Universidad de Costa Rica

Escuela de Ingeniería Eléctrica

IE-0624: Laboratorio de Microcontroladores

Reporte de laboratorio #2: GPIOs, Timers y FSM

Jose Mario Navarro Bejarano

B75398

II-2024

Contenido

Introducción

Los microcontroladores son dispositivos que permiten realizar tareas relativamente complejas de forma sencilla y automatizada. Entender su funcionamiento y sus características es fundamental para poder utilizarlos adecuadamente y aprovechar al máximo sus capacidades. En este laboratorio se amplia el conocimiento sobre estos, expandiendo lo que se aprendió en el laboratorio anterior.

En específico, en este laboratorio se buscó realizar un juego de simón dice haciendo uso de un ATTiny43-13, que encienda leds en cierto orden y permita al usuario accionar unos botones que estén en el mismo orden en que se encendieron los leds. Se deben presionar en el mismo orden para ganar la ronda y aumentar la dificultad, de lo contrario, se pierde el juego y se reinicia la secuencia. Para realizar este juego se hizo uso de los timer que incluye el microcontrolador, así como de las interrupciones que posee, permitiendo de esta forma que los periféricos puedan decidir cuándo notificar al CPU cuando se presiona un botón, y que las interrupciones de los timer sean las encargadas de decidir cuánto tiempo se encienden los leds.

Los archivos del laboratorio se pueden encontrar en el siguiente repositorio de GitHub: https://github.com/marionabe/Lab_micros

Nota teórica

Información General del ATtiny4313

El ATtiny4313 es un microcontrolador de 8 bits que posee, entre otras funciones, timers e interrupciones, así como pines GPIO.

En la figura 1 se muestra el diagrama obtenido de la hoja de datos del fabricante. En esta se pueden observar todos los pines con sus diferentes funcionalidades.

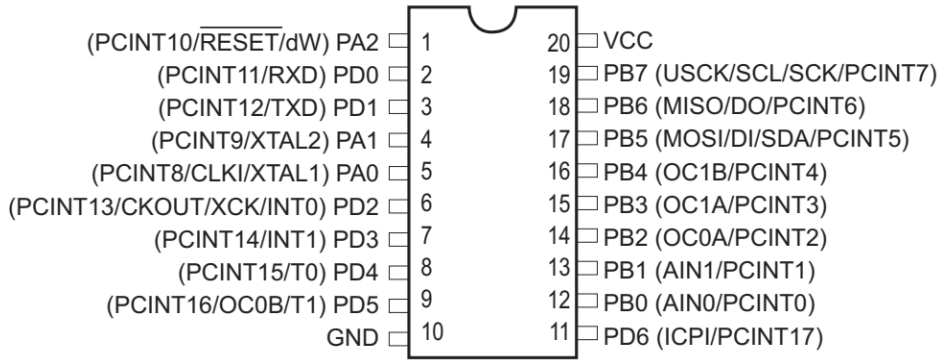

Figura 1: Diagrama de pines del ATtiny4313. Obtenido de la hoja de datos del fabricante

También es importante tener claras las especificaciones eléctricas de este microprocesador, en especial las corrientes en sus pines. En la figura 2 se muestra una tabla con el resumen de las características típicas de este microcontrolador.

Operating Temperature	-55°C to +125°C
Storage Temperature	-65°C to +150°C
Voltage on any Pin except $\overline{\text{RESET}}$ with respect to Ground	-0.5V to V_{CC}+0.5V
Voltage on $\overline{\text{RESET}}$ with respect to Ground	-0.5V to +13.0V
Maximum Operating Voltage	6.0V
DC Current per I/O Pin	40.0 mA
DC Current V_{CC} and GND Pins	200.0 mA

Figura 2: Características eléctricas del ATtiny4313. Obtenida de la hoja de datos del fabricante.

Por otra parte, en la figura 3 se muestra un diagrama de bloques de la estructura interna de este microcontrolador.

