FACULTAD DE INGENIERIA

Universidad de Buenos Aires

Laboratorio de Microprocesadores - 86.07

Trabajo Práctico 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:			-							
Autor			Seguimiento del proyecto							
Francisco	Rossi	99540								

Observaciones:

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

Fecha de aprobación				Firma J.T.P

Coloquio	
Nota final	
Firma profesor	

Índice

1. Introducción

En el siguiente informe se explica el diseño de dos programas escritos en lenguaje **Assembler** para copiar el valor de entrada de un pin al valor de salida de un puerto en el microcontrolador **ATMEGA328p**, particularmente, al pulsar un pulsador, valga la redundancia conectado a *PB0*, imponer un valor alto en *PB1* de manera que un LED conectado a *PB1* se encienda. Se realizará de dos maneras distintas.

a. Utilizando un resistor externo.

b. Utilizando la resistencia de pull-up interna del puerto

1.1. Objetivo

El objetivo es utilizar los resgistros de los puertos como entrada y salida. Y entender la utilidad de la resistencia de pull-up interna.

1.2. Descripción

Se conecta un LED con un resistor en serie al pin *PB1*, para controlar el encendido y apagado del mismo mediante un pulsador conecto al pin *PB0* mediante las instrucciones de Assembler correspondientes a los casos (a) y (b). En el caso (a) se utilizará un resistor externo y en el caso (b) se modificará el código para utilizar la resistencia de pull-up interna del puerto *PB0*.

2. Materiales

Se utilizaron los siguiente materiales para el proyecto:

a. 1 LED Blanco

b. 1 Resistor de 220 Ω

c. 1 Resistor de 10 $k\Omega$

d. 1 Pulsador.

e. 1 Microcontrolador ATmega328p (Utilizando el integrado con el Arduino Uno)

3. Diagrama en Bloques

En las **Fig. 1** y 2 se muestran los diagrama en bloques de cada circuito.

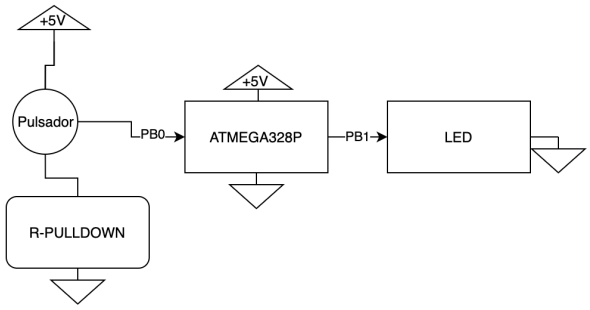

Figura 1: Diagrama en bloques en el caso (a).

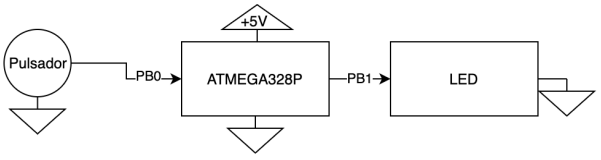

Figura 2: Diagrama en bloques en el caso (b).

4. Esquemático

En las **Fig. 3** y **Fig. 4** se muestra el esquemático del circuito en cada caso.

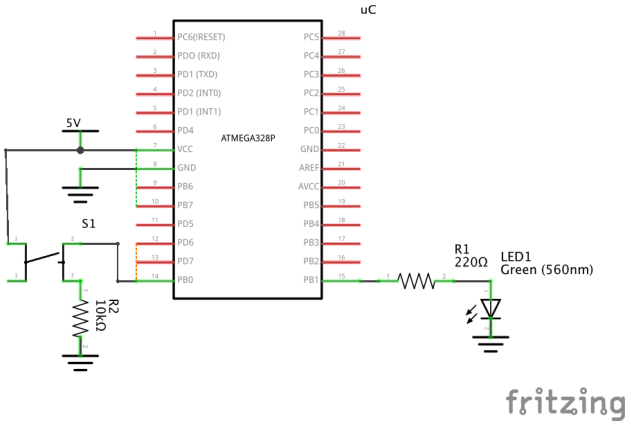

Figura 3: Esquemático del circuito implementado usando una R externa.

