Laboratorio 1: GPIO
Curso UCR: Eie-0624

Profesor: Marco Villalta Fallas

Dualok Fonseca Monge B42629

1. Introducción:

Este laboratorio consiste en el diseño y simulación de una tómbola de bingo, utilizando dos displays de 7 segmentos y el microcontrolador PIC12f683. Para simplificar el diseño, se requieren sacar 16 números aleatorios y luego parpadear los display de 7 segmentos con un 99 para indicar que se terminó el juego

1.1 Resumen:

Se generó el firmware del microcontrolador a partir de código en C en conjunto con una simulación en SimulIDE, donde se logró cumplir con todos los requisitos del laboratorio.

1.2 Conclusiones:

Como conclusión, se logró crear tanto la simulación y diseño del circuito, como el código en C de forma exitosa, Algunas consideraciones y recomendaciones importantes para el uso del PIC12F683 son:

- Es importante aprovechar al máximo la poca memoria disponible
- Utilizar caracteres sin signo en caso de que se requiera números menores a 255
- No crear variables innecesarias
- Es importante para laboratorios posteriores encontrar herramientas para debugear el programa.

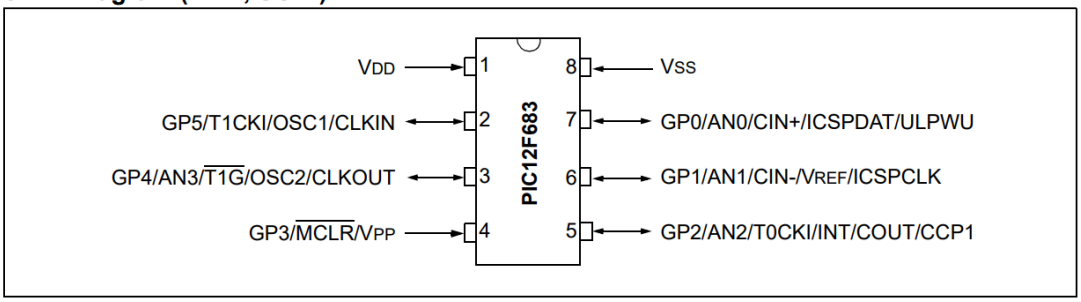
2. Nota teórica:

2.1 Información general del MCU:

El PIC12f683 es un microcontrolador CMOS de 8 bits y 8 pines el cual contiene un CPU RISC de alto rendimiento con únicamente 32 instrucciones por aprender, cuenta con capacidad de interrupción, tiene algunas características como un watchdog timer WDT con oscilador independiente. = En cuanto a memoria este PIC cuenta con 2048 palabras de memoria flash programable, 128 bytes de static RAM y 256 bytes de EEPROM. En cuanto a puertos de entrada y salida, posee 6 únicamente y 4 canales de 10bit A/D. La siguiente figura muestra el diagrama de pines del MCU.

PIC12F683

8-Pin Diagram (PDIP, SOIC)



Especificaciones eléctricas:

15.0 ELECTRICAL SPECIFICATIONS

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} ..... -0.3V to +6.5V
Voltage on $\overline{MCLR}$ with respect to Vss ..... -0.3V 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 ..... 95 mA
Maximum current into V_{DD} pin ..... 95 mA
Input clamp current, I_{IK} (V_I < 0 or V_I > V_{DD}) ..... ± 20 mA
Output clamp current, I_{OK} (V_o < 0 or V_o > 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 GPIO ..... 90 mA
Maximum current sourced GPIO ..... 90 mA

Note 1: Power dissipation is calculated as follows: $P_{DIS} = V_{DD} \times \{I_{DD} - \sum I_{OH}\} + \sum \{(V_{DD} - V_{OH}) \times I_{OH}\} + \sum (V_{OI} \times I_{OL})$.

† 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.

2.2 Periféricos

El microcontrolador PIC12f683 cuenta con los siguientes periféricos:

- 6 pines de entrada/salida con dirección individual controlable
- Fuente y drenaje de alta corriente para control directo de LEDs
- Comparador analógico
- Convertidor Analógico a digital (ADC)

Sin embargo, los únicos periféricos utilizados en este laboratorio son, la memoria de programa (FLASH) y los puertos de propósito general GPIO.

