Universidad de Buenos Aires
Facultad de Ingeniería

86.07 - Laboratorio de microprocesadores

TP4: Timers

Berard, Lucia Magdalena
101213 - lberard@fi.uba.ar

1er cuatrimestre de 2021

En el siguiente trabajo práctico se tiene como objetivo afianzar los conociminetos sobre configuración y uso de timers, generación de interrupción por eventos de timer, manejo de antirrebotes de teclas y verificación de PWM, mediante la realizacion de un programa que haga parpadear un LED en 3 frecuencias distintas

---

Índice

1. Timers: Parpadeo en distintas frecuencias

1.1. Introducción

El programa consiste en hacer parpadear el LED conectado al PB0, en 3 frecuencias distintas o que lo deje ENCENDIDO FIJO, según los valores que haya en las entradas INT0,INT1 según se indica en el siguiente cuadro:

INT0	INT1	ESTADO DEL LED	PERÍODO
0	0	ENCENDIDO FIJO	-
0	1	PARPADEA CON PRESCALER CLK/64	0.52 s
1	0	PARPADEA CON PRESCALER CLK/256	2.09s
1	1	PARPADEA CON PRESCALER CLK/1024	8.39s

Cuadro 1: Distintas configuraciones de frecuencia

También se calculó la frecuencia o período con que se prenderá el LED en los 3 casos con la siguiente ecuación:

$$\tau = 2\frac{2^{16}}{\frac{f_{clock}}{preescaler}} \tag{1}$$

Se utilizó la placa de Arduino cuya frecuencia del clock es 16MHz.

Para resolver esta práctica se usa el Timer1, interrupción por OVERFLOW. Cuando el LED esté ENCENDIDO FIJO el timer esta apagado. En los otros 3 casos, el timer cuenta los pulsos de clock divididos por prescaler 64, 256 y 1024 respectivamente. Cuando se produce un overflow (desborde) cambia el estado del LED, es decir, si está prendido se apaga y viceversa.

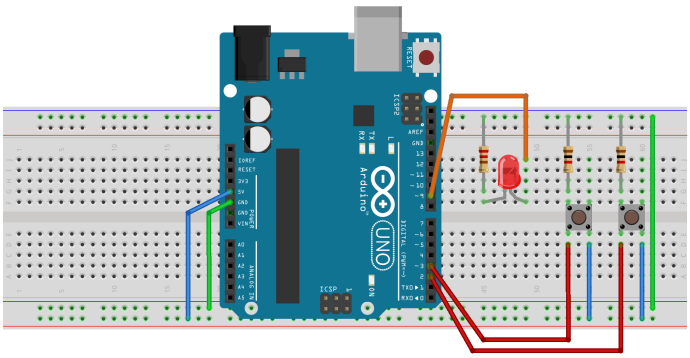

Figura 1: Diagrama esquemático

Se utilizó una placa Arduino, un LED, una resistencia de 220 Ω, 2 resistencias de 10 kΩ, cables para el conexionado, un protoboard y las entradas PD2 PD3 están conectados 2 switches, que como cualquier tecla produce rebotes al ser presionada por lo que se implementaron rutinas antirrebote para detectar correctamente las teclas.

