Laboratorio de Microprocesadores - 86.07

Trabajo Práctico Obligatorio N°2

Manejo de puertos

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

Controlar el estado de un LED, el cual se enciende al presionar un pulsador, y se apaga al soltarlo, mediante el manejo de los puertos del microprocesador.

2. Descripción

Para controlar el estado del LED se leyó el valor de entrada del puerto al cual se conectó el pulsador. En el caso de que fuera 1 se seteó el puerto al cual se conectó el LED, y en caso contrario se limpió el valor del puerto.

Una vez escrito el código, se hizo un *testeo* con el uso del software *AVR Studio* para que la respuesta del micro fuera la prevista. Para detectar el estado del pulsador y controlar el estado del LED, se hizo uso del puerto B del microcontrolador.

Luego, se repitió el desarrollo habilitando el resistor de *pull-up* mediante el seteo del bit adecuado, para así utilizar un resistor menos del circuito, a cambio de tan solo unas pocas líneas más de código.

3. Diagrama de conexiones en bloque

Las conexiones siguieron el esquema de la Figura 1.

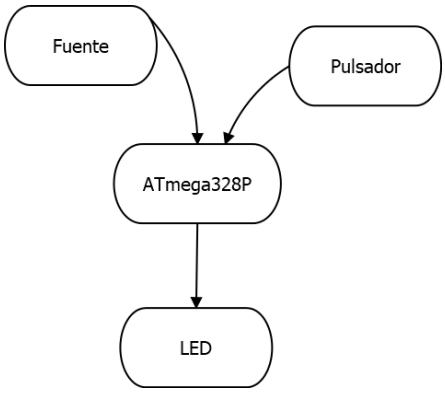
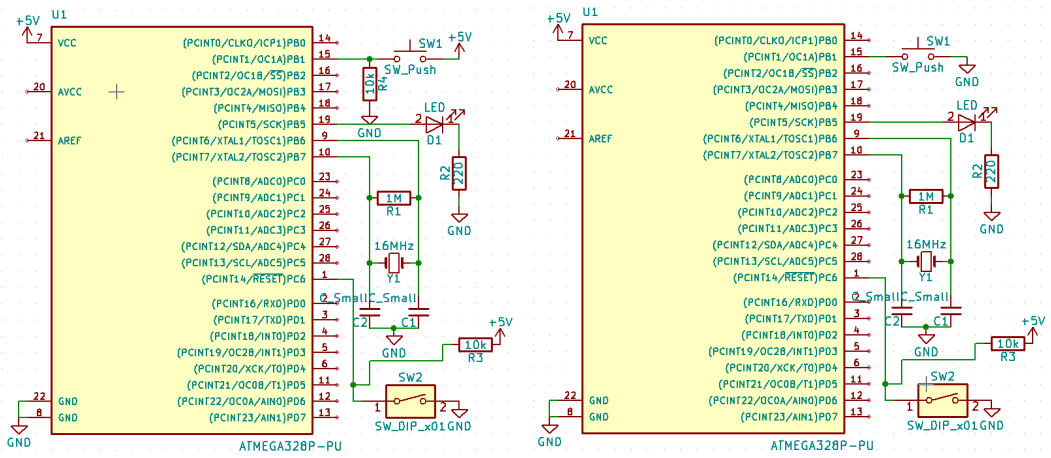


Figura 1: Diagrama de conexiones en bloque.

4. Esquemático

Las Figuras 2a y 2b muestran las conexiones eléctricas efectuadas en el práctico. Los componentes utilizados se encuentran en la sección siguiente.

Debe notarse que en el caso de la Figura 2b se utiliza un resistor de menos, ya que se reemplaza su uso por el resistor de *pull-up* del microcontrolador.



(a) Circuito con resistor de *pull-down*

(b) Circuito con resistor de *pull-up*

Figura 2: Circuito esquemático.