2.3 Registros

El microcontrolador tiene varios registros configurables, sin embargo para este laboratorio se utilizan únicamente el registro CONFIG y los registros de flujo de entradas y salidas TRISIO (Tristate input/output) y GPIO(General purpose input/output).

Hay 6 GPIO y el registro TRISIO dicta el flujo de estos pines.

- GPIO: Es un puerto de 6 bits bidireccional, este registro 1 es alto, 0 es bajo.

REGISTER 4-1: GPIO: GENERAL PURPOSE I/O REGISTER

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

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

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

bit 5-0 **GP<5:0>**: GPIO I/O Pin bit
1 = Port pin is > VIH
0 = Port pin is < VIL

- TRISIO: Configura el flujo de datos de los GPIO, el tercer bit de GPIO siempre se lee como un input, por lo que el TRISIO(3) siempre lee un 1. 1 es un input y 0 es un output

REGISTER 4-2: TRISIO GPIO TRI-STATE REGISTER

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

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

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

bit 5:4 **TRISIO<5:4>:** GPIO Tri-State Control bit
1 = GPIO pin configured as an input (tri-stated)
0 = GPIO pin configured as an output

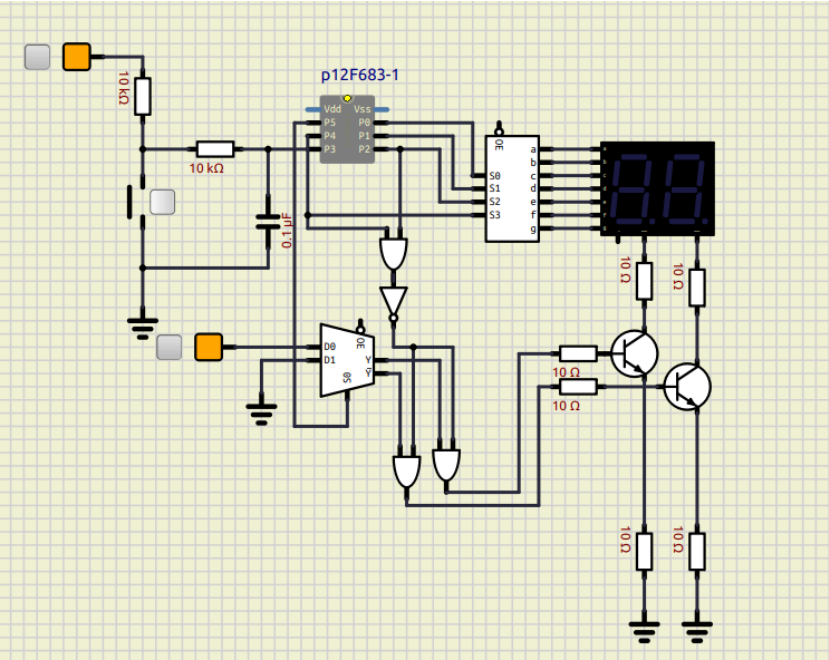
bit 3 **TRISIO<3>:** GPIO Tri-State Control bit
Input only

bit 2:0 **TRISIO<2:0>:** GPIO Tri-State Control bit
1 = GPIO pin configured as an input (tri-stated)
0 = GPIO pin configured as an output

- CONFIG

2.4 Diseño de circuito:

El diseño final del circuito es el siguiente



2.5 Lista de componentes

Algunos componentes los venden en sets, para calcular el precio se divide entre la cantidad de componentes por set, pero en realidad se compraría por ejemplo 10 resistores de 10k, etc. Los precios están basados en la tienda Jameco, aunque es una empresa de EEUU, da una idea de cómo andan los precios

Componente	Cantidad	Tipo / Valor	Precio c/u
Resistor	2	10k	$0.0055
Resistor	2	1k	$0.0055
Resistor	4	100	$0.0055
Mux 2 a 1	1	74LS157	$0.89
Conversor BCD-7seg	1	CD4511	$0.85
Compuerta AND	3	74LS08	$1.09
Inversor NOT	1	74LS04	$0.89
Transistores BJT	2	2N3904	$0.024
Display 7 segmentos	1	TM1637	$4.95