1.2. Desarrollo

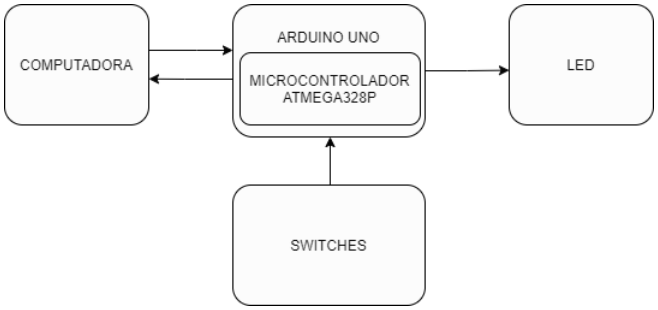

Figura 2: Diagrama en bloques

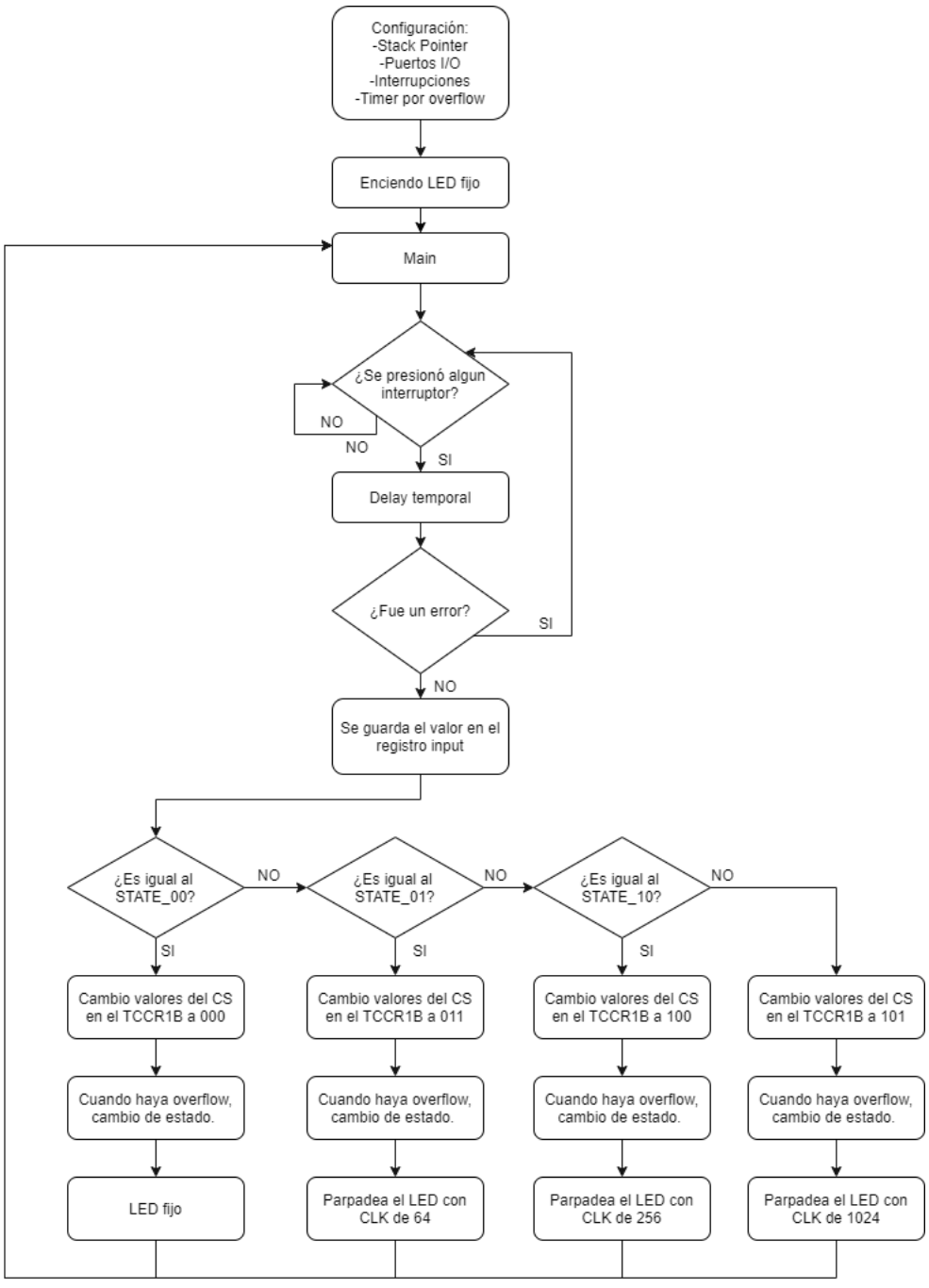

Figura 3: Diagrama de flujo

```
;Registros auxiliares
.def        aux1 = R16
.def        aux2 = R17

;Registro para entrada de interrupciones
.def        input = R19

;Configuracion rutinas de interrupcion:
.equ        MASK_INTs = 0x0C; 0000 11 00 ->ubicacion de INT0 e INT1

;Estados posibles - Configuracion frecuencia de parpadeo
.equ        STATE_01 = 0x04 ; 0000 01 00 -> clk 64
.equ        STATE_10 = 0x08 ; 0000 10 00 -> clk 256
.equ        STATE_11 = 0x0C ; 0000 11 00 -> clk 1024
.equ        STATE_00 = 0x00 ; 0000 00 00 -> fijo

;Modos del CLOCK
.equ        CS_011 = 0xFB ; 11111 011 -> clk 64
.equ        CS_100 = 0xFC ; 11111 100 -> clk 256
.equ        CS_101 = 0xFD ; 11111 101 -> clk 1024
.equ        CS_000 = 0xF8 ; 11111 000 -> Sin Clock
```

Listing 1: Registros y constantes utilizadas

1.2.1. Macros utilizadas

```
;Configuracion de los puertos ----------------------------------------------------------------
.macro configport
        ldi                aux1, @1
        out                @0, aux1
.endmacro

;Configuracion de las interrupciones ----------------------------------------------------------
.macro configINT
        lds         aux1, EICRA
        ori       aux1, (1 << ISC00)
        andi         aux1, ~((1<<ISC01))
        sts         EICRA, aux1

        lds         aux1, EICRA
        ori         aux1, (1 << ISC10)
        andi         aux1, ~((1<<ISC11))
        sts         EICRA, aux1
.endmacro

.macro configINT_MSK
        ldi         aux1,  (1<<INT0)|(1<<INT1)
        out         EIMSK,aux1
.endmacro

;Configuracion del timer ----------------------------------------------------------------------
.macro configTimer
        ;Modo Normal y Pin Tn flanco descendente

        ;WGM[1:0] = 00
        lds         aux1, TCCR1A
        andi         aux1, 0xfc ; 111111 00
        sts         TCCR1A, aux1

        ;WGM[2] = 0
        ;CS[2:0] 110: flanco descendente externo
        lds         aux1, TCCR1B
        andi         aux1, 0xf6 ; 1111 0 110
        sts         TCCR1B, aux1
.endmacro

.macro configTimerOverflow
        ldi         aux1, (1<<TOIE1)
        sts         TIMSK1, aux1
.endmacro

;Recordatorio: usar SEI despues de agregar estas macros!