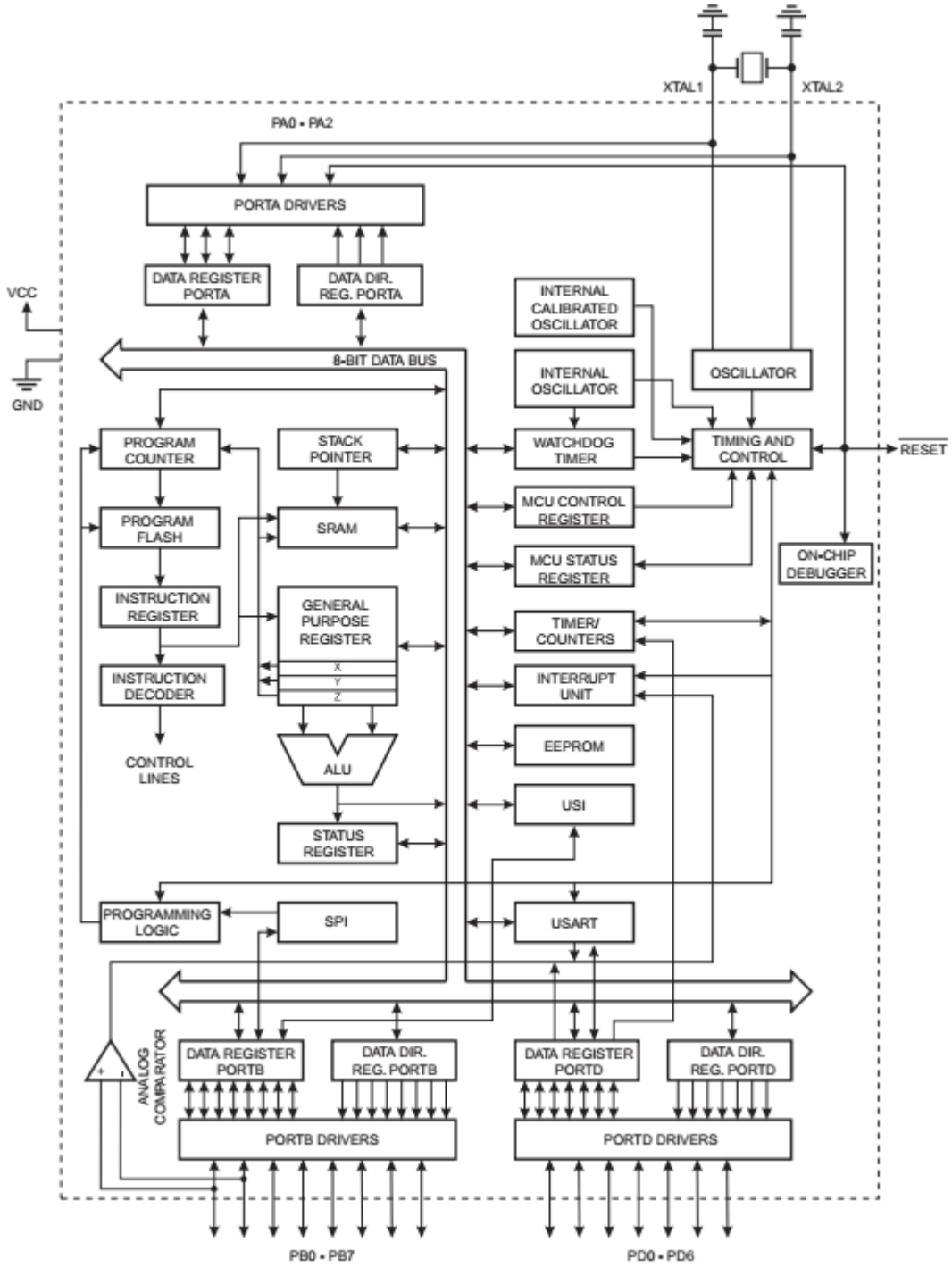

Figura 3: Diagrama del bloque de la construcción interna del ATtiny4313. Obtenido de la hoja de datos del fabricante.