Botón pulsador	1	red pushbutton	$1.09

2.6 Conceptos/Temas de laboratorio

Convertidor BCD a 7 Segmentos:

Se va a utilizar un convertidor de BCD a 7 Segmentos, esto permite mapear valores en BCD (4 bits) a sus correspondientes configuraciones de LED de 7 segmentos. De esta forma se tendrán 4 salidas para encender los LEDs, 1 entrada para el botón de push y sobra un puerto por si lo llegara a necesitar. Los números decimales codificados en binario son:

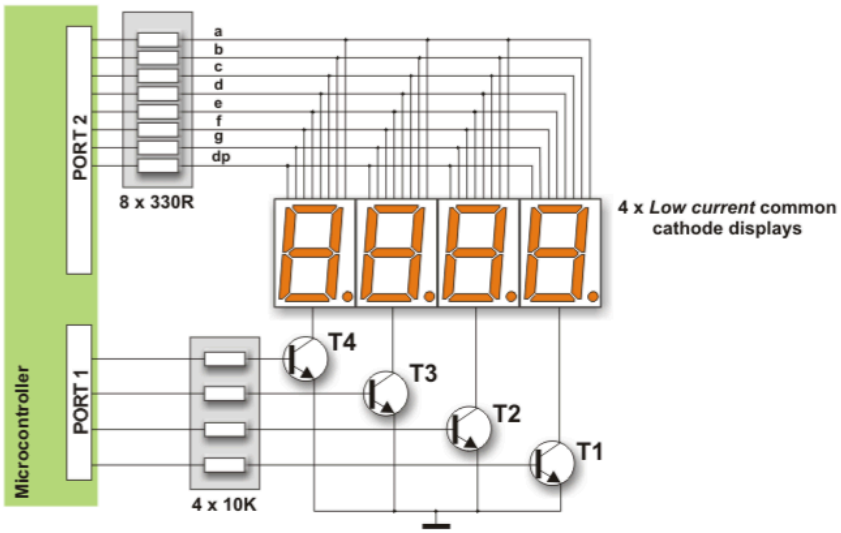
0000: 1 0011: 3 0110: 6 1001: 9

0001: 1 0100: 4 0111: 7

0010: 2 0101: 5 1000: 8

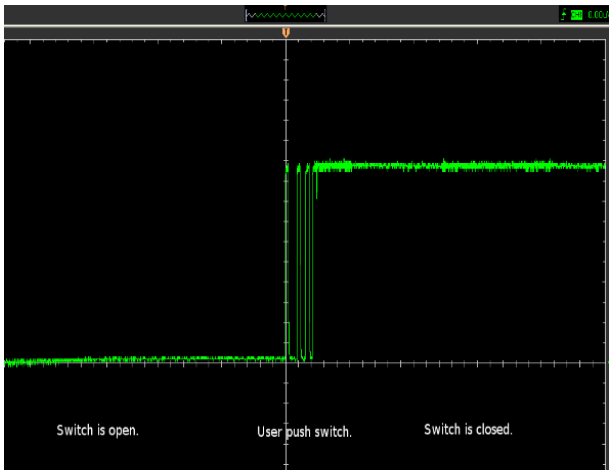
Multiplexación:

Se utilizará la técnica de multiplexación para dar la ilusión de que ambos displays de LED están encendidos al mismo tiempo, pero están conmutando con una frecuencia de 60hz haciendo uso de dos transistores BJT.

