

(6609) Laboratorio de Microcomputadoras

Proyecto:
(tp4 interrupción externa)

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
Registros para el uso de interrupciones externas	2
Registro de control de interrupciones externas:	2
Máscara de interrupción externa .	2
SEI.	2
Listado de componentes:	3
Diagrama en bloques:	4
Circuito esquemático:	5
Diagrama de flujos:	6
Código:	7
Resultado:	8
Conclusiones:	8

Objetivo:

El objetivo del trabajo práctico es capturar una interrupción externa y a partir de la misma prender o apagar leds.

Desarrollo.

Se utiliza la interrupción INT0 del Atmega328p, por flanco.

INT0 Se encuentra en PD2. (pin 9 de arduino Uno).

Cuando se detecta el flanco se llama a la ISR correspondiente, en este caso INT0addr.

En la rutina de interrupción de llama a la función delay del tp para titilar el led.

Registros para el uso de interrupciones externas

Registro de control de interrupciones externas:

En este registro se elige la forma de activación de la interrupción.

En este caso se utiliza interrupción por flanco, pero podría ser por cambio de valor.

The external interrupt control register A contains control bits for interrupt sense control.

Bit	7	6	5	4	3	2	1	0	
(0x69)	—	—	—	—	ISC11	ISC10	ISC01	ISC00	EICRA
Read/Write	R	R	R	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Table 12-2. Interrupt 0 Sense Control

ISC01	ISC00	Description
0	0	The low level of INT0 generates an interrupt request.
0	1	Any logical change on INT0 generates an interrupt request.
1	0	The falling edge of INT0 generates an interrupt request.
1	1	The rising edge of INT0 generates an interrupt request.

Máscara de interrupción externa .

Bit	7	6	5	4	3	2	1	0	
0x1D (0x3D)	—	—	—	—	—	—	INT1	INT0	EIMSK
Read/Write	R	R	R	R	R	R	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Se pone el 1 el bit correspondiente a la interrupción que se va a utilizar. En este caso bit 0.

SEL.

Habilita interrupciones globales, setea el bit de interrupciones en el status reg. Ambas, La global interrupt flag y el correspondiente bit en EIMSK tiene que estar activados para que se produzca la interrupción.

108.1. Description

Sets the Global Interrupt Flag (I) in SREG (Status Register). The instruction following **SEI** will be executed before any pending interrupts.

Operation:

(i) $I \leftarrow 1$

Syntax:

Operands:

Program Counter:

(i) **SEI**

None

$PC \leftarrow PC + 1$

16-bit Opcode:

1001	0100	0111	1000
------	------	------	------

Listado de componentes:

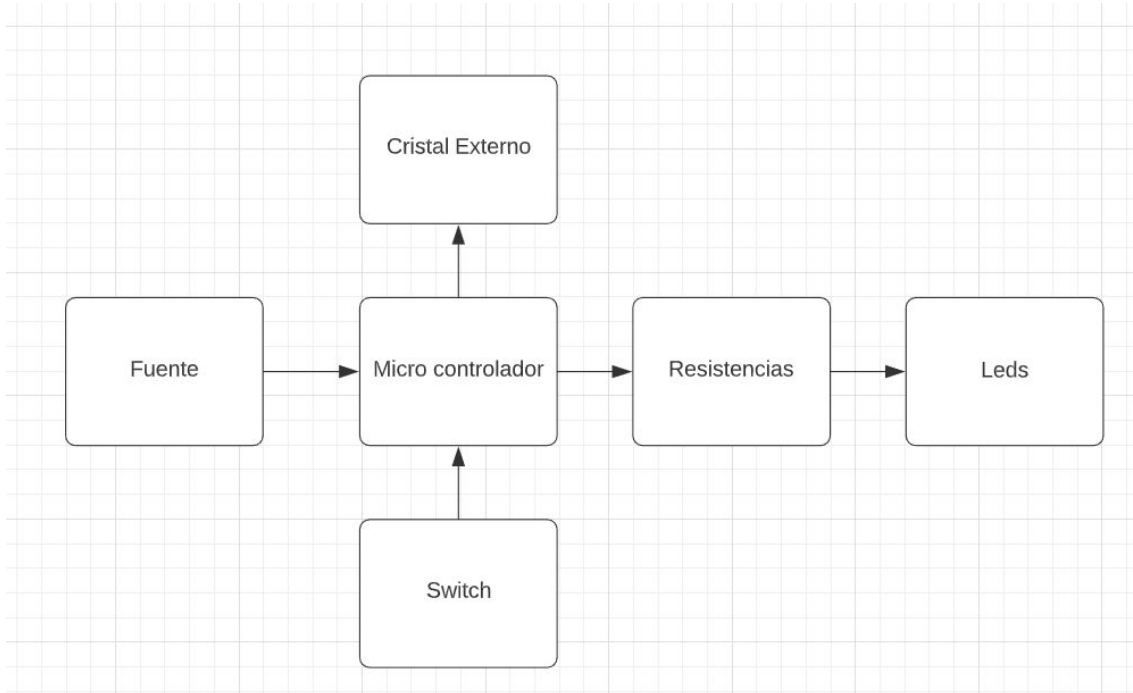
Placa arduino UNO Atmega 328p \$659 aprox 10 usd.

