Universidad de Buenos Aires
Facultad De Ingeniería
Año 2021 – 1.er Cuatrimestre

(82.07) Laboratorio de microprocesadores

Entrega N.º 4: Timers y PWM
Curso 01

Docentes a cargo:
Stola, Gerardo Luis
Salaya, Juan Guido
Cofman, Fernando

Integrante	Padrón	Correo electrónico
Lützelschwab, Nahila	100686	nlützelschwab@fi.uba.ar

Índice

1. Objetivos del proyecto

El objetivo del presente trabajo práctico es conocer el correcto manejo de los diferentes timers con sus configuraciones y modos de funcionamiento; la generación de interrupciones por eventos de timer; el manejo de antirrebotes de teclas;y la verificación de PWM a través de la variación de brillo de un LED.

2. Descripción del proyecto

El presente trabajo consistió en diseñar un programa que permita hacer parpadear un diodo LED a diferentes frecuencias, determinadas por el clock interno del Arduino UNO y prescalers. Luego se realizó otro programa que permita aumentar y disminuir la intensidad del diodo LED haciendo uso de la funcionalidad de modulación por ancho de pulsos, conocido como PWM, el cual permite modificar el ciclo de trabajo de la señal emitida por el pin conectado al diodo sin modificar su frecuencia.

2.1. Lista de componentes

A continuación se listan los componentes utilizados para la implementación del proyecto.

- Placa Arduino UNO
- Protoboard de 830 puntos
- Cables macho-macho para protoboard
- Dos pulsadores tact switch
- Seis resistencias de 330Ω
- Un diodo LED rojo

3. Esquemático

Como muestra el esquemático 1, se hizo uso de una placa Arduino UNO basada en el microcontrolador ATmega328p, a la cual se conectaron dos pulsadores a los pines PD2 y PD3 conectados a resistencias de $1k\Omega$ (en la práctica se utilizaron 3 resistencias de 330Ω) y un diodo LED rojo con una resistencia de 330Ω conectado a PB1.

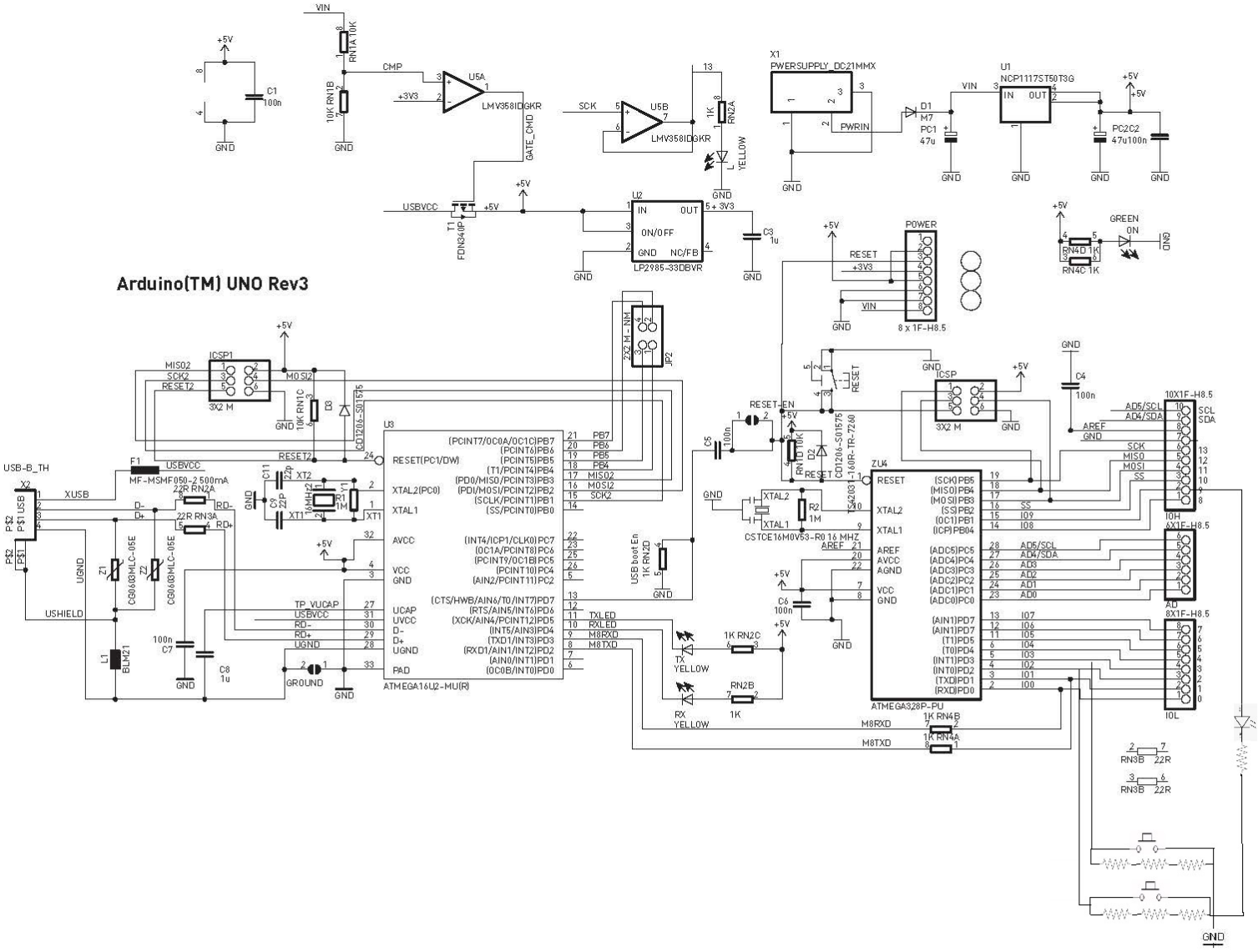

Figura 1: Esquematico completo del circuito implementado

4. Software

Mediante el software Microchip Studio, se desarrolló el programa a implementar en código Assembly.