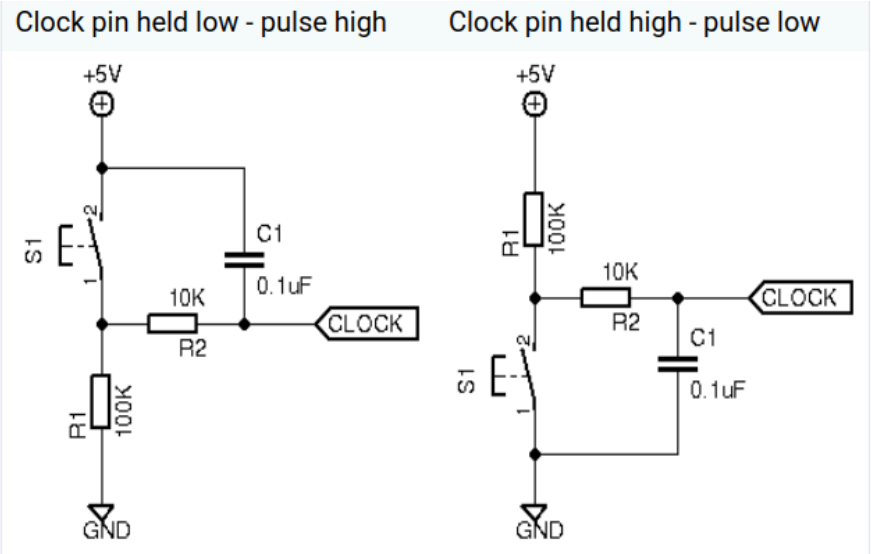


Como forma de probar este funcionamiento, se creó una simulación con un valor cambiante con la misma frecuencia de conmutación, esto se discute en la sección *3.2 Análisis de circuito*

Debouncing: Al activar un botón en un microcontrolador, ocurre un fenómeno que no es perceptible para el ojo humano, pero que suele confundir al microcontrolador. Se trata de un rebote mecánico al activar el botón, como se muestra en la siguiente figura



Hay 2 formas de abordar este problema: La primera es utilizando el hardware, mediante un circuito RC para evitar los cambios bruscos de frecuencia y la otra es agregando tiempo dentro del software, el capacitor funciona como un circuito abierto a bajas frecuencias. En la figura, además se agregan las resistencias de pull down y pull up correspondientes.



La segunda es asignar un valor de rebote al software tal que llegue a un máximo la primera vez que es activado el botón y si el valor rebote no es cero, que baje en cada ciclo y hasta que llegue a 0 nuevamente puede ser leída la entrada como una pulsación.

3. Desarrollo y análisis

3.1 Análisis del programa:

☐ **Generacion de numeros aleatorios**

Para la generación de números aleatorios se hace uso de un linear feedback shift register (registro de desplazamiento con realimentación lineal), se emplea una función que recibe un número, inicialmente una semilla, de 8 bits y realimenta los bits 7, 5, 4 en una compuerta XOR en el bit menos significativo y se desplaza hacia la derecha

El polinomio de realimentación es:

$$x^7 + x^5 + x^4 + x^0$$

☐ **Comprobación de números repetidos**

Se utiliza una función llamada check_num que revisa si el numero que salio en la góndola del bingo salio repetido, si salio repetido, entonces no lo guarda en el array de números y retorna el mismo índice para volver a probar con otro número, en caso de no estar repetido, lo guarda en el array de números y retorna el siguiente índice.

☐ **Modificación de registros, bits y macros necesarios del PIC**

Los registros utilizados y modificados en este laboratorio fueron.

- TRISIO = 0x00 para asignar el flujo de datos y setear todos los GPIO como salidas, aunque el bit TRISIO[3] siempre lee 1, es decir siempre es una entrada.
- GPIO: Este fue el registro más utilizado, ya que para obtener las combinaciones necesarias de salidas para representar los números en los LED de 7 segmentos, fue necesario estar cambiando el estado (alto o bajo) de este registro.
- Se modifica el registro CONFIG ubicado en la dirección 0x207, mediante la instrucción:

```
typedef unsigned int word;
word __at 0x207 __CONFIG = (_WDTE_OFF & _WDT_OFF & _MCLRE_OFF);
```

De este modo se desactiva el temporizador de vigilancia o WatchdogTimer (WDT) y el master clear (MCLR)

