico

Al unir todos los componentes antes expuestos se obtiene el circuito mostrado en la figura 7. Recapitulando, el circuito se encuentra conformado por el microcontrolador PIC12f675, un botón pulsador, un decodificador BCD, dos displays de 7 segmentos y el filtro pasivo diseñado.

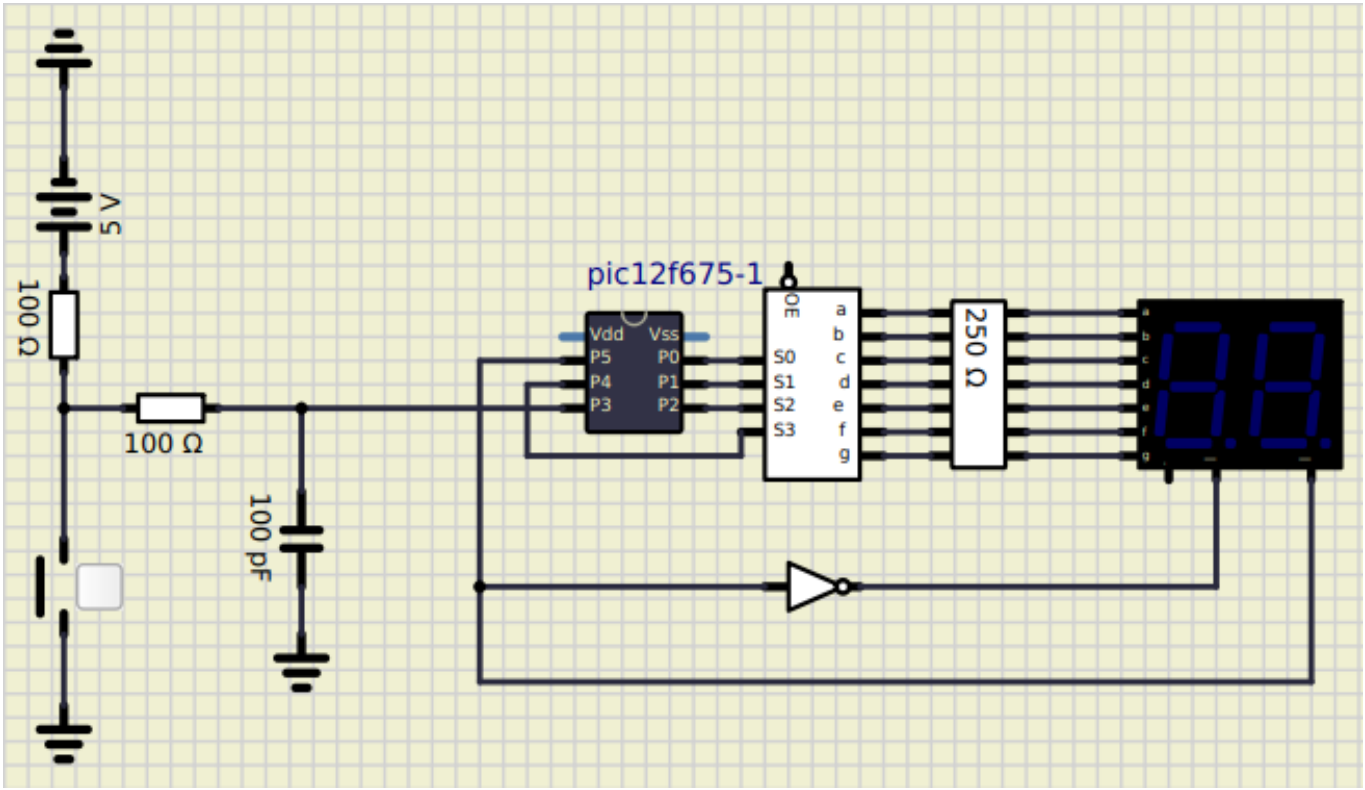

Figura 7: Esquemático del bingo electrónico diseñado

El GPIO3 funciona como la entrada de las señales generadas cuando se presiona el pulsador, mientras que los periféricos GPI0, GPI1, GPI2 y GPI4 operan como un grupo de salidas que juntas conforman un número binario de 4 bits. Estas 4 salidas se conectan a las entradas del decoder BCD, el cual se encargará de decodificar las series de 4 bits recibidas desde los GPIOs. El decodificador entonces pondrá en alto ($5\,V$) las salidas correspondientes a la combinación de 4 bits recibida. La corriente en dichas terminales pasará por el bloque de resistencias hasta llegar a los dos displays, encendiendo los segmentos respectivos.