Registros utilizados:

Para poder completar este laboratorio fue necesario hacer uso de diversos registros que posee el microcontrolador, a continuación, se dará una pequeña explicación de los registros mas relevantes y cómo fueron utilizados dentro del laboratorio.

Registros DDRB:

El registro DDRB es el que contiene la información sobre si cada pin se configura como entrada o como salida. Para el laboratorio fue utilizado para indicar que los pines conectados a los botones son entradas y los pines conectados a los leds son salidas.

DDRB – Port B Data Direction Register

Bit	7	6	5	4	3	2	1	0	
0x17 (0x37)	DDB7	DDB6	DDB5	DDB4	DDB3	DDB2	DDB1	DDB0	DDRB
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Figura 4: Bits del registro DDRB. Obtenida de la hoja de datos del fabricante.

Registros GIMSK:

En el registro GIMSK se configura las interrupciones que se habilitan. Se utilizó para configurar las interrupciones de los botones.

GIMSK – General Interrupt Mask Register

Bit	7	6	5	4	3	2	1	0	
0x3B (0x5B)	INT1	INT0	PCIE0	PCIE2	PCIE1	–	–	–	GIMSK
Read/Write	R/W	R/W	R/W	R/W	R/W	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	

Figura 5: Bits del registro GIMSK. Obtenida de la hoja de datos del fabricante.

Desarrollo

FSM

Para resolver este laboratorio, se creó la máquina de estados que se muestra en la siguiente figura.