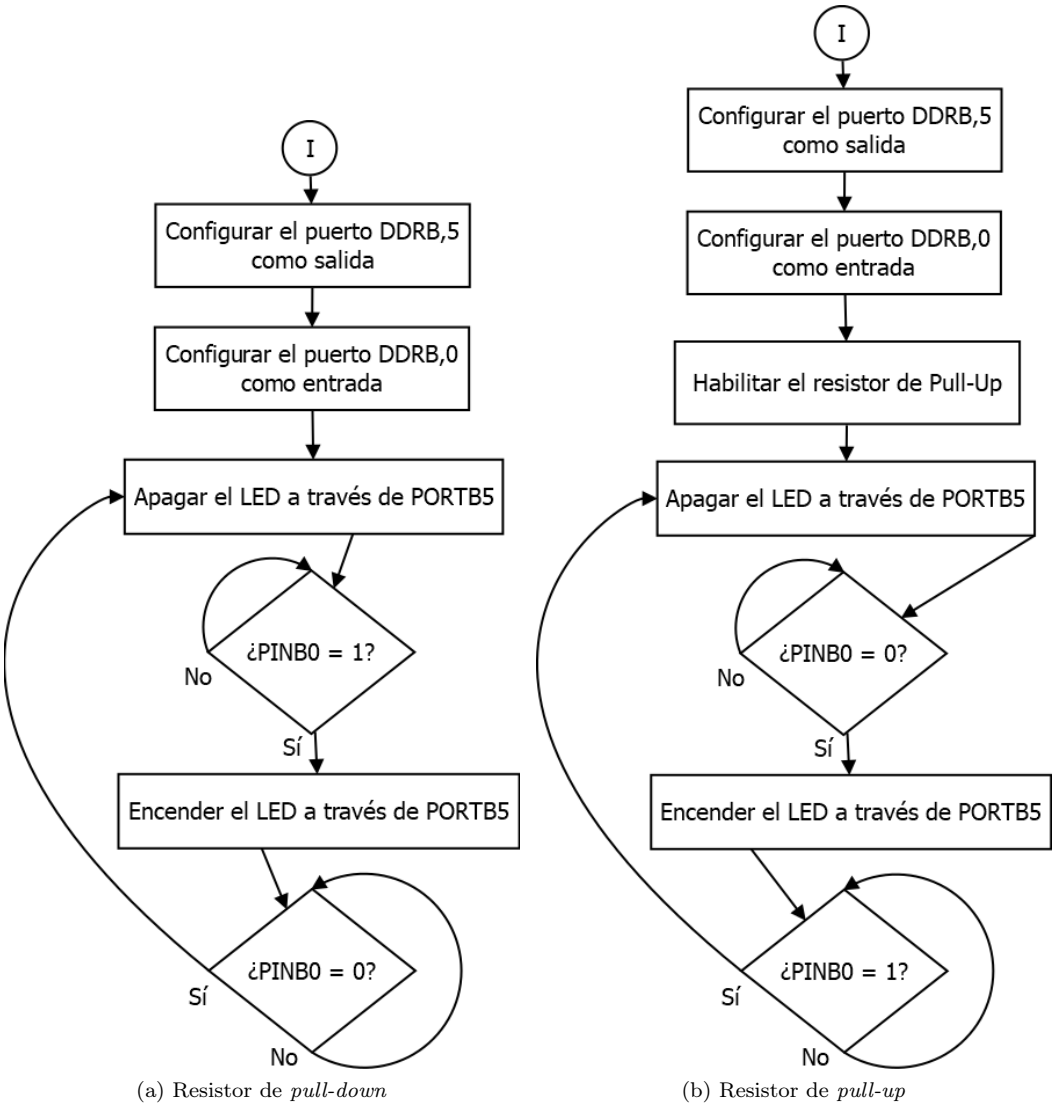
5. Listado de componentes

Los componentes utilizados fueron los listados a continuación:

- Placa de desarrollo Arduino UNO
- Resistor de 220Ω
- Resistor de 10kΩ
- LED de color rojo
- Pulsador
- Protoboard
- Cables unipolares

6. Diagrama de flujo

En las Figuras 3a y 3b se presentan los pasos a seguir por el programa original, y con el resistor de *pull-up* habilitado, respectivamente. Los pasos a seguir pueden parecer similares, pero se remarca la diferencia al leer el puerto de entrada. Ya que el tener el pulsador contra masa o contra V_{CC}, el estado del puerto no corresponde al mismo estado del pulsador en cada caso.



