



(6609) Laboratorio de Microcomputadoras

Proyecto:
(*tp2 pulsador*)

Profesor:	Ing. Guillermo Campiglio
Cuatrimestre / Año:	1ro/2020
Turno de clases prácticas:	Miércoles
Jefe de Trabajos Prácticos:	Pedro Martos
Docente guía:	

Autores			Seguimiento del proyecto									
Nombre	Apellido	Padrón										
Cristian	Simonelli	87879										

Observaciones:

Fecha de aprobación

Firma J.T.P.

COLOQUIO	
Nota final	
Firma Profesor	

Objetivo:	2
Desarrollo.	2
Resistencia de pullup.	2
Lectura de puerto.	3
Listado de componentes:	3
Diagrama en bloques:	4
Circuito esquemático:	4
Diagrama de flujos:	5
Código: Con resistencia de pull up interna desactivada.	6
Qué cambios habría que hacer si no hubiera resistencia de pull up?	7
Resultado:	9
Conclusiones:	9

Objetivo:

El objetivo del trabajo práctico es prender y apagar un led dependiendo del estado de un pin del microcontrolador. Dicho pin cambia de estado según se pulse o no un switch.

Para realizar lo antedicho, es necesario manejar puertos digitales (entrada salida) y resistencias de pull up.

Desarrollo.

Se medirá el estado de un solo pin del microcontrolador.

Es necesario usar los 3 registros que provee avr.

PORTX, DDRX, PINX.

DDRX es el encargado de la configuración como entrada o salida. Si determinado bit de este puerto se encuentra en estado alto, el pin correspondiente será de salida y viceversa.

PINX es el registro de lectura de entrada, solo se usa para leer el estado real del pin.

PORTX este registro se usa para setear estado en determinado pin, si el mismo estaba configurado como salida (por DDRX).

Sin embargo, si el pin estaba seteado como entrada, este indica si se activa (HIGH) o no (LOW) la resistencia interna de pull up.

Resistencia de pullup.

La resistencia de pull up fija un estado default en el caso en el cual el dispositivo conectado al pin del microcontrolador no defina uno de los estados y el pin “quede al aire”.

Si no se puede asegurar un estado lógico estable asociado al estado físico (por decir de alguna manera a la tensión asignada al pin) el comportamiento del programa puede ser indefinido.

En el avr la resistencia de pull up se encarga de fijar un estado alto ante una eventual no conexión del pin.

En nuestro caso el switch fija el estado bajo cuando es pulsado, conectado grnd, pero no hace nada cuando no se pulsa. En ese caso el pin no tiene un estado definido.

La resistencia de pull up conectada a 5v fija en este caso el estado alto.

El programa identifica el estado alto como “no presionado” y apaga el led.

Lectura de puerto.

Una vez que se puede asegurar el valor leído, hay que tener en cuenta que la instrucción IN lee todo el byte.

Para recuperar el valor del pin específico se hace un and con una máscara que representa el valor del pin (00000001b en el ejemplo). Si el resultado del and es 0 quiere decir que se lee estado bajo.

Por qué no hacer cpi contra 00000001? porque si por alguna razón el puerto tuviera algun otro pin en estado alto eso daría falso. Incluso en este caso el bit5 puede estar en alto si el led está prendido.

En este trabajo práctico no hace falta, pero si tuviéramos que recuperar el valor de más de un bit, por ejemplo 3 bits en cualquier posición del byte, habría que hacer un shift.

Supongamos que 3 bits del PORTB representa un valor de los 8 posibles.

PORTB 00101000.

Tenemos que recuperar ese 101.

La op seria:

$(\text{PORTB} \text{ and } 00111000) \gg 3$

(\gg) es el shift sin carry.