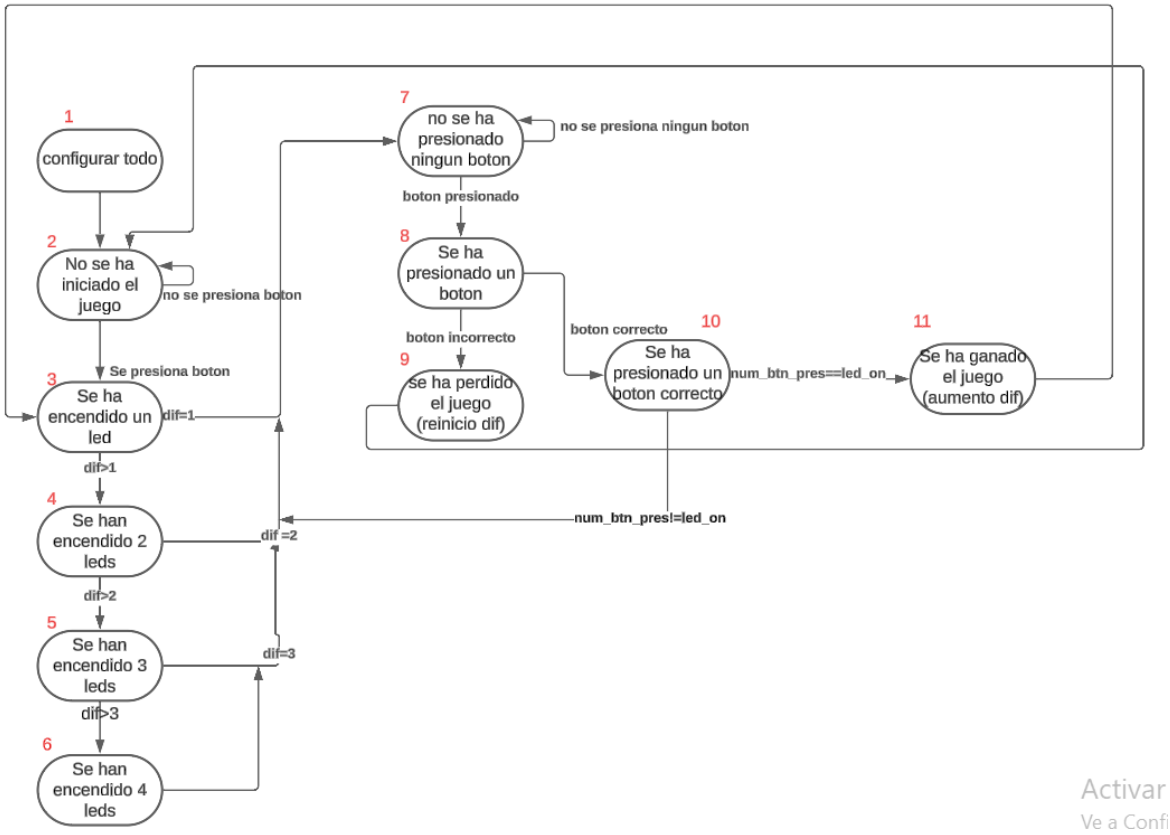

Figura 6: Diagrama de la FSM utilizada para resolver el juego.

Como se observa en la imagen anterior, la maquina de estados comienza en un estado en donde se realizan algunas configuraciones, especialmente sobre los registros de interrupciones y timers.

Luego de este estado, la maquina se queda esperando a que se presione algun botón, y a partir de este punto decidirá si enciende 1, 2, 3 o 4 leds, dependiendo de si se tiene la dificultad establecida en 1, 2, 3 o 4. A partir de la dificultad 4, se encienden todos los leds, pero conforme se aumenta la dificultad, se disminuye el tiempo en el que se mantiene encendido cada led. Luego de esta serie de estados, la maquina llega a un estado en el que espera a que el usuario presione los botones en el orden correcto, en este punto se utiliza un conjunto de 3 estados para decidir si ya se terminó la ronda, si se ganó el juego, si se perdió el juego, o si se debe seguir esperando a que se presione el siguiente botón.

Codigo utilizado

A continuación, se mostrarán algunas imágenes del codigo que se utilizó. Debido a la extensión de este (303 líneas) solo se mostrarán las secciones mas relevantes para este laboratorio.

En la siguiente figura se muestra una sección del codigo que corresponde a la ISR que se activa cuando se presiona un botón. Se puede ver que lo primero que se hace es, de hecho, desactivar las interrupciones para evitar que estas se sigan activando mientras se mantiene la pulsación actual. Mediante las interrupciones del timer estás interrupciones se vuelven a activar luego de algunos milisegundos, que sea lo suficiente para que el botón se deje de presionar. Luego se utiliza una serie de instrucciones para calcular si el botón que se presionó corresponde al botón que se debía presionar para continuar el juego.

```
258 ISR (PCINT2_vect){
259     algun_boton_presionado=1;
260     GIMSK = 0X00; //Deshabilitar las interrupciones temporalmente
261     pinD3_actual= PIND &(1<<PIND3);
262     pinD4_actual= PIND &(1<<PIND4);
263     pinD5_actual= PIND &(1<<PIND5);
264     pinD6_actual= PIND &(1<<PIND6);
265 
266     if (estado_actual == No_se_ha_presionado_ningun_boton){
267         num_btn_presionados++;
268         if (pinD3_actual != pinD3_anterior){
269             if (secuencia_de_leds[num_btn_presionados-1]==3){
270                 boton_correcto=1;
271             }
272         }else if(pinD4_actual != pinD4_anterior){
273             if (secuencia_de_leds[num_btn_presionados-1]==4){
274                 boton_correcto=1;
275             }
276         }else if (pinD5_actual != pinD5_anterior){
277             if (secuencia_de_leds[num_btn_presionados-1]==2){
278                 boton_correcto=1;
279             }
280         }else if (pinD6_actual != pinD6_anterior){
281             if (secuencia_de_leds[num_btn_presionados-1]==1){
```

Figura 7: Sección del codigo correspondiente a la ISR que activan los botones.