(a) Resistor de *pull-down* (b) Resistor de *pull-up*

Figura 3: Diagrama de flujo.

7. Código fuente

En el primer programa se controla el estado del LED utilizando un resistor de *pull-down*, por lo que el pulsador deja pasar un '1' lógico al ser presionado y un '0' lógico al soltarse. De esa forma se tuvo en cuenta que al leer un '1' en el puerto DDRB0 se debería encender el LED, y apagarlo en caso contrario.

En el segundo programa se puede ver el uso del resistor de *pull-up* interno del microcontrolador, ya que este se habilita vía software. Es así como la lógica del programa cambia, ya que ahora el presionar el pulsador representa un '0' lógico a la entrada, y un '1' lógico al soltarlo.

```
 1: .include "m328pdef.inc"
 2:
 3: .dseg
 4:
 5: .EQU IN_PORT = PINB
 6: .EQU OUT_PORT = PORTB
 7: .EQU PULSE_PIN = 0
 8: .EQU LED_PIN = 5
 9:
10: .cseg
11: .org 0x0000
12:     jmp main
13:
14: .org INT_VECTORS_SIZE
15: main:
16:     ldi R20,(0<<PULSE_PIN | 1<<LED_PIN)
17:     out DDRB,R20
18:
19:
20: led_off:
21:     cbi OUT_PORT,LED_PIN    ; apagar led
22:
23: no_pulse:
24:     sbis IN_PORT,PULSE_PIN  ; si se apreta el boton va a prender el led
25:     rjmp no_pulse
26:
27: led_on:
28:     sbi OUT_PORT,LED_PIN    ; prender led
29:
30: pulse:
31:     sbic IN_PORT,PULSE_PIN  ; si se suelta el boton va a apagar el led
32:     rjmp pulse
33:
34:     rjmp led_off
```

```
 1: .include "m328pdef.inc"
 2:
 3: .dseg
 4:
 5: .EQU IN_PORT = PINB
 6: .EQU OUT_PORT = PORTB
 7: .EQU PULSE_PIN = 0
 8: .EQU LED_PIN = 5
 9:
10: .cseg
11: .org 0x0000
12:     jmp main
13:
14: .org INT_VECTORS_SIZE
15: main:
16:     ldi R20,(0<<PULSE_PIN | 1<<LED_PIN)
17:     out DDRB,R20
18:
19:     sbi OUT_PORT,PULSE_PIN  ; pull-up enabled
20:
21:
22: led_on:
23:     sbi OUT_PORT,LED_PIN    ; prender led
24:
25: no_pulse:
26:     sbic IN_PORT,PULSE_PIN  ; si se apreta el boton va a apagar el led
27:     rjmp no_pulse
28:
29:
30: led_off:
31:     cbi OUT_PORT,LED_PIN    ; apagar led
32:
33: pulse:
34:     sbis IN_PORT,PULSE_PIN  ; si se suelta el boton va a prender el led
35:     rjmp pulse
36:
37:     rjmp led_on
```

8. Costos

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

- Arduino UNO - $850
- Resistores - $50
- LED rojo - $20
- Pulsador - $10
- Protoboard - $240
- Cables unipolares - $150

Sumando un costo total de $1320.

9. Conclusiones

El uso del resistor de *pull-up* resultó eficiente a la hora de armar el circuito, además de abaratar el costo de este. El precio a pagar fue programar algunas líneas más de código y pensar una lógica alternativa al problema. Se estima que a gran escala el uso de este resistor podría resultar beneficioso, aunque en este caso, dada la poca cantidad de líneas de código escritas, el costo fue prácticamente empezar todo de 0.