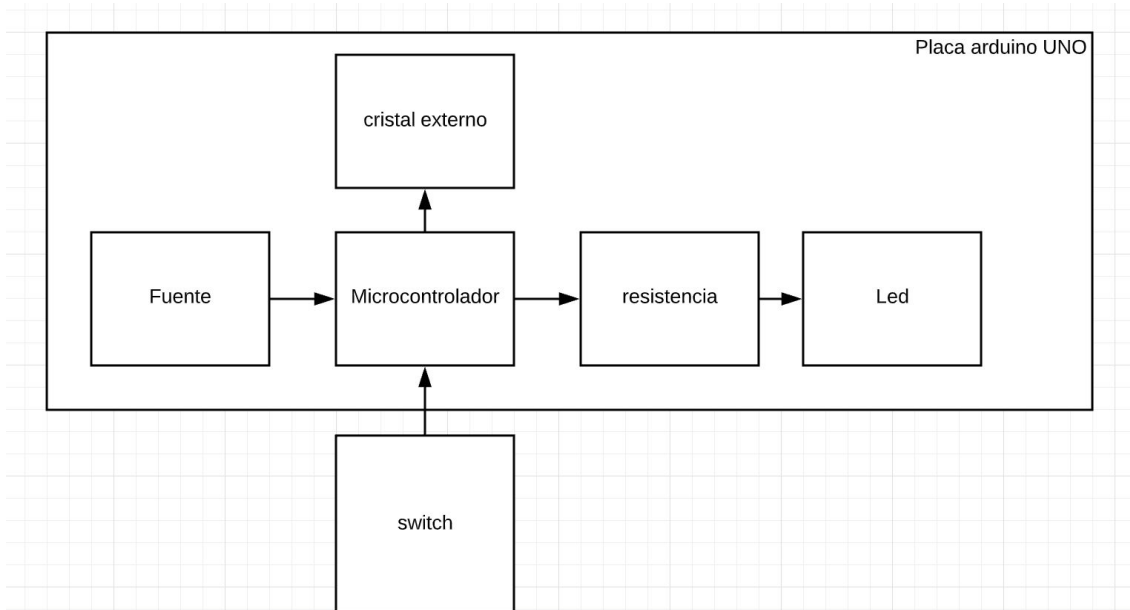
A efectos de este tp solo importa si es 0 o no.

Listado de componentes:

Placa arduino UNO Atmega 328p \$659 aprox 10 usd.

Cable. (emula el switch)

Diagrama en bloques:



Circuito esquemático:

(en el circuito implementado se usa la resistencia de pullup interna del atmega328p)