☐ **Lectura del estado del botón:**

Se le llama PUSH_B a la entrada de lectura GP3. Para leer el botón, se utiliza el puerto GP3, como se utiliza una red de pull up, el pin siempre está en alto, hasta que se presiona el botón se registra la entrada como un cero lógico, por esta razón para la lectura se utiliza !PUSH_B. Como recomendación de la lectura sobre

debouncing, se utilizó también un valor de rebote en el código, de modo que cada vez que se presiona el botón, el valor de rebote llega a su máximo y el programa no registra más entradas hasta que ese valor de rebote llegue a 0. Cada vez que se registra una entrada se llaman las funciones para generar el número, obtener las decenas y unidades y verificar si el número ya existe.

```
else if(!PUSH_B && (valor_rebote == 0)){
        valor_rebote = 30;
        numero = lfsr(numero);
        decenas = numero/10;
        unidades = numero%10;
        index = check_num(tiros, index, numero);
}
// logica para el debouncing
else if(valor_rebote > 0){
        valor_rebote--;
}
```

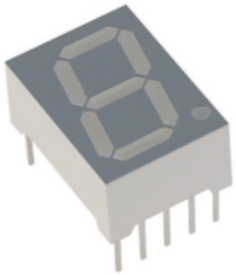
- ☐ **Diagrama de flujo del programa principal:**

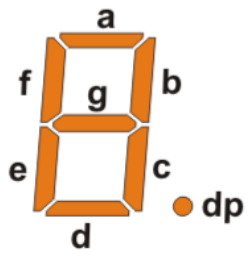
3.2 Análisis del circuito:

Para diseñar el circuito, se dividió el problema en varios subproblemas y se fue ejecutando cada uno paso a paso, para esto se tomó en cuenta lo siguiente:

- ☐ **LED de 7 segmentos**

  Se trata de varios LEDS en la misma carcasa de plástico, donde de acuerdo a la configuración, se pueden escribir números (0 al 9) y letras. Para este laboratorio nos interesa con que configuracion de 8 bits se genera cada número





Digits to display	Display Segments							
	dp	a	b	c	d	e	f	g
0	0	1	1	1	1	1	1	0
1	0	1	1	0	0	0	0	0
2	0	1	1	0	1	1	0	1
3	0	1	1	1	1	0	0	1
4	0	0	1	1	0	0	1	1
5	0	1	0	1	1	0	1	1
6	0	1	0	1	1	1	1	1
7	0	1	1	1	0	0	0	0
8	0	1	1	1	1	1	1	1
9	0	1	1	1	1	0	1	1

- [ ] **Cantidad de GPIO disponibles como salidas**

  Según el registro GPIO, se tienen 6 puertos disponibles, pero solo 5 son salidas, ya que el puerto GP3 solo se puede configurar como entrada. Es decir, hay que idear una forma de encender ambos LEDs de 7 segmentos con esas 5 salidas, dígase GP0, GP1, GP2, GP4 y GP5.

REGISTER 4-1: GPIO: GENERAL PURPOSE I/O REGISTER

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

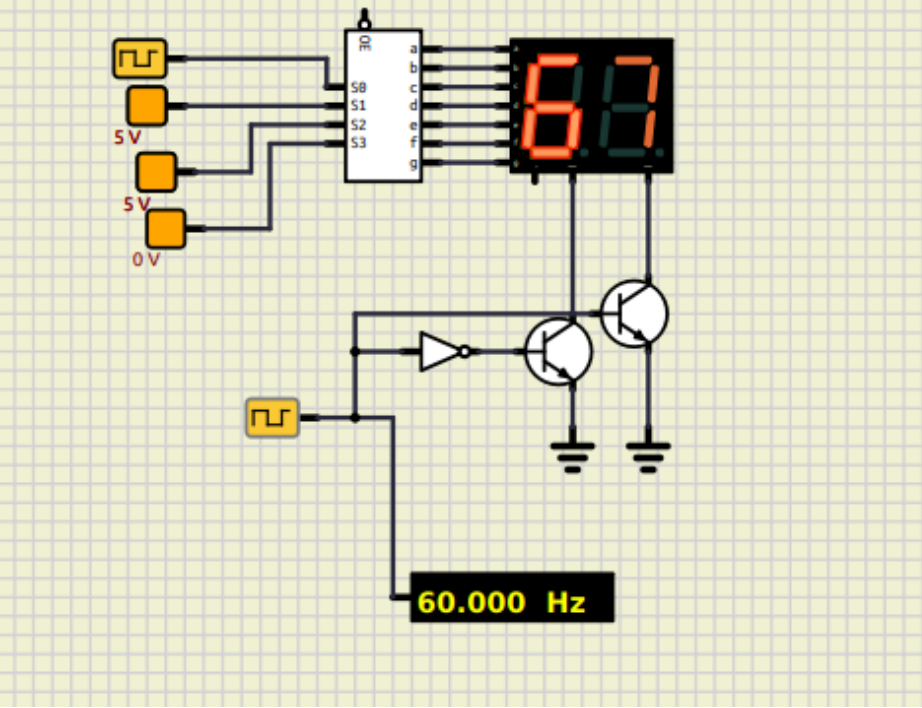
Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

