Laboratorio de Microprocesadores - 86.07

Trabajo Práctico Obligatorio N°4

Interrupción Externa

Profesor:			Ing. Guillermo Campiglio								
Cuatrimestre/Año:			1°/2020								
Turno de las clases prácticas			Miércoles								
Jefe de trabajos prácticos:			Ing. Pedro Ignacio Martos								
Docente guía:			Ing. Fabricio Baglivo, Ing. Fernando Pucci								
Autores			Seguimiento del proyecto								
Nombre	Apellido	Padrón									
Santiago	López	100566									

Observaciones:

.....
.....
.....
.....
.....
.....
.....
.....
.....

Fecha de aprobación				Firma J.T.P

Coloquio	
Nota final	
Firma profesor	

Índice

1. Objetivo

Hacer manejo de las interrupciones externas del microcontrolador. Esto implica dejar el programa en un punto no determinado previamente, realizar una rutina alternativa, y luego volver al punto de partida.

2. Descripción

Se comenzó el programa inicializando el puerto para el manejo de los LEDs, el puerto para el manejo de las interrupciones y el *stack pointer* para el llamado de las interrupciones y subrutinas.

El contenido de la interrupción consiste en cortar el proceso actual, hacer parpadear un led a una frecuencia de 1Hz 5 veces, y luego reanudar el proceso interrumpido.

Para detectar las interrupciones de utiliza un circuito de pull-down, pero simplemente se podría habilitar la resistencia de pull-up del microcontrolador, tan solo conectando el pulsador entre GND y el pin de la interrupción.

3. Diagrama de conexiones en bloque

Las conexiones siguieron el esquema de la Figura 1.

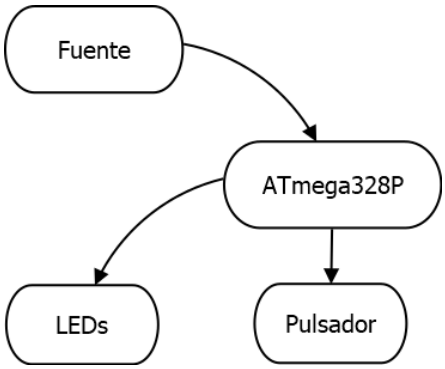

Figura 1: Diagrama de conexiones en bloque.

4. Esquemático

La Figura 2 muestra las conexiones eléctricas efectuadas en el práctico. Los componentes utilizados se encuentran en la sección siguiente. En caso de querer utilizarse la resistencia de *pull-up* interna del micro, se conectaría el puslador entre el pin de la interrupción y tierra, seteando el micro para detectar flancos descendentes en lugar de flancos ascendentes. De esta forma se ahorra el uso del resistor externo de 10kΩ.

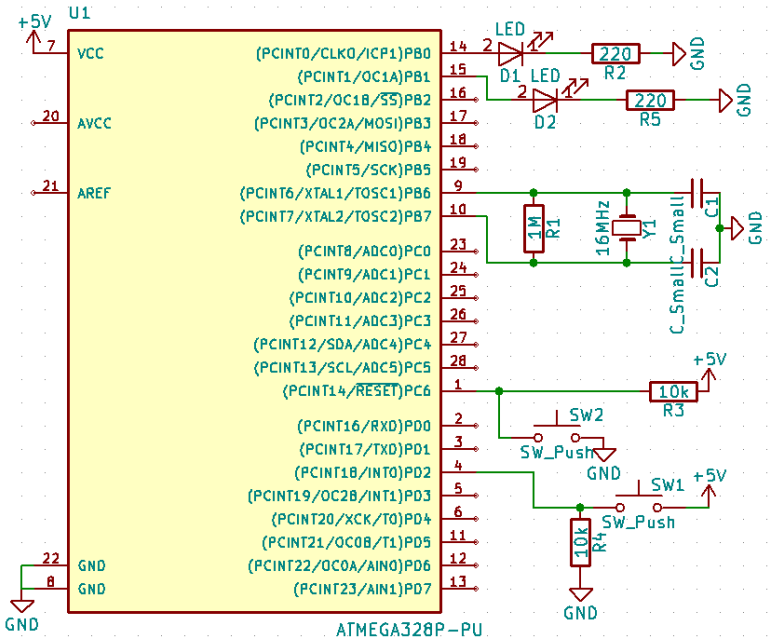

Figura 2: Circuito esquemático.

5. Listado de componentes

Los componentes utilizados fueron los listados a continuación:

- Placa de desarrollo Arduino UNO
- Resistores de 220Ω y 10kΩ
- LEDs de color rojo
- Pulsador
- Protoboard
- Cables unipolares

6. Diagrama de flujo

En la Figura 3 se presentan los pasos a seguir por el programa. Dado el objetivo del práctico, se decidió abstraerse de la implementación del tiempo de espera entre el encendido y apagado del LED durante la interrupción.