;Inicializacion del Stack Pointer-------------------------------------------------------------
.macro initSP
        ldi                aux1, HIGH(RAMEND)
        out                SPH, aux1
        ldi                aux1, LOW(RAMEND)
        out                SPL, aux1
.endmacro
```

Listing 2: Macros para configuración e inicialización

```
.macro readINTs
        ;Obtengo la señal del interruptor y guardo en aux
        in              aux1, PIND
        ;Aplico mascara para que solo traiga los valores de INT0 e INT1
        andi        aux1, MASK_INTs
        ;Espero
        debounceTime
        ;Guardo la señal en input
        in              input, PIND
        andi        input, MASK_INTs
        ;Comparo para evitar errores.
        cp              aux1, input
        brne        INTend
.endmacro
```

Listing 3: Macros para leer Interrupciones

```
;-----------------------------------------------------------------------------------------
;                                                                          Delay por timer
;-----------------------------------------------------------------------------------------
.macro delayByTimer
        clr             aux2
        out             TCNT0, aux2

        lds             aux2, TCCR0A
        andi        aux2, ~((1 << WGM01)|(1<<WGM00))
        out             TCCR0A, aux2

        lds             aux2, TCCR0B
        ori             aux2, (1<<CS02)|(1<<CS00)
        andi        aux2, ~((1 << WGM02)|(1<<CS01))
        out             TCCR0B, aux2

        delayLoop:
                in              aux2, TIFR0
                sbrs        aux2, TOV0
                rjmp        delayLoop

                ldi             aux2, 0
                out             TCCR0A, aux2
                out             TCCR0B, aux2
                ldi             aux2, 0x01
                out             TIFR0, aux2

.endmacro

;-----------------------------------------------------------------------------------------
;                                                  Debounce Time con delay por timer
;-----------------------------------------------------------------------------------------
.macro debounceTime
                ldi             aux2, 2

        debounceLoop:
                delayByTimer
                dec             aux2
                cpi             aux2,0
                brne        debounceLoop
.endmacro
```