En la siguiente figura, lo que se muestra es el fragmento del codigo correspondiente a la ISR que activan las interrupciones de los timers. En el estado de configuración, el timer se configura para que genere la interrupción cada vez que se llena el contador, sin embargo, por la alta frecuencia esto es casi inperceptible, por ello se utiliza la variable asistente counter_int que permita llevar un conteo de las veces que se ha levantado una interrupción, y de esta manera poder ejecutar las acciones solo cuando se desee, es decir, apagar los leds solamente cuando pase un tiempo suficiente para poder se observable. Como se muestra en la linea 294, la dificultad del nivel actual, tambien se toma en cuenta a la hora de contar cada cuantas interrupciones se deben tomar acciones para apagar los leds. También se puede observar en la linea 298 que se utiliza otra variable asistente counter_desbloque_de_click para decidir cuándo se deben volver a activar las interrupciones de los pulsadores.

```
291 ISR (TIMER1_COMPA_vect){
292     counter_int++;
293     counter_desbloque_de_clic++;
294     if (counter_int == 60-(dificultad*5)){
295         PORTB= 0x0;
296         counter_int=0;
297     }
298     if (counter_desbloque_de_clic==15){
299         GIMSK = 0X10; // Volver a habilitar las interrupciones PCINT
300         counter_desbloque_de_clic=0;
301     }
302     TCNT1 = 0x0; //establecer el contador del timer en cero.
303 }
```

Figura 8: Fragmento del codigo que corresponde a la ISR que se activa con las interrupciones del timer.

Circuito implementado:

En la siguiente figura se muestra el circuito que se implementó para completar este laboratorio:

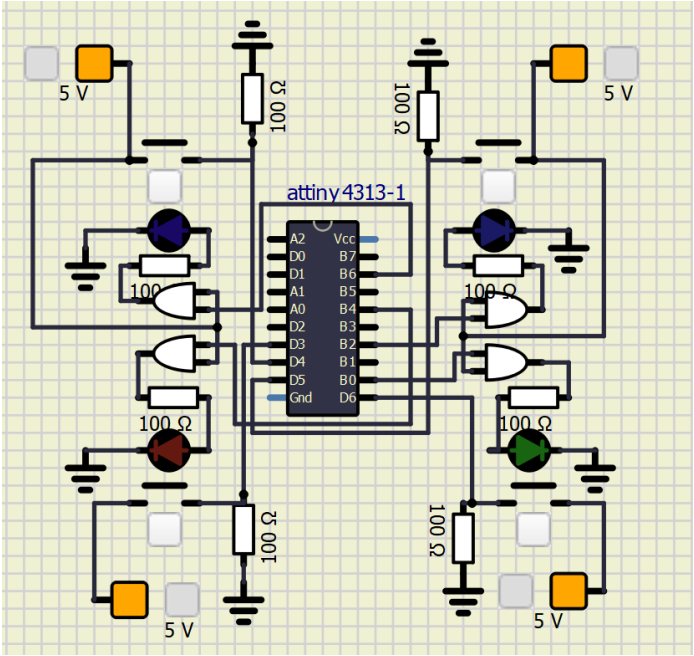

Figura9: Circuito implementado. Fuente: Creación propia.

En la imagen anterior se puede observar que el microcontrolador recibe los datos desde cada uno de los 4 pulsadores, estos pulsadores activaran la ISR correspondiente a la interface D, y dentro de la ISR se revisará de cuál botón proviene el click, y con base en esta información se decidirá si se ha presionado el botón correcto o el incorrecto dependiendo del punto en el que se encuentre el juego. Igualmente, en las terminales B se conectaron los leds, mediante una resistencia para limitar la corriente de cada led, y una compuerta and, que funcione a modo de buffer, para regenerar la tensión y evitar sobrecargar las salidas del microcontrolador

Conclusiones y recomendaciones

-Se logró hacer uso correcto de los timers, para el control del tiempo en que se encendió cada led.

-Se logró implementar las interrupciones para poder notificar al microcontrolador cuando se presionaba un botón.

-Se logró implementar el juego simón dice en su mayoría, aunque, por razones desconocidas, en algunas simulaciones el juego finaliza, aunque se presionen correctamente los botones. Esto puede estar relacionado con los timer o con alguna variable que está provocando un error en el código.

-Es importante cuando se trabaje con microcontroladores y con alguna función en específico, tener claro todos los registros que se ven involucrados en su funcionamiento, pues el desconocimiento sobre alguno de estos puede provocar el malfuncionamiento del circuito implementado.

Bibliografía