Para poder utilizar los dos displays al mismo tiempo se optó por utilizar el GPIO5 como un selector entre ambos displays. En el firmware se configuró la salida de este periférico para que durante toda la simulación esté cambiando entre $0\,V$ y $5\,V$ a una muy alta frecuencia. Por otra parte, en la figura 7 se puede observar que los cátodos comunes de los displays están conectados al GPIO5, con la importante diferencia de que uno de los dos tiene un inversor de por medio. Con esta configuración se estará cambiando el display operante a una velocidad imperceptible, y esto virtualmente permite utilizar ambos displays al mismo tiempo.

A continuación se presenta una tabla con los precios aproximados de todos los elementos empleados en la construcción del circuito presentado en la figura 7.

Componente	Cantidad	Precio
Microcontrolador PIC12F675	1	$4.00
Resistencia de 100 Ω	2	$1.50
Resistencia de 250 Ω	7	$7.50
LED display de 7 segmentos	2	$2.50
Decodificador BCD	1	$3.95
Compuerta lógica inversora	1	$4.00
Total	$23.45	

Tabla 1: Tabla de precios de los componentes empleados

3. Desarrollo/Análisis de resultados

En el código implementado para el firmware primeramente se realizaron las siguientes configuraciones para los registros de TRISIO, GPIO, ANSEL y CMCON:

```
typedef unsigned int word;
word __at 0x2007 __CONFIG = (_WDTE_OFF & _WDT_OFF & _MCLRE_OFF);  // WDT y MCLR OFF
        .
        .
        .
ANSEL = 0;
CMCON = 7;
TRISIO=0b00001000;
GPIO = 0x00;
```

De esta manera se configuraron todos los GPIOs para que operaran en un estado preestablecido bajo y también para que funcionaran como salida, todo menos el GPIO3, el cual se configuró como una entrada.

Por otra parte, en el código se declararon las siguientes variables enteras antes de entrar a la función main():

```
int counter;
int binary0;
int binary1;
int binary2;
int binary3;
int binary4;
int binary5;
int binary6;
```

```
int binary7;
int tens;
int units;
```

La variable *counter* sirve como contador de los ciclos que el microcontrolador está en ejecución, mientras que las variables *tens* y *units* sirven para guardar las decenas y las unidades que conforman a cada número aleatorio generado. Por otra parte todas las variables *binary* tienen la función de guardar temporalmente el estado de las salidas de los periféricos GPIO0, GPIO1, GPIO2 y GPIO4.

Dentro de la función main, se ejecuta un loop while infinito, dentro del cual el *counter* aumenta con cada ciclo. Más adelante se observa que la variable *tens* se incrementa en 1 si *units* llega a 9, y además si *tens* supera a 9 esta variable se resetea a 0. Después se puede observar que al presionar el pulsador (ocasionando una lectura de 0 en el GPIO3), se asignan distintos valores a las variables *binary*, dependiendo del valor que tienen *units* y *tens*.

```
void main(){
    .
    .
    .
        while(1)
        {

                if(units == 9){
                        tens = tens + 1;
                }

                if (tens > 9){
                        tens = 0;
                }

                else if(GP3 == 0){
                        if(units == 0){
                                binary0 = 0;
                                binary1 = 0;
                                binary2 = 0;
                                binary3 = 0;
                                // delay(50);
                                continue;
                        }

                        if(units == 1){
                                binary0 = 1;
                                binary1 = 0;
                                binary2 = 0;
                                binary3 = 0;
                                // delay(50);
```

```
            continue;
    }
    if(units == 2){
            binary0 = 0;
            binary1 = 1;
            binary2 = 0;
            binary3 = 0;
            // delay(50);
            continue;
    }
    .
    .
    .
    if(tens == 0){
            binary4 = 0;
            binary5 = 0;
            binary6 = 0;
            binary7 = 0;
            // delay(50);
            continue;
    }

    if(tens == 1){
            binary4 = 1;
            binary5 = 0;
            binary6 = 0;
            binary7 = 0;
            // delay(50);
            continue;
    }
    if(tens == 2){
            binary4 = 0;
            binary5 = 1;
            binary6 = 0;
            binary7 = 0;
            // delay(50);
            continue;
    }
    .
    .
    .
```

Finalmente, se pasan los valores guardados en las variables *binary* a los GPIOs y estos conjuntos de valores se intercambian a una frecuencia alta (con solo un delay de 10), al mismo tiempo que se cambia el estado en la salida del GPIO5. De esta manera los códigos binarios para las decenas y las unidades se pasan prácticamente al mismo tiempo por los mismos GPIOs. Después de esto se verifica

que la variable *units* no supere 9, del contrario se vuelve a resetear a 0, y después se incrementa en una unidad por cada ciclo.

```
delay(10);
                GP5 = ~GP5;

                GP0 = binary0;
                GP1 = binary1;
                GP2 = binary2;
                GP4 = binary3;

                delay(10);
                GP5 = ~GP5;

                GP0 = binary4;
                GP1 = binary5;
                GP2 = binary6;
                GP4 = binary7;

                if (units > 9){
                        units = 0;
                }

                units = units + 1;
```

En la sección de Apéndices se presentan algunos ejemplos de los resultados obtenidos con el diseño implementado. Con ellos se puede verificar que se logró cumplir con el objetivo de generar un número aleatorio de dos dígitos, utilizando el microcontrolador PIC12f675, y desplegarlo en dos displays.