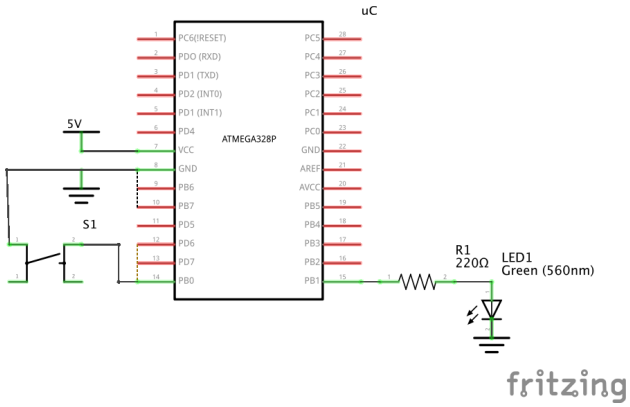

Figura 4: Esquemático del circuito implementado usando la R interna del puerto.

En las **Fig. 5** y **Fig. 6** se muestra el circuito implementado en cada caso.

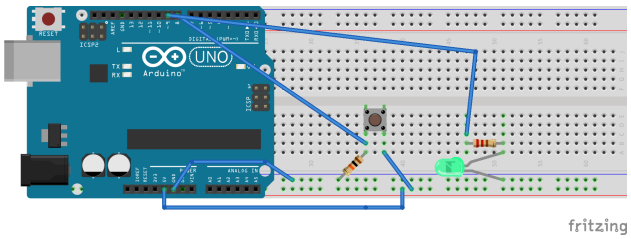

Figura 5: Circuito implementado usando una R externa.

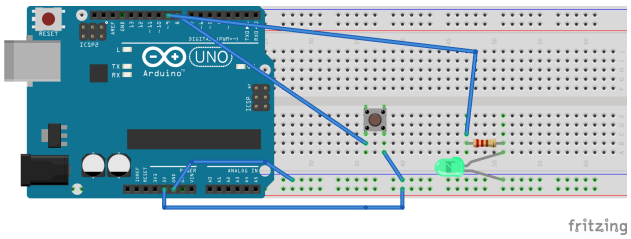

Figura 6: Circuito implementado usando la R interna del puerto.

En ambos casos también estan en simultaneo los circuitos de reset y de clock externo como se muestra en la **Fig. 7**.

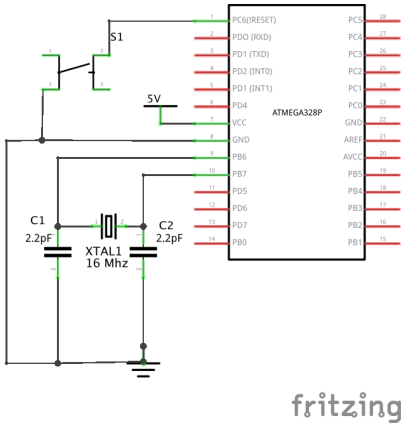

Figura 7: Circuito de reset y clock externo

5. Diagrama de flujo

En las **Fig. 8** y **Fig. 9** se muestran los diagramas de flujo del programa (a) y (b) respectivamente.

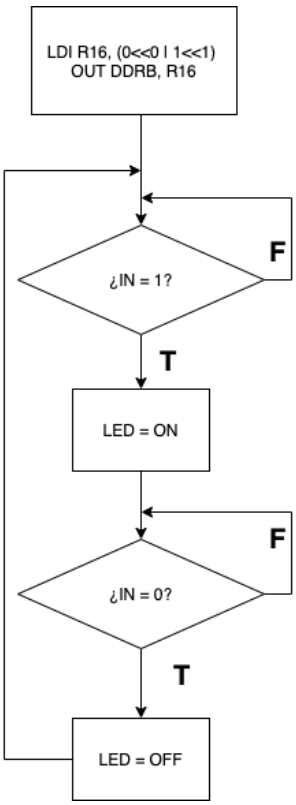

Figura 8: Diagrama de flujo en el caso (a).

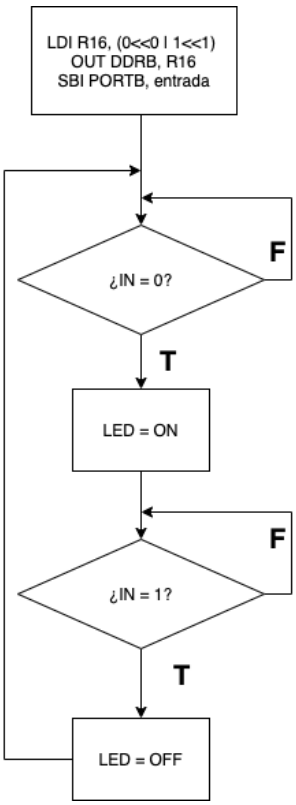

Figura 9: Diagrama de flujo en el caso (b).