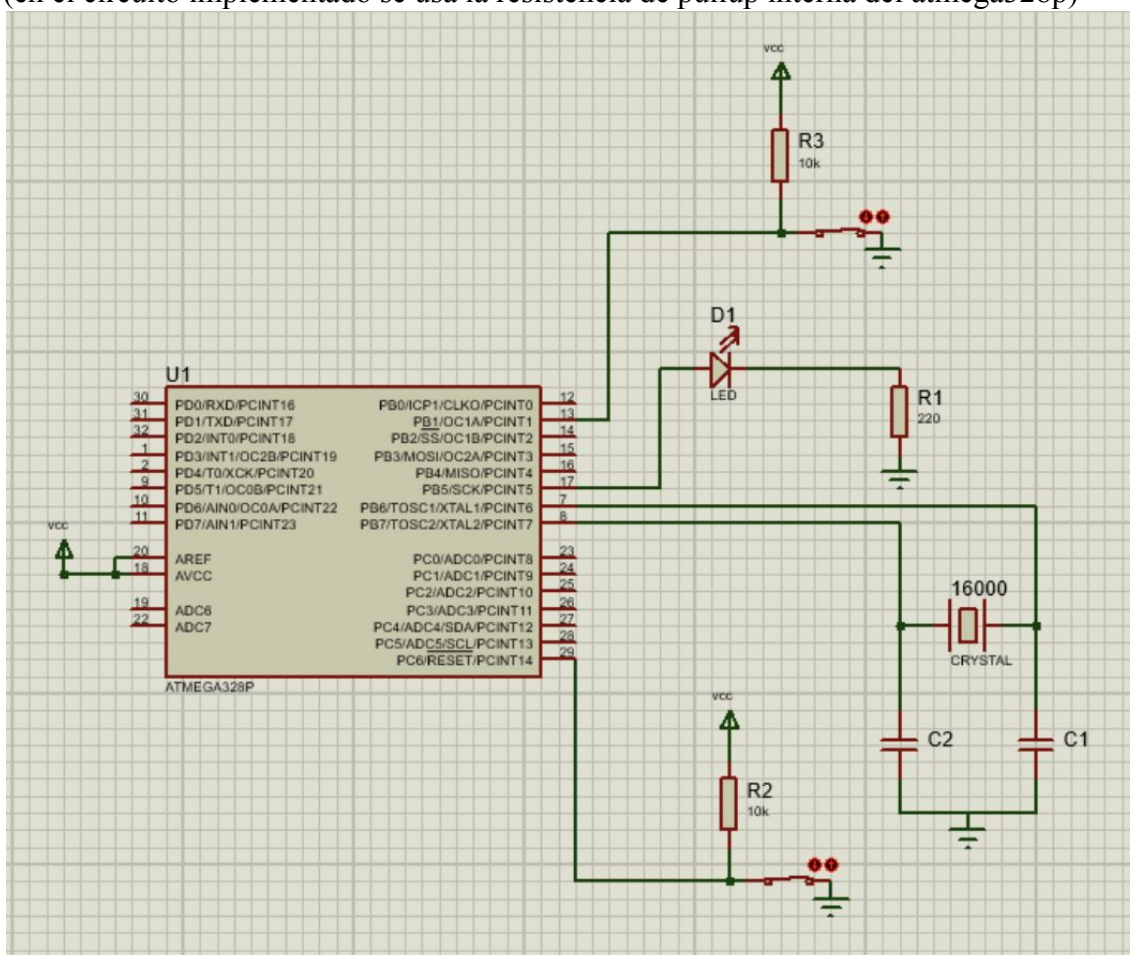
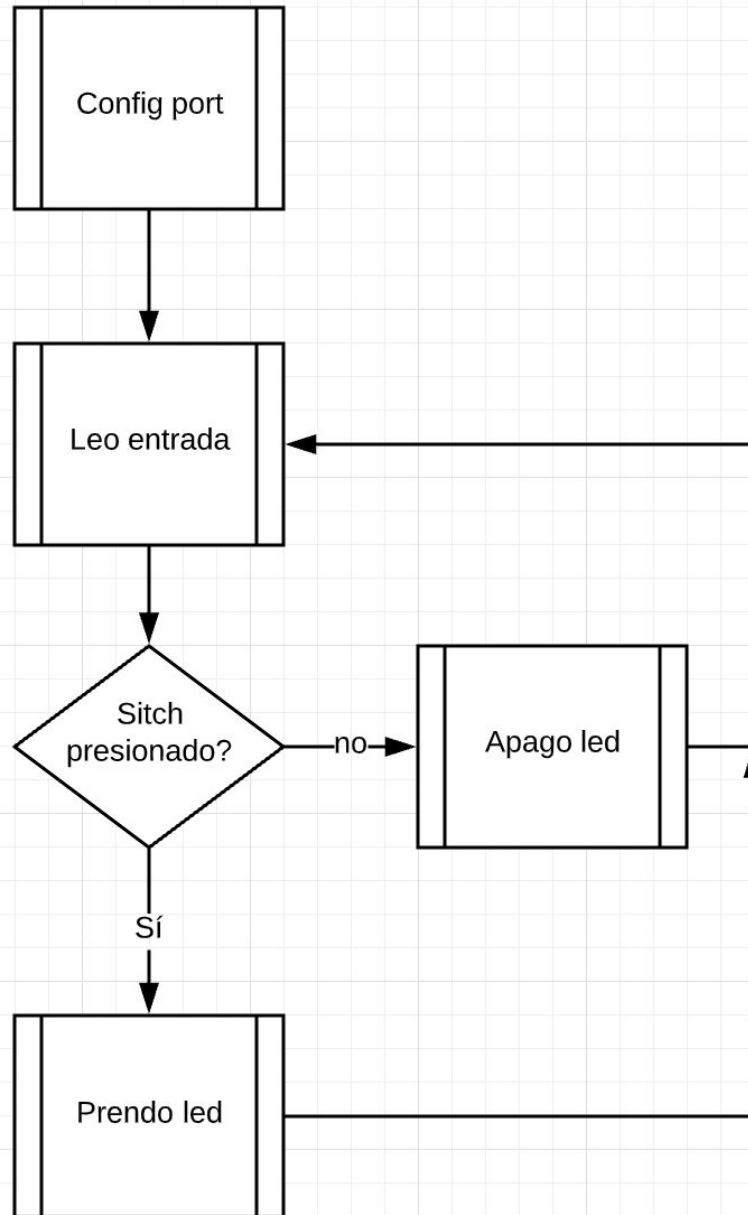


