Fiuba
Facultad de Ingeniería
Universidad de Buenos Aires

(6609) Laboratorio de Microcomputadoras

Proyecto:
(tp6 timer)

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:

Controlar la frecuencia de encendido de un led mediante timers.

Desarrollo.

Utilizando 2 switches se seleccionará la frecuencia de encendido de un led, para eso se utilizaran timers.

Timers.

El avr328p cuenta con 3 timers, 2 de 8 bits y uno de 16.
Cada uno de ellos suma 1 en cada ciclo de reloj sin necesidad de intervenir programáticamente.
Cuenta con distintos modos de operación, en este tp se usará en timer con la interrupción por overflow.
Como la frecuencia de operación del microcontrolador es de 16MHz, los timers llegaron al overflow muy rápidamente, por ello cada uno cuenta con prescalers.
Cada prescaler funciona como divisor de frecuencia, que puede dividirse por 1, 8, 64, 256, 1024.

De esta forma se alcanzan tiempos para operaciones del orden de los segundos, necesarios para algunas aplicaciones, que tienen que ver con interacción humana.

Tabla de memoria.

El programa es simple. Se hace polling de ambos switches, dependiendo del valor de los mismos se enciende el timer y se setea el prescaler.

switch 1	switch 2	timer status	prescaler
0	0	off	n/a
0	1	on	64
1	0	on	256
1	1	on	1024

Tabla 1

Se utilizan las entradas del puerto D 2 y 3.
En el trabajo practico se pide usar las entradas 1 y 2 con resistencias de pull down.
El problema es que al poner rx/tx en pulldown el avrdude no logra subir el programa al micro.
Por lo tanto y con el fin de demostrar el uso de resistencias de pull down, se usan las entradas 2 y 3.
PIND 0000XX00, se ve que las posibles combinaciones son
00000000 = 0

00000100 = 4
00001000 = 8
00001100 = 12

Se usan estas combinaciones directamente en el programa para redirigir la lógica, si usar branches.
Se carga en Z la posición de la tabla correspondiente dependiendo de la entrada.

```
i00:            ldi     timer_on, 0                     ; time
                cbi     PORTB, 1
                ret
                nop
i01:            ldi     prescaler, ( 1<<CS10 | 1<<CS11 )   ; 64 prescaler
                ldi     timer_on,  ( 1 << TOIE1 )          ; time
                ret
                nop
i10:            ldi     prescaler, ( 1<<CS12 )             ; 256 prescaler
                ldi     timer_on,  ( 1 << TOIE1)           ; time
                ret
                nop
i11:            ldi     prescaler, ( 1<<CS10 | 1<<CS12 )   ; time
                ldi     timer_on,  ( 1 << TOIE1)           ; 1024 prescaler
                ret
```

es decir, Z = i00 es la posición 0 de la tabla 1.
Cada entrada de la tabla mide 4 words, por lo tanto:
posición(i00) + 0 bytes es i00.
posición(i00) + 4 bytes es i01.
posición(i00) + 8 bytes es i10.
posición(i00) + 12 bytes es i11.

O sea la posición de la lógica de tabla 1 en memoria de código será:
posición(i00) + PIND.

Registros de timers.

TCNT1H and TCNT1L – Timer/Counter1

Este registro tiene el valor del contador del timer 1.

Bit	7	6	5	4	3	2	1	0	
(0x85)	TCNT1[15:8]								TCNT1H
(0x84)	TCNT1[7:0]								TCNT1L
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

El mismo es de 16 bytes, para cargar un dato en este registro hay que respetar el siguiente orden.
TCNT1H <- high(valor)
TCNT1L <- low(valor)

De esta forma se asegura que se leen y escriben simultáneamente.
El microcontrolador escribe la parte alta en un registro temporal de 8 bits antes de realmente pasar ese dato TCNT1H, en el segundo paso se copian esos 8 bits en el registro TCNT1H, al mismo tiempo que el programa copia TCNT1L. De esta forma ambos registros se escriben al mismo tiempo. De no ser así el valor del timer no sería confiable.

TCCR1A – Timer/Counter1 Control Register B

Bit	7	6	5	4	3	2	1	0	
(0x81)	ICNC1	ICES1	–	WGM13	WGM12	CS12	CS11	CS10	TCCR1B
Read/Write	R/W	R/W	R	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

De este registro se usan los 3 bits menos significativos para controlar el prescaler.