6. Códigos

Ambos códigos comparten que basado en el valor de *PB0* se determina el valor de *PB1*, en (a) el resistor externo actua como un resistor de pull-down, es decir que al pulsar el botón se impone un uno lógico en *PB0*. El programa lee este valor y lo copia a *PB1* encendiendo el LED, cuando se suelta el pulsador el valor de *PB0* es cero, en definitiva el valor que se lee, se copia. Que puede describirse como $PB1 = PB0$.

En cambio, el código (b) al utilizar la resistencia de pull-up interna de *PB0* la lógica es negada, al pulsar el botón impongo un valor bajo en *PB0*, pero como quiero encender el LED al pulsar el botón debo invertir la lógica para imponer un uno lógico en *PB1* cuando *PB0* esta en estado bajo. Copio el valor negado. Que puede describirse como $PB1 = \overline{PB0}$.

6.1. a) Utilizando un resistor externo

```
1  .include "m328pdef.inc"
2
3  .equ entrada = 0
4  .equ salida = 1
5
6  .cseg
7  .org 0x0000
8        JMP main
9
10 .org INT_VECTORS_SIZE
11
12 main:
13
14       LDI R16, (0<<0 | 1<<1) ; Configuro PB0 como entrada y PB1 como salida.
15       OUT DDRB, R16
16
17 bajo:
18       SBIS PINB,entrada ; mientras el estado de PB0 sea bajo quedara en loop.
19       JMP bajo
20       SBI PORTB,salida ; si el valor de entrada es alto se impone un valor de 1 logico en la
     salida.
21 alto:
22       SBIC PINB,entrada ; mientras el estado de PB0 sea alto quedara en loop.
23       JMP alto
24       CBI PORTB,salida ; si el valor de entrada es alto se impone un valor de 0 logico en la
     salida.
25       JMP bajo  ; reinicio el ciclo
```

6.2. b) Utilizando la resistencia de pull-up interna del puerto

```
.include "m328pdef.inc"

.equ entrada = 0
.equ salida = 1

.cseg
.org 0x0000
      JMP main

.org INT_VECTORS_SIZE

main:

      LDI R16, (0<<0 | 1<<1) ; Configuro PB0 como entrada y PB1 como salida.
      OUT DDRB, R16
      SBI PORTB, entrada ; habilito la R de pullup interna del puerto de entrada.

bajo:
      SBIC PINB,entrada ; mientras el estado de PB0 sea alto (no presiono el bot n) quedara en
    loop.
      JMP bajo
      SBI PORTB,salida ; cuando detecto un estado en bajo de la entrada (presiono el bot n)
    prendo el led de salida.

alto:
      SBIS PINB,entrada ; mientras el estado de PB0 sea bajo (el bot n est  presionado) quedara
     en loop.
      JMP alto
      CBI PORTB,salida ; cuando detecto un estado en alto de la entrada (suelto el bot n) apago
    el led de salida.
      JMP bajo ; reinicio el ciclo

```

Es valioso observar que los únicos cambios en uno y otro son el de habilitar la red de pull-up interna y negar la lógica cambiando SBI por CBI y viceversa.

7. Resultados

Se logró diseñar dos programas para el microcontrolador ATMEGA328P que controlen el encendido y apagado de un LED colocado en un puerto del arduino a partir de un pulsador conectado en otro puerto del arduino. Se realizó en el primer caso utilizando una resistencia externa de 10 k y en el segundo caso, nos ahorramos este componente utilizando la resistencia de pull-up interna del puerto y negando la lógica del código anterior.

8. Conclusiones

Se logró controlar el valor de un pin a partir del valor en otro, pudiendo ahorrar un componente (el resistor de pull-down externo) para realizar esto a partir de una modificación en el código. Esto es valioso en proyectos donde en vez de ahorrarnos un solo resistor podemos ahorrarnos miles, utilizando en cambio la resistencia de pull-up interna de cada puerto para imponer un cero o un uno.