4. Conclusiones y recomendaciones

- El PIC12f675 es un microcontrolador con capacidades básicas y cuenta con poca memoria. Debido a esto no se pudo implementar un algoritmo muy complejo para generar los números aleatorios y tampoco se pudo guardar un registro de los números que ya salieron en una ejecución.
- La mayor limitación que se tuvo fue a nivel de hardware, ya que solo se tenían 6 GPIOs para conectar al pulsador y para pasar dos números decimales desde el microcontrolador hasta los dos displays de 7 segmentos. Sin embargo, esta limitación se superó implementando la técnica de multiplexación en las salidas de los GPIOs.

5. GIT

El código fuente que se generó para este laboratorio se encuentra en el siguiente enlace:

https://github.com/jdgg98/Laboratorio-de-Microcontroladores/tree/master/lab1-bingo

6. Bibliografía

[1] Microchip. *PIC12F629/675 Data Sheet 8-Pin FLASH-Based 8-Bit CMOS Microcontrollers*, February 2003.

7. Apéndices

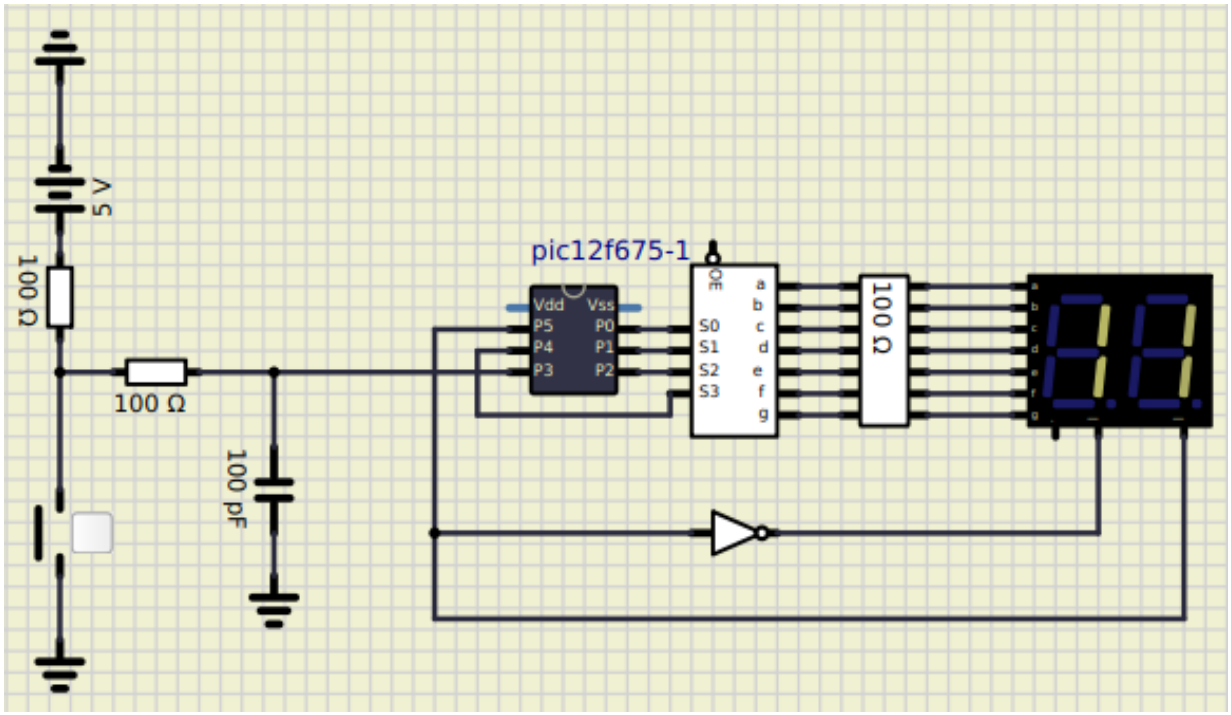

Figura 8: Número aleatorio generado: 11

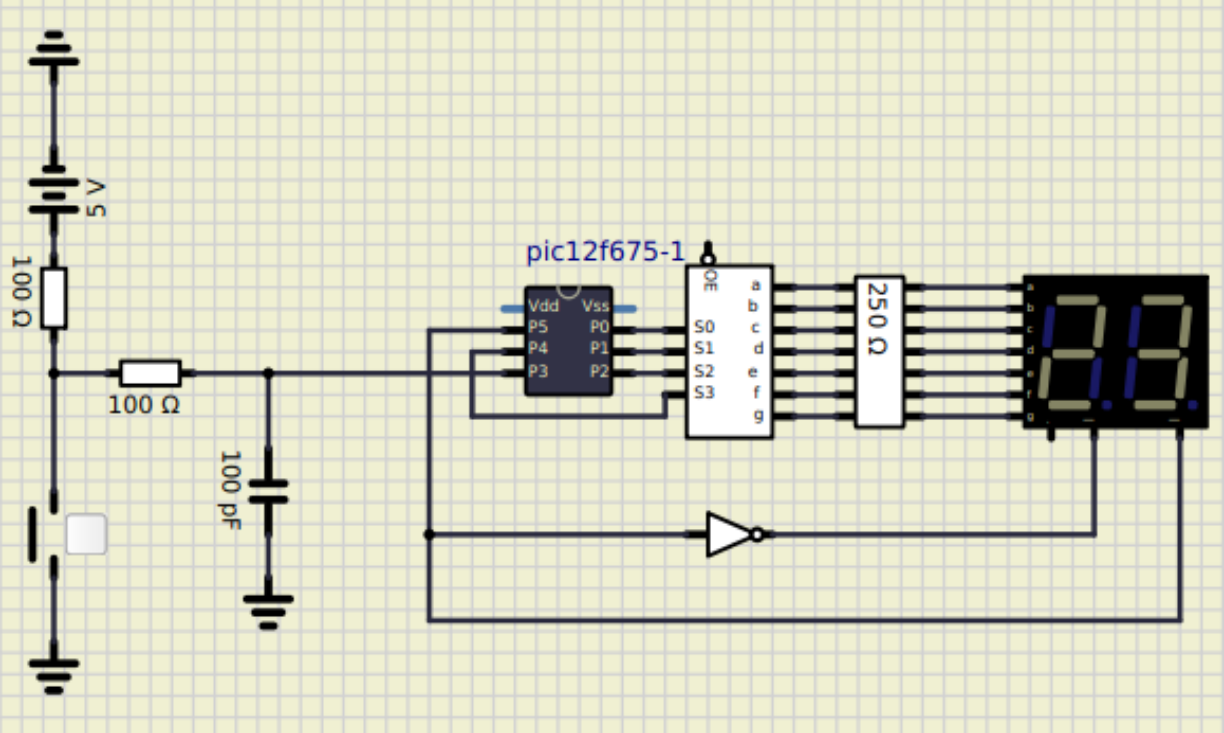