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

bit 5-0 **GP<5:0>**: GPIO I/O Pin bit
1 = Port pin is > VIH
0 = Port pin is < VIL

☐ **Multiplexación:**

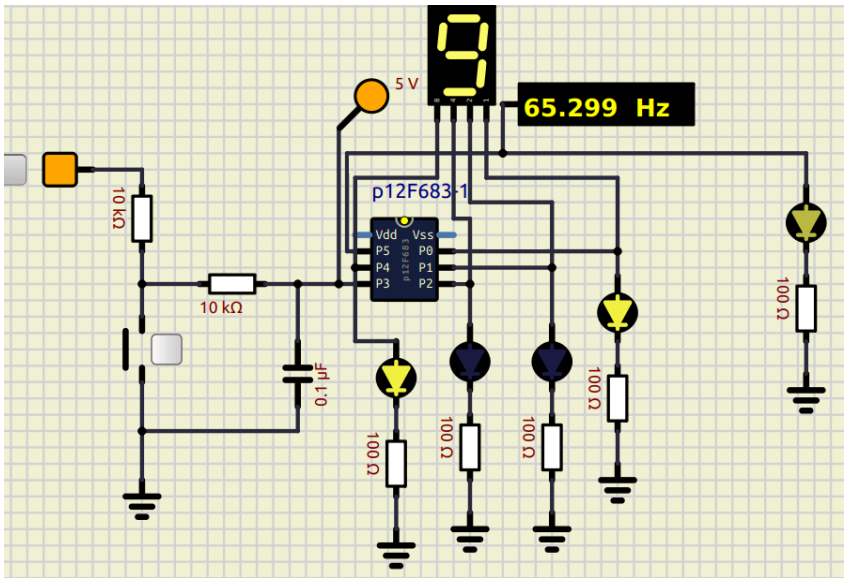
Se utilizará la salida que falta, GP5 para multiplexar los LEDs, para probarlo, se simuló sin código este prototipo inicial, donde el reloj cambiando a 60hz representa el comportamiento esperado de GP5



☐ **Circuito contador de pulsaciones:**

Luego se creó otra sección del circuito para asegurar que el contador de pulsaciones del botón funciona y que cuenta hasta 10, guardando los datos en un arreglo, como se explico en la seccion *Analisis de programa*,