CS12	CS11	CS10	Description
0	0	0	No clock source (Timer/Counter stopped).
0	0	1	$clk_{I/O}$/1 (no prescaling)
0	1	0	$clk_{I/O}$/8 (from prescaler)
0	1	1	$clk_{I/O}$/64 (from prescaler)
1	0	0	$clk_{I/O}$/256 (from prescaler)
1	0	1	$clk_{I/O}$/1024 (from prescaler)
1	1	0	External clock source on T1 pin. Clock on falling edge.
1	1	1	External clock source on T1 pin. Clock on rising edge.

En el programa, selecciona entre los 3 valores encuadrados en la imagen anterior.

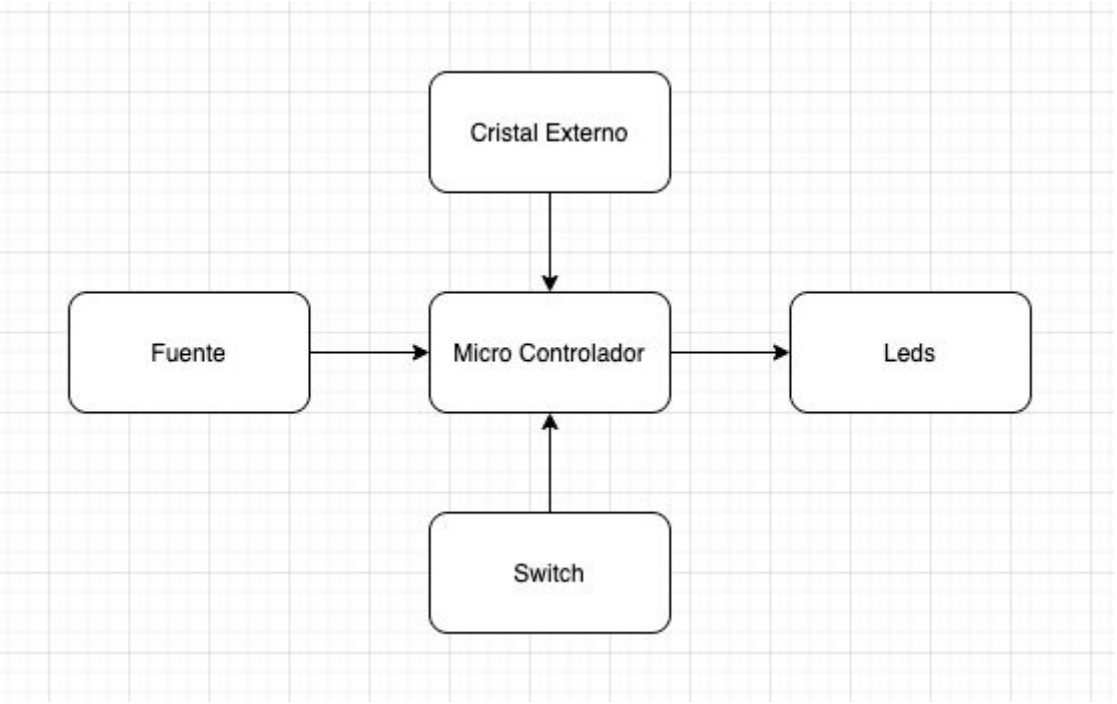
Debounce:

No hace falta por la forma en hacer polling de las entradas.

Listado de componentes:

Placa arduino UNO Atmega 328p $659 aprox 10 usd.
1 led (pack de 10) $70.
1 resistencia 220 ohm 1/8w 1% $50.
2 switch $10.