Listing 4: Macros debounce y delay utilizando timers

1.2.2. Código principal

```
.include "m328pdef.inc"
.include "macros.inc"


.cseg
;Interupciones------------
.org INT0addr
```

```
8           rjmp        INTaction
9   .org INT1addr
10          rjmp        INTaction
11  .org 0x001A ; Timer/Counter1 Overflow
12          rjmp        OVFaction
13  ;-------------------------
14
15  .org 0x0000
16          jmp                 config
17  .org INT_VECTORS_SIZE
18
19  ;-----------------------------------------------------------------------
20  ;                       Configuracion de los puertos, timer e interrupciones
21  ;-----------------------------------------------------------------------
22  config:
23          ;inicializo el stack
24          initSP
25
26          ;Interrupcion por overflow del timer1
27          configTimer
28          configTimerOverflow
29
30          ;Interrupciones
31          configINT
32          configINT_MSK
33
34          ;PORTB como salida(LED)
35          ;0x01 porque solo uso PB0 (puerto 8 arduino)
36          configport DDRB,0x01
37
38          ;PORTD como entrada (Pulsadores)
39          configport DDRD,0x00
40
41          ;Empieza con LED encendido fijo
42          sbi PORTB, PB0
43
44          sei
45
46  ;-----------------------------------------------------------------------
47  ;                                                               Main
48  ;-----------------------------------------------------------------------
49  ;El main principal no hace nada, se maneja todo con interrupciones
50  main:
51          rjmp main
52
53  ;-----------------------------------------------------------------------
54  ;                                       Rutina de interrupcion - Pulsador
55  ;-----------------------------------------------------------------------
56  INTaction:
57          ;Leo valor de las interrupciones
58          readINTs
59
60          ;Reviso cual es el estado correspondiente
61          cpi                 input, STATE_00
62          breq        no_clk
63
64          cpi                 input, STATE_01
65          breq        clk_64
66
67          cpi                 input, STATE_10
68          breq        clk_256
69
70          cpi                 input, STATE_11
71          breq        clk_1024
72
73          rjmp        INTend
74
75  INTend:
76          reti
77
78  ;-----------------------------------------------------------------------
79  ;                                                   Distintas frecuencias
80  ;-----------------------------------------------------------------------
81  no_clk:
82          ;CS_000: 0xF8 = 11111 000 -> Sin Clock
83          lds                 aux1, TCCR1B
```

```
84          andi        aux1, CS_000
85          sts             TCCR1B, aux1
86          sbi             PORTB, 0
87          rjmp        INTend
88
89  clk_64:
90          ;CS_011: 0xFB = 11111 011
91          lds             input, TCCR1B
92          ori             input, (1<<CS10)|(1<<CS11)
93          andi        input, ~((1<<CS12))
94          sts             TCCR1B, input
95          rjmp        INTend
96
97  clk_256:
98          ;CS_100: 0xFC = 11111 100
99          lds             input, TCCR1B
100         ori             input, (1<<CS12)
101         andi        input, ~((1<<CS11) | (1<< CS10))
102         sts             TCCR1B, input
103         rjmp        INTend
104
105 clk_1024:
106         ;CS_101: 0xFD = 11111 101
107         lds             input, TCCR1B
108         ori             input, (1<<CS10)|(1<<CS12)
109         andi        input, ~((1<<CS11))
110         sts             TCCR1B, input
111
112 ;---------------------------------------------------------------------
113 ;                                   Rutina de interrupcion - Overflow
114 ;---------------------------------------------------------------------
115 OVFaction:
116         sbi             PINB, 0
117 reti
```