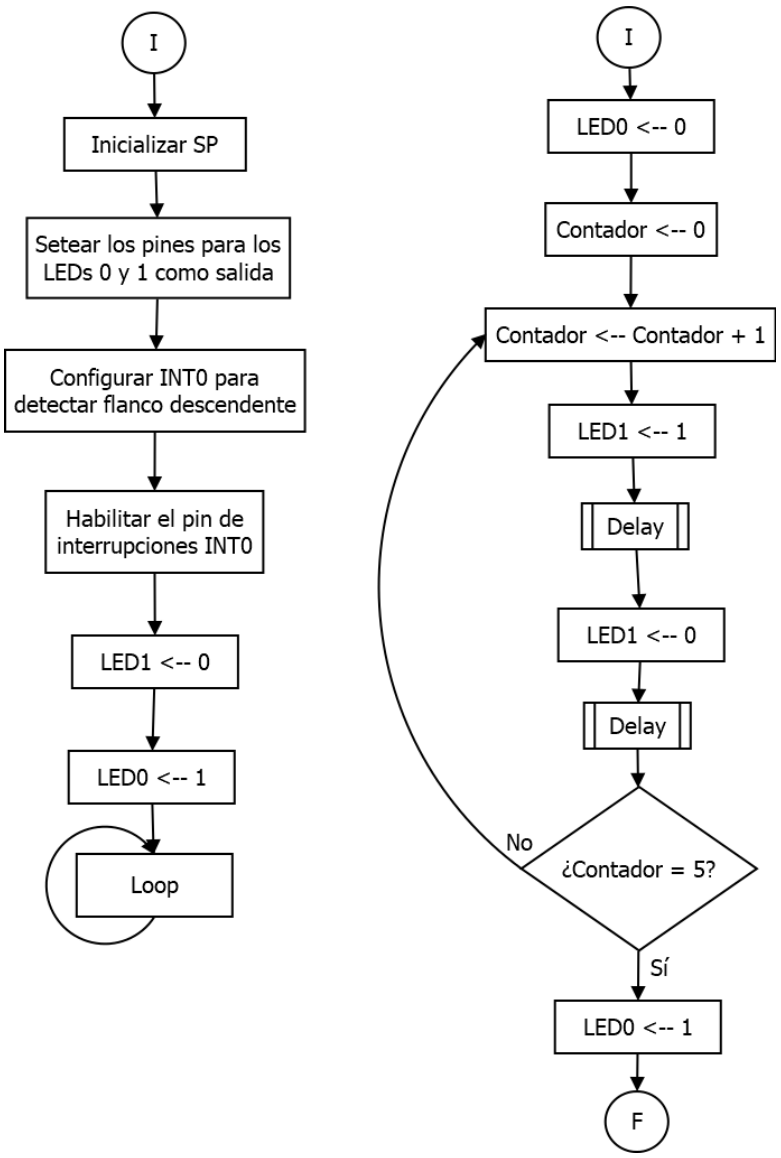

Figura 3: Diagrama de flujo.

7. Código fuente

En el programa se puede ver el uso del stack pointer para poder hacer llamados a subritunas desde la interrupcciones, sin que se pierda a donde volver en el código.

Se decidió detectar las interrupciones por flanco ascendente, ya que no se cuenta con detección de estado alto. En el caso de utilizar la resistencia de pull-up, se pudo haber utilizado la detección de estado bajo, resolviendo ese inconveniente, como también la detección de flanco descendente, como se explicó previamente.

```
 1: .include "m328pdef.inc"
 2:
 3: .def conf = r16
 4: .def count = r17
 5: .def count1 = r18
 6: .def count2 = r19
 7: .def count3 = r20
 8: .equ max_twinkle = 5
 9: .equ int_led = 1
10: .equ main_led = 0
11: .equ led_port = PORTB
12:
13: .macro INIT_SP
14:     ldi @0,low(RAMEND)
15:     out SPL,@0
16:     ldi @0,high(RAMEND)
17:     out SPH,@0
18: .endmacro
19:
20: .macro SETUP_LED_PORT
21:     ldi @0,0x03
22:     out DDRB,@0
23: .endmacro
24:
25: .macro SETUP_INTERRUPT
26:     ldi @0,0x03         ; seteo las interrupciones sobre int0 como flanco ascendente
27:     sts EICRA,@0
28:     ldi @0,(1<<INT0)    ; habilito las interrupciones en el pin INT0
29:     out EIMSK,@0
30:     sei
31: .endmacro
32:
33: .cseg
34: .org 0x0000
35:     jmp main
36:
37: .org INT0addr
38:     jmp isr_int0
39:
40: .org INT_VECTORS_SIZE
41: main:
42:
43: // configuracion de SP
44:     INIT_SP conf
45:
46: // configuro el puerto que maneja los leds
47:     SETUP_LED_PORT conf
48:
49: // configuracion de interrupciones
50:     SETUP_INTERRUPT conf
51:
52:     cbi led_port,int_led
53:     sbi led_port,main_led
54:
55: here:
56:     nop
57:     nop
58:     nop
59:     rjmp here
60:
61: // interrupciones
62: isr_int0:
63:     cbi led_port,main_led
64:
65:     clr count
66:
67: twinkle:
68:     inc count
69:     sbi led_port,int_led
```

```
 70:     call delay
 71:     cbi led_port,int_led
 72:     call delay
 73:     cpi count,max_twinkle
 74:     brlo twinkle
 75:
 76:     sbi led_port,main_led
 77:     reti
 78:
 79: // subrutinas
 80: delay:
 81:     // inicializo los contadores
 82:     clr count1  ;  4 * 255
 83:     clr count2  ; (4 * 255 + 5) * 255
 84:     clr count3  ; (4 * 255 + 5) * 255 + 5) * 32 * 1/f = 0,5s
 85: loop:
 86:     inc count1
 87:     cpi count1,0xff
 88:     brlo loop
 89:
 90:     clr count1
 91:     inc count2
 92:     cpi count2,0xff
 93:     brlo loop
 94:
 95:     clr count2
 96:     inc count3
 97:     cpi count3,0x20
 98:     brlo loop
 99:
100:     ret
```

8. Costos

A continuación se presenta un listado de los costos de los componentes utilizados en el práctico.

- Arduino UNO - $850
- Resistores - $50
- LEDs rojo - $120
- Pulsador - $10
- Protoboard - $240
- Cables unipolares - $150

Sumando un costo total de $1420.

9. Conclusiones

El uso de interrupciones ahorra el tener que leer cambios en entradas, resultando mucho más sencillo, a nivel de código, detectar ciertas señales.