Diagrama de flujos:



Código: Con resistencia de pull up interna desactivada.

```
.include "m328pdef.inc"                ; Valid definitions to 238p

.equ pin_led = 5                        ; the built in led is the pin 13
(5th pin in B port)

.equ pin_button = 0x01                  ; pin connected to the input switch
.equ portb_conf = 0x20

.org 0x0000                             ; The next instruction has to be
written to add 0x0000

                rjmp     main            ; Relative jump to main
.org INT_VECTORS_SIZE                    ; inter vector

main:

                ldi      r20, HIGH(RAMEND) ; Load r20 with the last ram address
higher byte

                out      sph, r20         ; Load higher byte in sp with r20

                ldi      r20, LOW(RAMEND) ; Load r20 with the last ram address
lower byte

                out      spl, r20         ; Load lower byte in sp with r20

                ldi      r20, portb_conf  ; Set port b conf
                out      DDRB, r20        ; Set potb b conf

                ldi      r21, pin_button  ; Set r21 a mask in order to read
only one bit

read:           in       r20, PINB        ; Read portb
                and      r20, r21         ; and with the mask
                breq     led_on           ; if z flag is setted the port was
low, so the switch is pushed

                cbi      PORTB, pin_led   ; led off
                jmp      read             ; goto read

led_on:         sbi      PORTB, pin_led   ; red on
                jmp      read             ; goto read
```

Qué cambios habría que hacer si no hubiera resistencia de pull up?

El siguiente es el código con resistencia de pull up interna:

```
.include "m328pdef.inc"                ; Valid definitions to 238p

.equ pin_led = 5                        ; the built in led is the pin 13
(5th pin in B port)
.equ pin_button = 0x01                  ; pin connected to the input switch
.equ portb_conf = 0x20

.org 0x000                              ; The next instruction has to be
written to add 0x0000

                rjmp      main           ; Relative jump to main
.org INT_VECTORS_SIZE                    ; inter vector

main:
                ldi        r20, HIGH(RAMEND) ; Load r20 with the last ram address
higher byte
                out        sph, r20        ; Load higher byte in sp with r20
                ldi        r20, LOW(RAMEND) ; Load r20 with the last ram address
lower byte
                out        spl, r20        ; Load lower byte in sp with r20

                ldi        r20, portb_conf ; Set port b conf
                out        DDRB, r20       ; Set portb b conf
                ldi        r20, pin_button ; Set pullup resistor for the input
pin
                out        PORTB, r20      ; Set pullup resistor for the input
pin

                ldi        r21, pin_button ; Set r21 a mask in order to read
only one bit
read:          in         r20, PINB        ; Read portb
                and        r20, r21        ; and with the mask
                breq       led_on          ; if z flag is setted the port was
low, so the switch is pushed

                cbi        PORTB, pin_led ; led off
                jmp        read            ; goto read
led_on:        sbi        PORTB, pin_led   ; led on
                jmp        read            ; goto read
```


En cuanto a hardware, se quita la resistencia que está en el mismo pin que el switch, por lo tanto se ahorra un componente.

Resultado:

Se logró controlar el encendido del led con el uso de un switch externo.

Conclusiones:

El microcontrolador avr 328p provee de resistencias de pull up y 3 registros para el control de puertos. Es necesario tener bien en claro el papel que cumple cada uno, más que nada PORTx.

Seguramente una implementación con interrupciones sería mas optima.