1.3. Video funcionando

https://drive.google.com/file/d/1piaqrd21Il57fAb-9QNMsvDR1QBYc_YI/view?usp=sharing

1.4. Repositorio Github

https://github.com/fiuba-labo-de-micro-miercoles/2021_1c_trabajos_practicos-lmberard/tree/master/TP4_TIMERS

2. Conclusiones

En principio se tuvo demoras ya que el enunciado sugería conectar los interruptores al PD0 y PD1 pero luego de ser consultado en la práctica, se cambiaron los pines a PD2 y PD3 para utilizar interrupciones y de esta manera se logró hacer funcionar el programa de acuerdo a lo pedido.

Al haber utilizado macros en los ejercicios anteriores se ahorró tiempo para las configuraciones. Además se modificaron los métodos de debounce y delay para que utilicen el timer.

3. Bibliografía

- AVR Datasheet
- AVR Manual de instrucciones
- Esquemático Arduino UNO

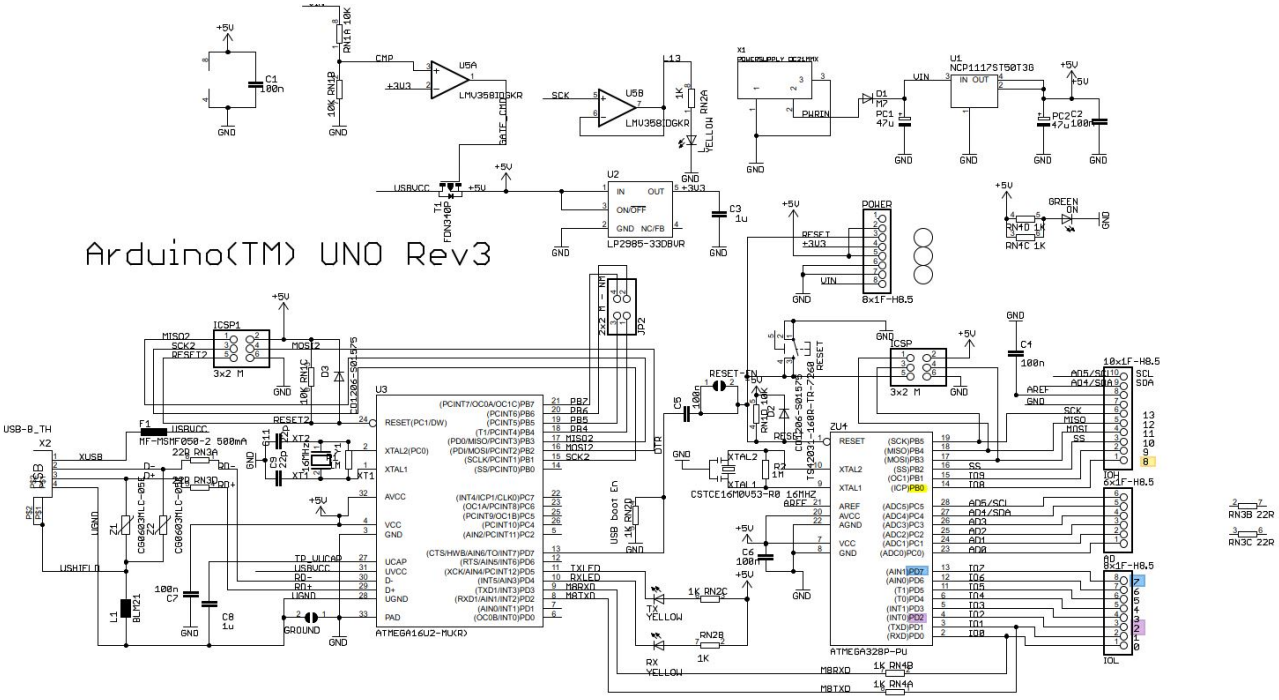

Figura 4: Esquemático Arduino UNO