Figura 9: Número aleatorio generado: 23

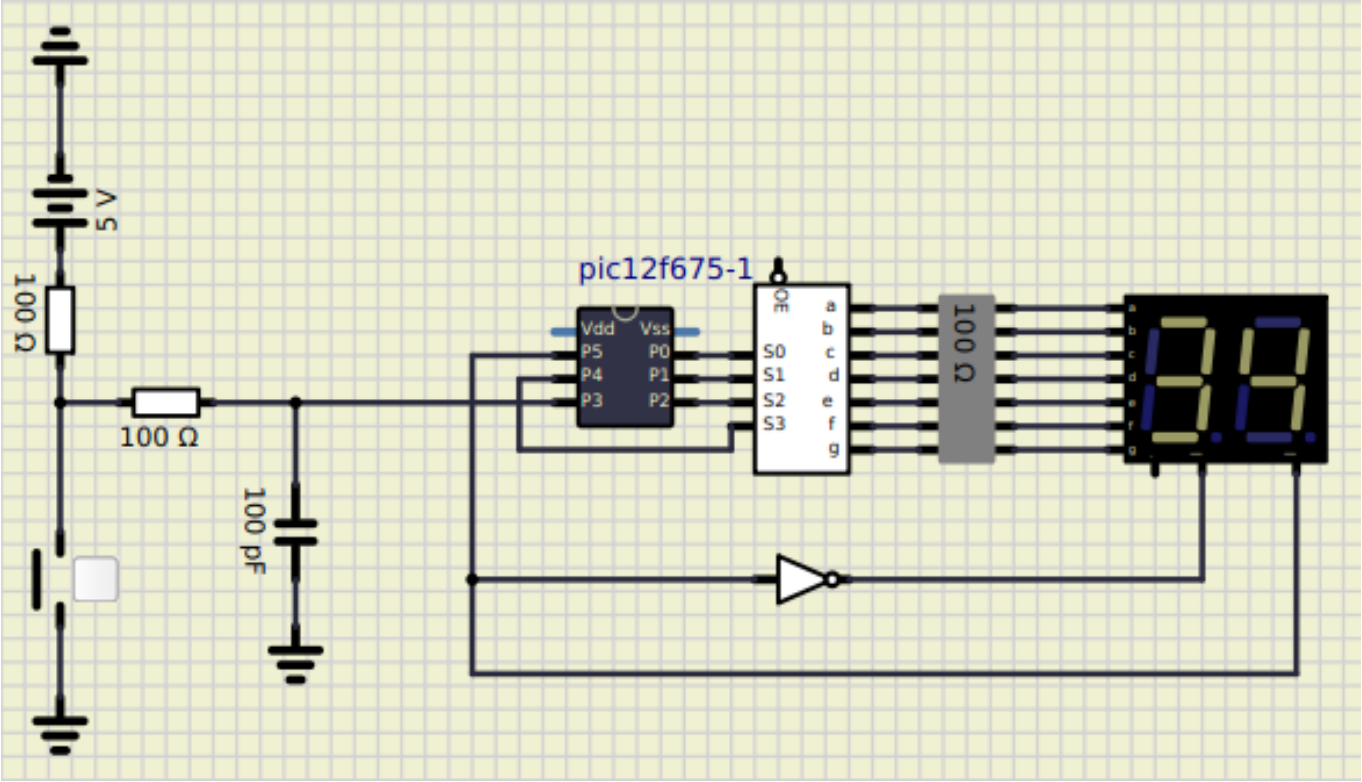

Figura 10: Número aleatorio generado: 34

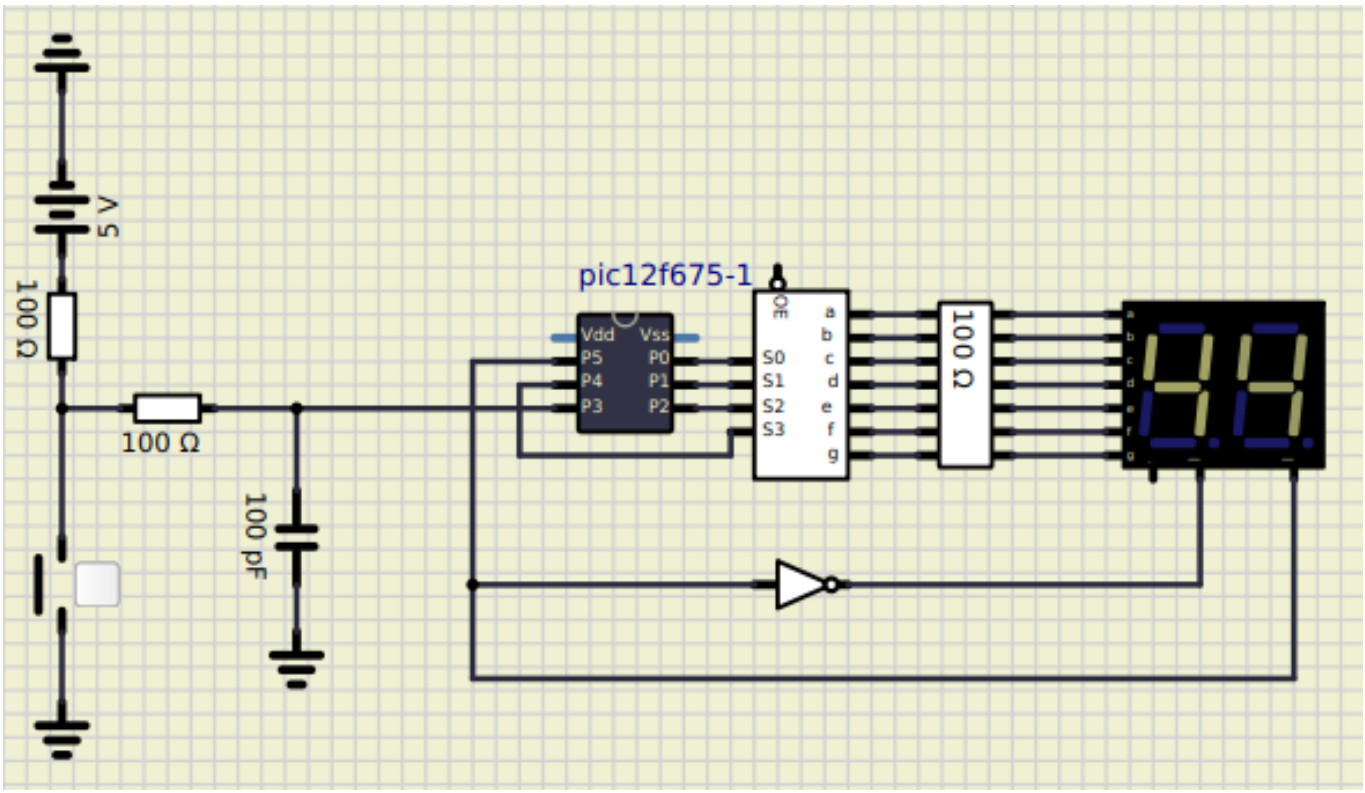

Figura 11: Número aleatorio generado: 44

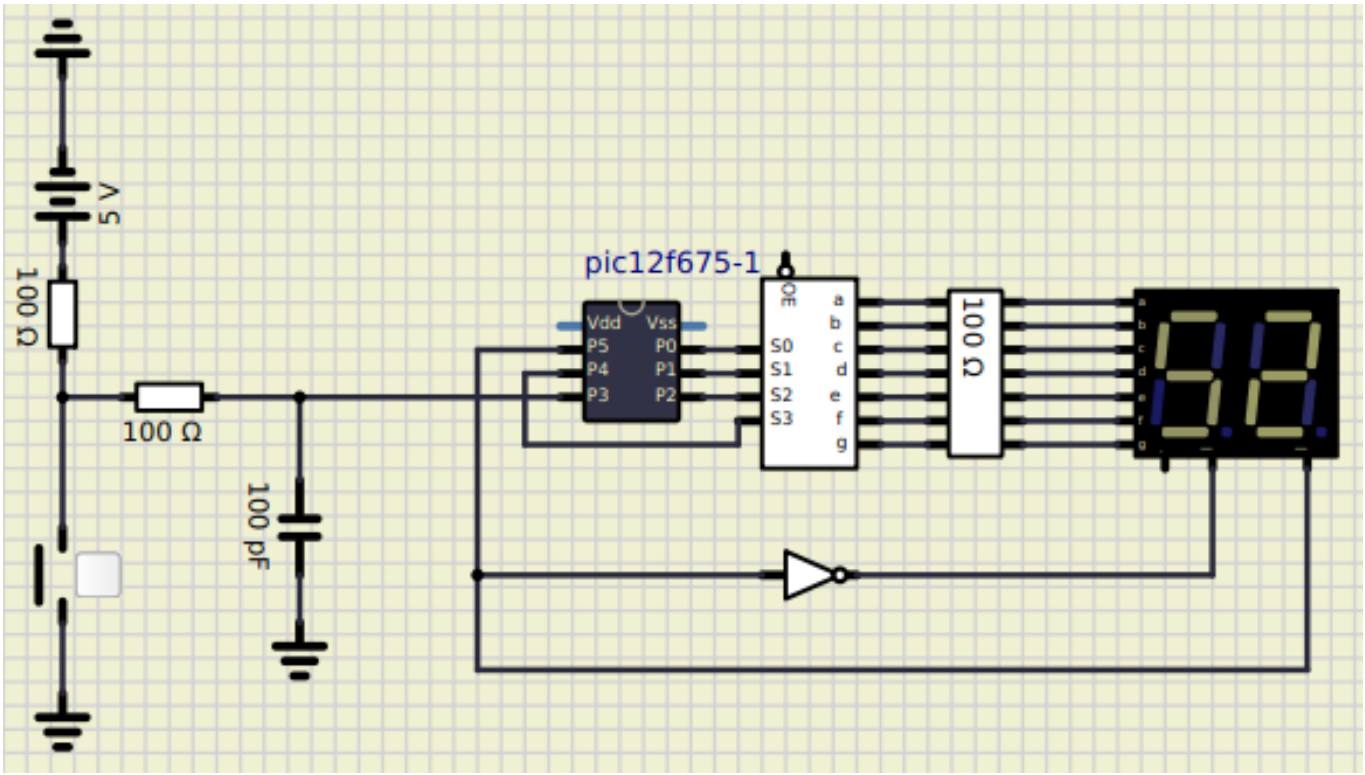

Figura 12: Número aleatorio generado: 52