2 led (pack de 10) \$70.

2 resistencias 220 ohm 1/8w 1% \$50.

1 switch \$10.

Diagrama en bloques:



Circuito esquemático (con resistencia de pull up):

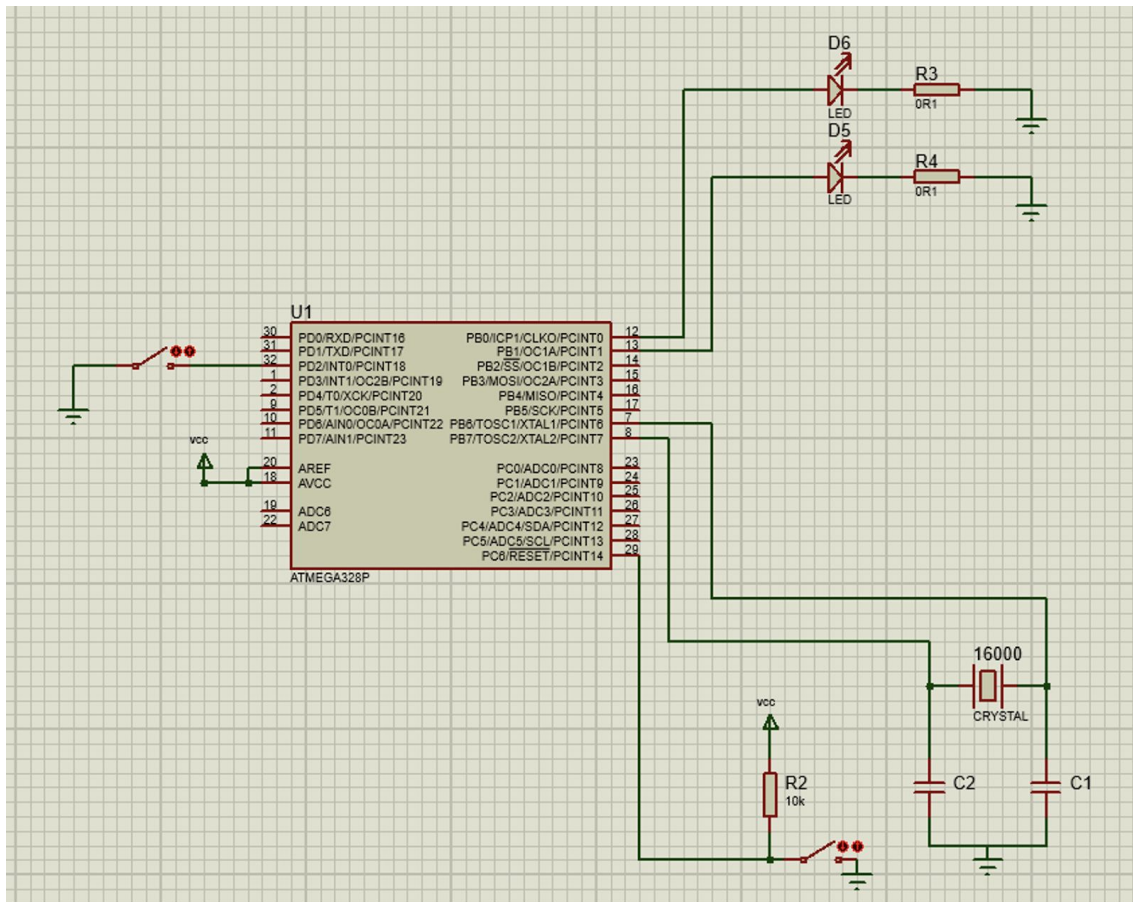
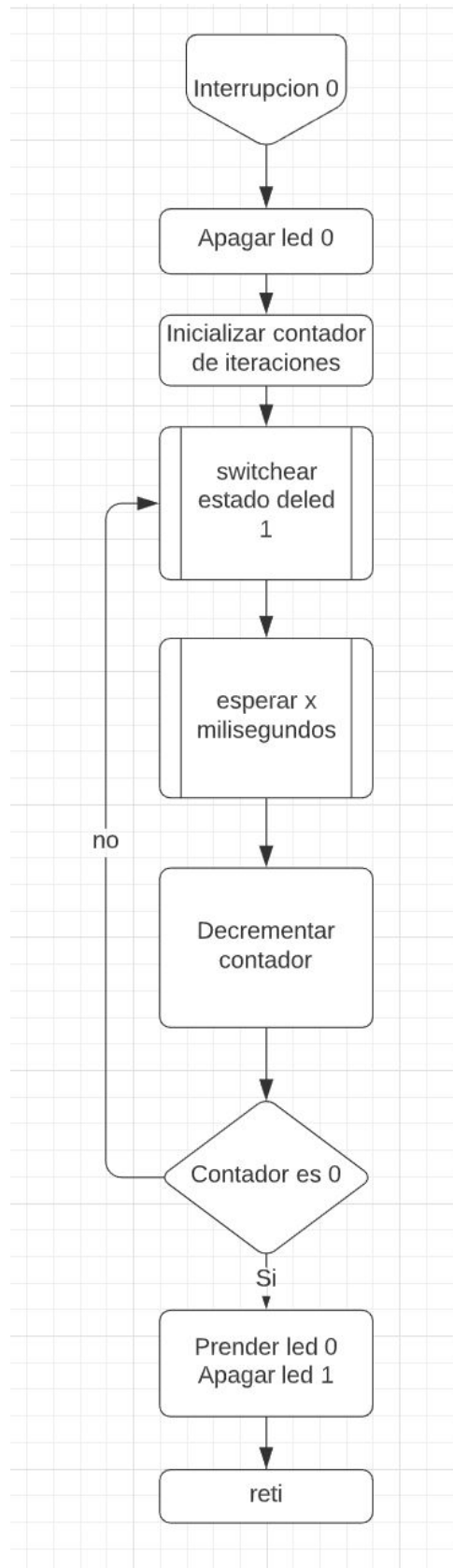