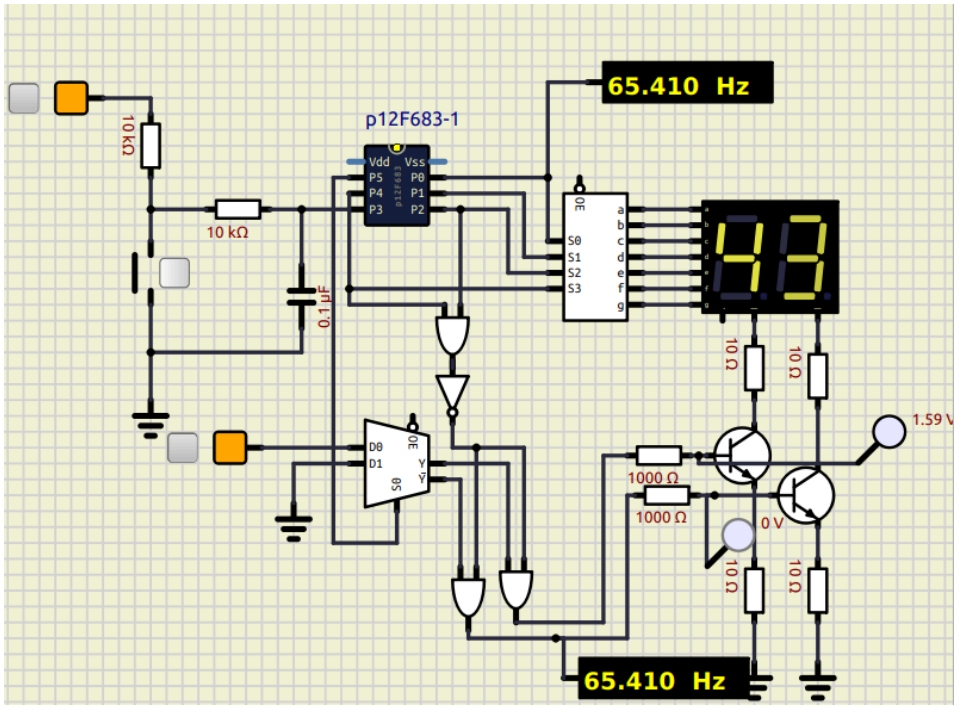
El contador se puede ver funcionando en la siguiente imagen.



Se puede notar que para este punto, se agregó una red de pull up y un filtro RC para filtrar las altas frecuencias y así minimizar el efecto del rebote del botón

- [ ] **Análisis del circuito completo**

Anteriormente se analizó cómo funciona la multiplexión de la salida GP5 y la red de pull up para la lectura del botón en la entrada GP3, en el circuito completo se puede observar además un Mux, que se utiliza para conmutar la salida y lógica combinacional agregada al circuito de conmutación. Nótese cómo las salidas GP4 y GP2(correspondientes a los 2 bits más significativos del número en BCD) se conectan a una compuerta AND seguido de un inversor esto genera un 1 constante en la salida a menos que GP4 = GP2 = 1. Ya que ningún número en BCD en el alfabeto numérico decimal requiere ambos bits en alto, es decir (1 1 X X), se utilizan estos bits como método para bloquear ambos transistores. de este modo se pueden apagar ambos display de 7 segmentos al poner una combinación tipo (1 1 X X). En el código se elige hacerlo con un 15, equivalente a F en hexadecimal



Con lo explicado anteriormente, el circuito logra mostrar hasta 10 números pseudo aleatorios y al final se parpadea el número 9, intercambiando el mapeo a BCD con F para crear el parpadeo

4. Conclusiones y recomendaciones

Como conclusión el laboratorio se realizó con éxito, se lograron los objetivos del programa. Haciendo uso de los registros del programa y los puertos de entrada y salida de propósito general, se logró diseñar y simular una góndola de bingo, mostrando 10 valores aleatorios diferentes en dos display de 7 segmentos, utilizando técnicas de electrónica y programación. El programa y el diseño de circuito funcionan aparentemente de forma perfecta.

Como recomendaciones, es necesario utilizar bien la memoria del MCU, ya que es muy limitada en comparación con una computadora. Se puede hacer uso de unsigned char para representar números de 8 bits sin signo, que van desde 0 hasta 255. Es de suma importancia. además controlar el rebote de los botones mecánicos, ya que a simple vista no notamos estas perturbaciones que generan cambios muy rápidos en el estado del botón, llegando a registrarse en el microcontrolador. Por

último, es necesario controlar bien los retardos y sincronización de los cambios de estado de los puertos.

5. Bibliografía

[1]. Christoffersen J.(2015). Switch bounce and how to deal with it, https://www.allaboutcircuits.com/technical-articles/switch-bounce-how-to-deal-with-it/

[2]. CMOS Microcontrollers.PIC12F683 Data Sheet 8-Pin FLASH-Based 8-Bit, 2010.

[3]. Milan Verle. PIC microcontrollers

6. Git

El repositorio de git se encuentra disponible en el enlace: https://github.com/Dualock/Microcontroladores-Labs

Específicamente en la carpeta: Microcontroladores-Labs/laboratorios/lab1-GPIO/Codigo y Simulacion