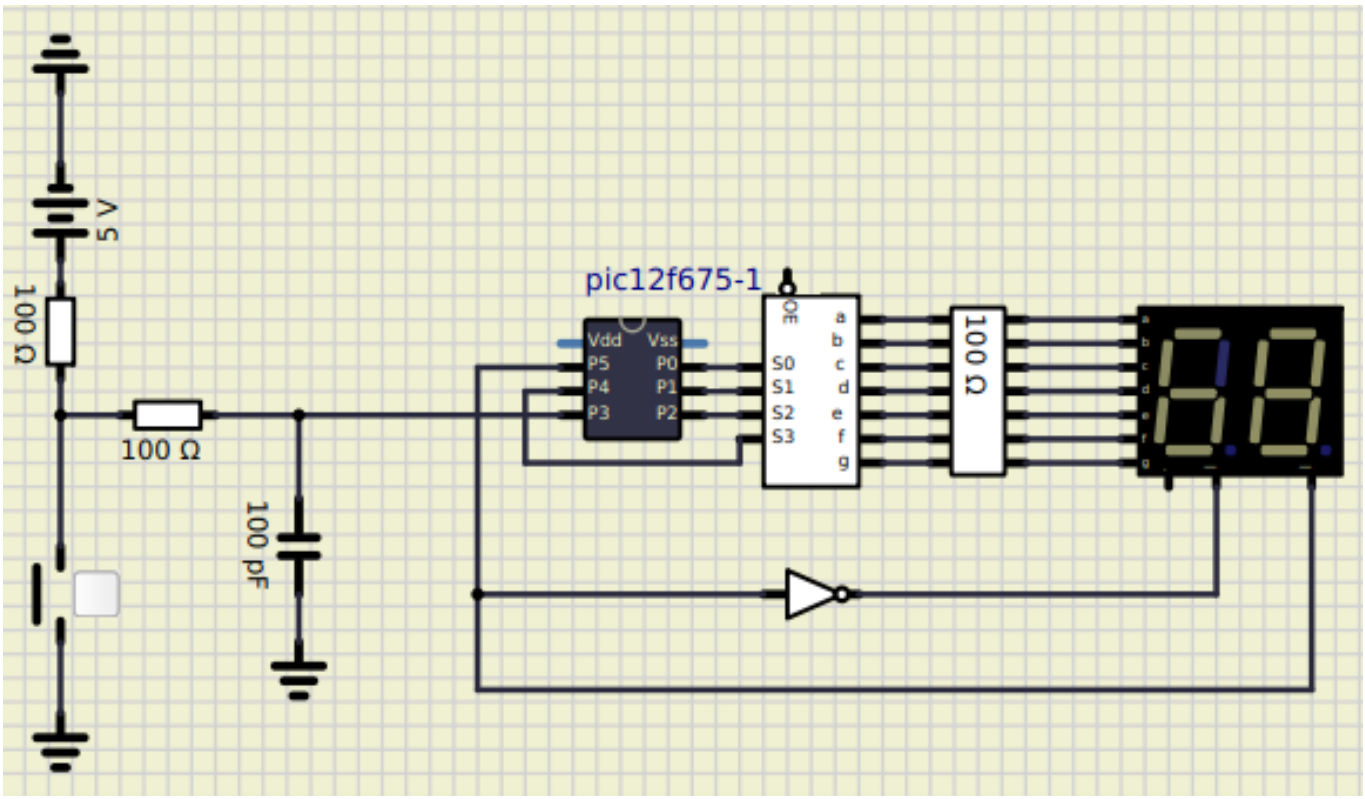

Figura 13: Número aleatorio generado: 68

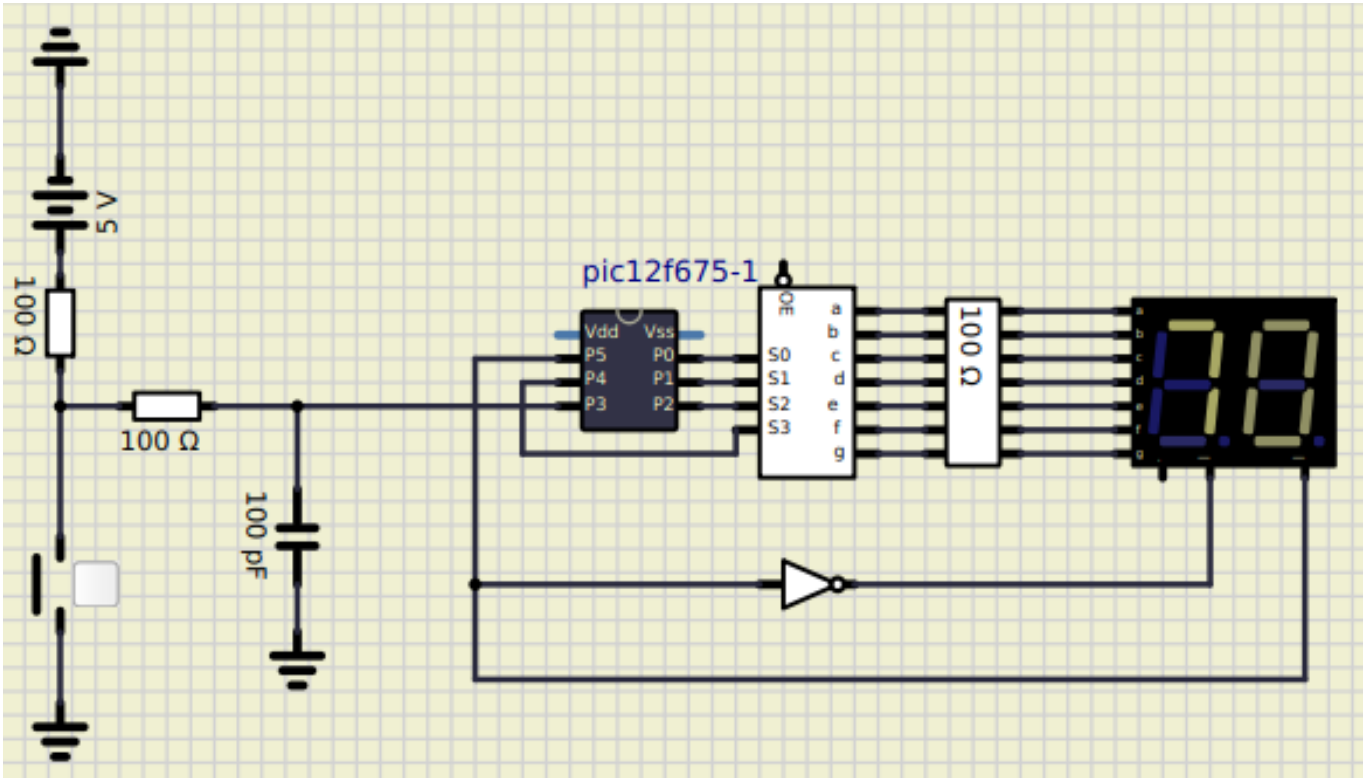

Figura 14: Número aleatorio generado: 70

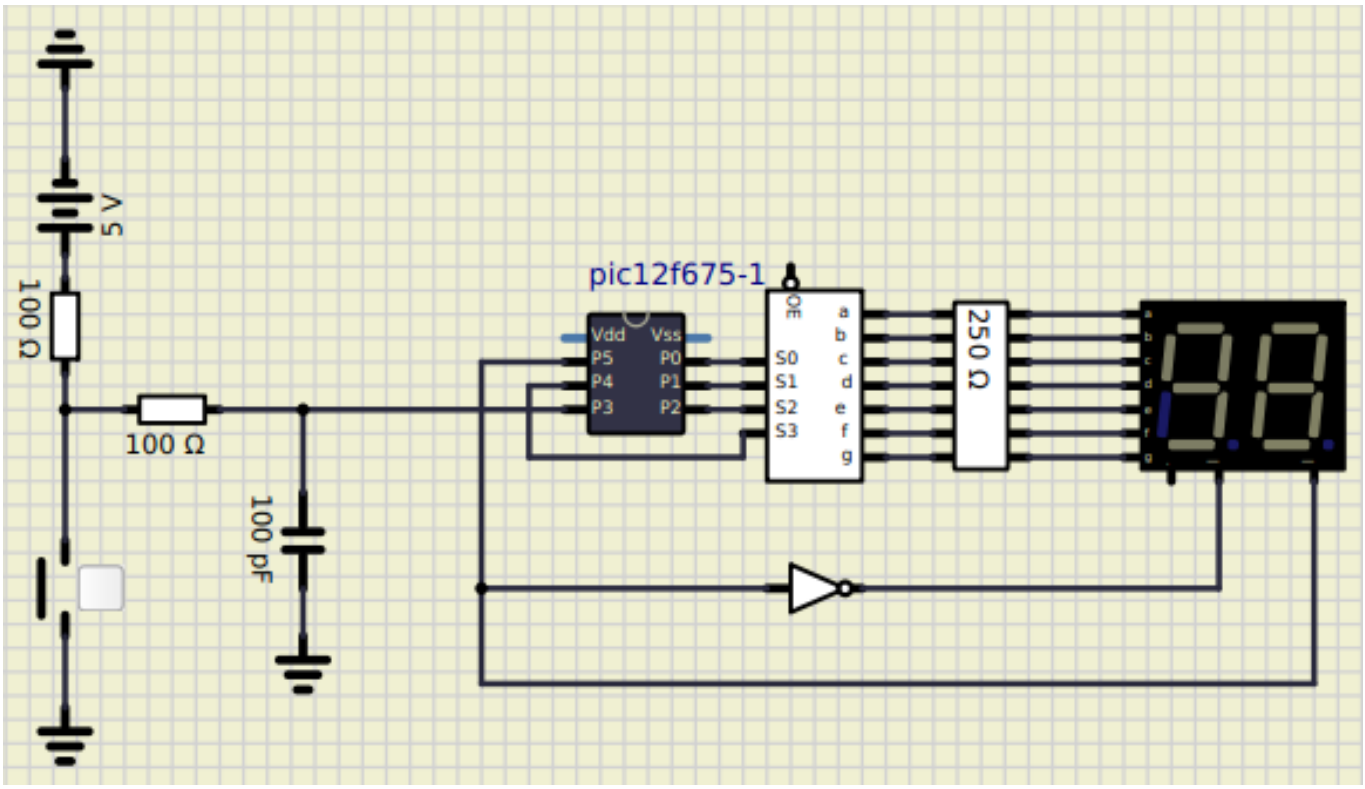

Figura 15: Número aleatorio generado: 98