Diagrama de flujos:



Código:

```
;
; ejercicio 4
;
; Author : cristian Simonelli
;

.include "m328pdef.inc"

.def dummyreg = r21

.cseg
.org 0x0000
        jmp      configuracion
.org INT0addr
        jmp      isr_int0

.org INT_VECTORS_SIZE
configuracion:
; Inicializacion stack pointer
        ldi      dummyreg, low(RAMEND)
        out      spl, dummyreg
        ldi      dummyreg, high(RAMEND)
        out      sph, dummyreg

        ldi      dummyreg, 0xfb      ; Port D 2
        out      DDRD, dummyreg

        ldi      dummyreg, 0xff      ; Port b OUTPUT
        out      DDRB, dummyreg
        ldi      dummyreg, (1 << ISC01 | 1 << ISC00) ; 0x02      ; IE0 rising edge
(ISC01=1;ISC00=1)
        sts      EICRA, dummyreg
        ldi      dummyreg, (1 << INT0) ; 0x01      ; turn IE0 on
        out      EIMSK, dummyreg

        cbi      PORTB, 1
        sbi      PORTB, 0

        sei

main:

        jmp main

isr_int0:
```

```

        push    r20
        cbi     PORTB, 0           ; turn led 0 off
        ldi     r20, 10           ; Load the counter
loop:    call    switch_led        ; Switch led1 state
        call    delay             ; delay half second aprox
        dec     r20               ; dec counter
        brne    loop             ; if is not 0 loop again
        cbi     PORTB, 1         ; Turn led 1 off, because if counter
is set in a odd number
                                           ; led will remain on after the
execution of this proc

        sbi     PORTB, 0         ; turn led 0 off
        pop     r20              ; restore r20
        reti

; turns on the led 1 if it is turned off and vice versa
switch_led:
        push    r16              ; Save the r16 value in the stack
        push    r17              ; Save the r16 value in the stack
        in      r16, PORTB       ; Read port b
        ldi     r17, 0x02        ; load 0000 0010
        eor     r16, r17         ; switch state of bit 2 in r16
        out     PORTB, r16       ; load portb with the bit switched
        pop     r17              ; restore r17
        pop     r16              ; restore r16
        ret

; Delay function from tpl
delay:                                     ; Delay procedure
        push    r20              ; Save the r20 value in the stack
        push    r21              ; Save the r21 value in the stack
        push    r22              ; Save the r22 value in the stack
        ldi     r22, 40          ;
loop1:    ldi     r21, 255         ;
loop2:    ldi     r20, 255         ;
loop3:    dec     r20             ; decrement r20 by 1
        brne    loop3           ; If r20 had reached 0, z flag would
have been seted
                                           ; and we will jump to loop 3
        dec     r21              ; The same as above
        brne    loop2
        dec     r22
        brne    loop1
        pop     r22              ; Set r22, r21, r20 to the same value
that
        pop     r21              ; it had before entere this proc
        pop     r20
        ret

```


Que hay que modificar para usar pull-ups:

Primero, se encienden las pull up internas del puerto.

```
ldi    dummyreg, 0x04 ; Pullups 2
out    PORTD, dummyreg
```

Segundo se utiliza el int0 en falling edge.

```
ldi    dummyreg, (1 << ISC01) ; 0x02 ; IE0 falling edge
        (ISC01=1;ISC00=0)
sts    EICRA, dummyreg
```

Resultado:

Se logra controlar la interrupción 0 por flanco.

Conclusiones:

Se puede controlar la interrupción 0 por flanco, sin embargo cada tanto, se dispara 2 veces.

Esto debe ser producto del rebote, probablemente se tendría que implementar un control del rebote del switch por software.