Diagrama en bloques:



Circuito esquemático:

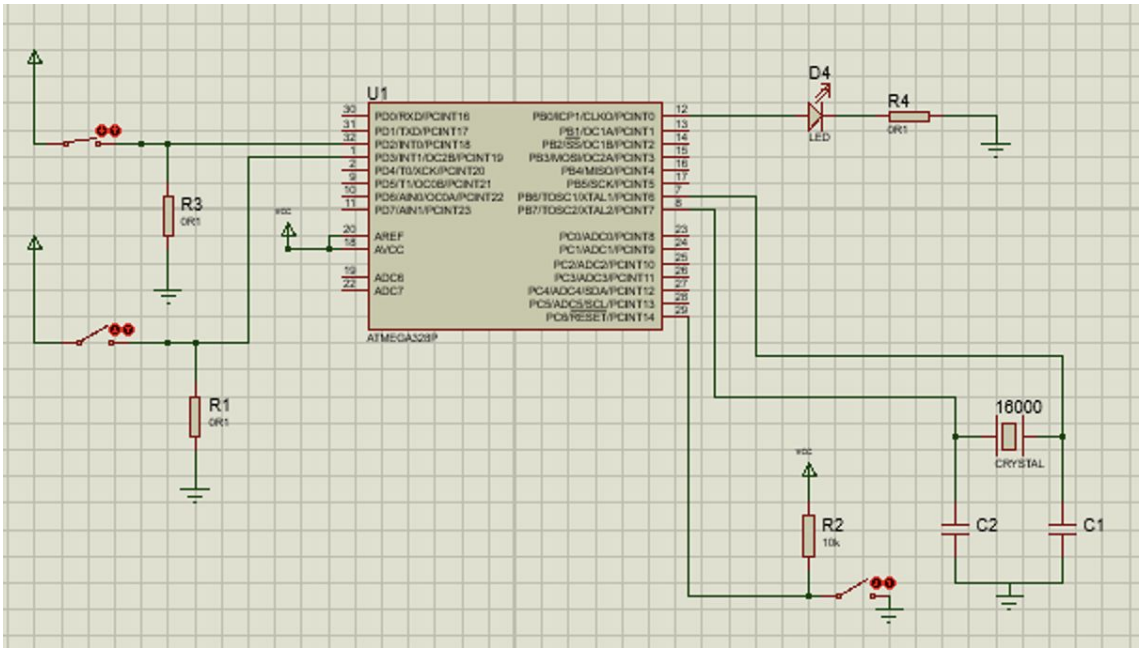
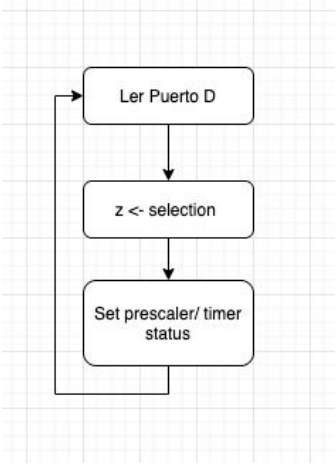
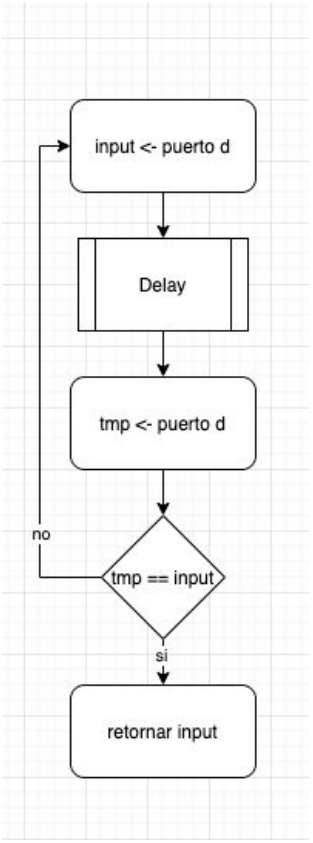


Diagrama de flujos:

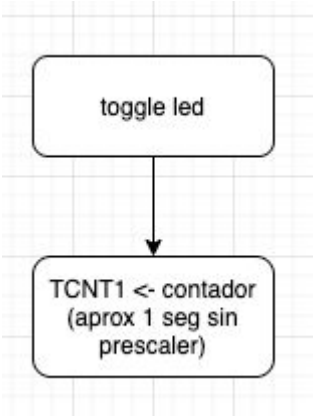
main



Leer puerto:



isr:



Código:

```
;
; ejercicio 6
;
; Author : cristian Simonelli
;



.include "m328pdef.inc"

.def dummyreg = r21
.def prescaler = r22
.def timer_on = r23
.def post_value = r19
.def input = r18

.equ TIMER_COUNT = 0xf9e6                    ; 1 second aprox

.cseg
.org 0x0000
          jmp     configuracion
.org OVF1addr
          jmp     isr_timovf1

.org INT_VECTORS_SIZE
configuracion:
; Inicializacion stack pointer
              ldi     dummyreg,low(RAMEND)
              out     spl,dummyreg
              ldi     dummyreg,high(RAMEND)
              out     sph,dummyreg
; port config
              ldi     dummyreg,0xf3          ; Port D 2/3 as input
              out     DDRD,dummyreg          ;
              ldi     dummyreg,0xff          ; Port b as output
              out     DDRB,dummyreg          ;

; timer 1 config
              ldi     dummyreg, high(63974)
              sts     TCNT1H,  dummyreg
              ldi     dummyreg, low(63974)
              sts     TCNT1L,  dummyreg

              ldi     dummyreg, 0
              sts     tccr1a, dummyreg

              sei

main:
```

```
                ldi     zl, low(i00)                    ; Loads Z with the first
position of the table
                ldi     zh, high(i00)                   ; Multiply by 2 is not
necessary in this case
                                                        ;   icall is world addressed
                call    read_input                      ; reads input, treates bounce,
stores data in input var
                add     zl, input                       ; Ads tableposition + (input
selection)
                ldi     dummyreg, 0                     ;   ( see table definition )
                adc     zh, dummyreg                    ;
                icall                                   ; calls whatever z is pointing
to
                sts     tccr1b, prescaler               ; Sets new prescaler info
                sts     TIMSK1 , timer_on               ; Sets new timer status (on or
off)
                jmp     main                            ; loop

; Each entry in this table weights 8 bytes (4 worlds) and represents an user selection
; After execute any of this entries both prescaler and time_on will be loaded
i00:             ldi     timer_on, 0                     ; time
                cbi     PORTB, 1
                ret
                nop
i01:             ldi     prescaler, ( 1<<CS10 | 1<<CS11 )  ; 64 prescaler
                ldi     timer_on,  ( 1 << TOIE1 )       ; time
                ret
                nop
i10:             ldi     prescaler, ( 1<<CS12 )           ; 256 prescaler
                ldi     timer_on,  ( 1 << TOIE1)        ; time
                ret
                nop
i11:             ldi     prescaler, ( 1<<CS10 | 1<<CS12 )  ; time
                ldi     timer_on,  ( 1 << TOIE1)        ; 1024 prescaler
                ret

; timer 1 overflow isr
isr_timovf1:

                call    toggle_led                      ; Switch led1 state
                ldi     r24, high(TIMER_COUNT)          ; Sets value into tcnt1 that
makes
                sts     TCNT1H,  r24                    ; timer 1 overflows every 1
second aprox
                ldi     r24, low(TIMER_COUNT)           ;
                sts     TCNT1L,  r24                    ;
                reti

; turns on the led 1 if it is turned off and vice versa
toggle_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

read_input:
                in      input, PIND                   ; Read port D
                call    delay                         ; Delay waiting for portd to
change (debounce)
                in      post_value, PIND              ; Read port D
                cp      input, post_value             ; If this values are differente,
                brne    read_input                    ;    could indicate that this is
bouncing
                andi    input, 0x0c                   ; this is actually unnecessary,
but clear info
                ret


; Delay function from tp1 (ligth version)
delay:                                                 ; Delay procedure
                push    r20                           ; Save the r20 value in the stack
                push    r22                           ; Save the r22 value in the stack
                ldi     r22, 40                       ;
loop1:           ldi     r20, 40                       ;
loop2:           dec     r20                           ; decrement r20 by 1
                brne    loop2                         ; If r20 had reached 0, z flag
would have been seted
                                                      ;   and we will jump to loop 2
                dec     r22                           ; The same as above
                brne    loop1
                pop     r22                           ; it had before entere this proc
                pop     r20
                ret                                   ; go back to main
```

Resultado:

Se logra controlar la frecuencia de un timer en base al cambio de prescaler.

Conclusiones:

Si bien se puede obtener el mismo resultado que mediante el uso de timers, usando lógica en el programa, el uso de los mismos es beneficioso ya que hace el programa más simple, controlable.
El contador del timer, la comparación contra un valor fijo y el manejo de interrupciones ,de forma paralela a la ejecución, permiten una programación más enfocada en la lógica del programa en sí que al control de eventos.