Se diseñó un programa que permitiera hacer parpadear un diodo LED conectado al pin 1 del puerto B, en tres frecuencias distintas o que lo deje encendido fijo de acuerdo a los valores de las entradas conectadas a los pines 2 y 3 del puerto D como se muestran en la tabla 1:

Para ello fue necesario considerar que la placa Arduino UNO presenta un oscilador de cristal de 16MHz; el registro TCNT1, que contiene el valor del timer en cada instante, es de 16 bits y tendrá un tiempo de desborde o overflow según el prescaler utilizado. Una vez producidas dos interrupciones por desborde, se tiene un período para la conmutaicón entre estados, por lo tanto el período será dos veces el tiempo de desborde de acuerdo a la siguiente fórmula:

$$\tau = \frac{2^{16} cycles \cdot prescaler}{16MHz}$$

$$T = 2 \cdot \tau$$

PD2	PD3	Estado de LED	Período
0	0	Encendido fijo	
0	1	Parpadea con prescaler clk/64	524ms
1	0	Parpadea con prescaler clk/256	2,1s
1	1	Parpadea con prescaler clk/1024	8,4s

Tabla 1: Valores lógicos de los pines necesarios para mantener el LED encendido fijo o parpadear según los prescalers

Para su implementación fue necesario configurar el vector de interrupciones del Timer1 por overflow OVF1addr. Dicha interrupcion se basa en leer el estado actual del LED y se le aplica una máscara y mediante una EXOR proporciona el estado inverso de la salida. Por lo tanto, el software se basa en una constante lectura del estado de las entradas conectadas a los pulsadores, se las compara con los valores de la tabla para determinar su comportamiento, modificando el registro TCCR1B.

4.1. Diagrama de flujo de los Timers

En la figura 2 se puede observar el diagrama de flujo correspondiente al software principal de los Timers.

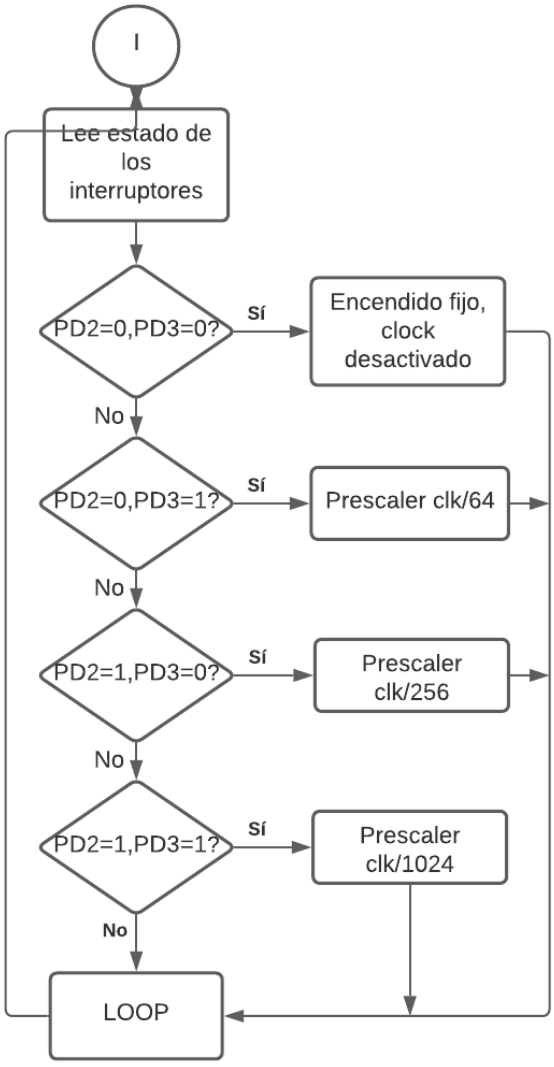

Figura 2: Diagrama de flujo Timers

En la figura 3 se puede ver el diagrama de flujo que corresponde al algoritmo PWM. Se diseñó un programa que permita aumentar y disminuir el brillo del diodo LED. Este se logró a partir del modo Fast PWM, el cual permite modificar el ciclo de trabajo de la señal emitida porel pin conectado al diodo sin modificar su frecuencia, alimentando de esta manera al LED. Por lo tanto se generó un tren de pulsos de ancho variable y período constante en el bit 1 del puerto B mediante el Timer1. Fue necesario configurar el Timer1 en modo comparación para modificar el registro de comparacion OC1A (conectado al LED). A su vez, se debió implementar una rutina de delay a modo de visualizar el efecto de la modificación del ciclo de trabajo de la señal a partir del brillo del LED.

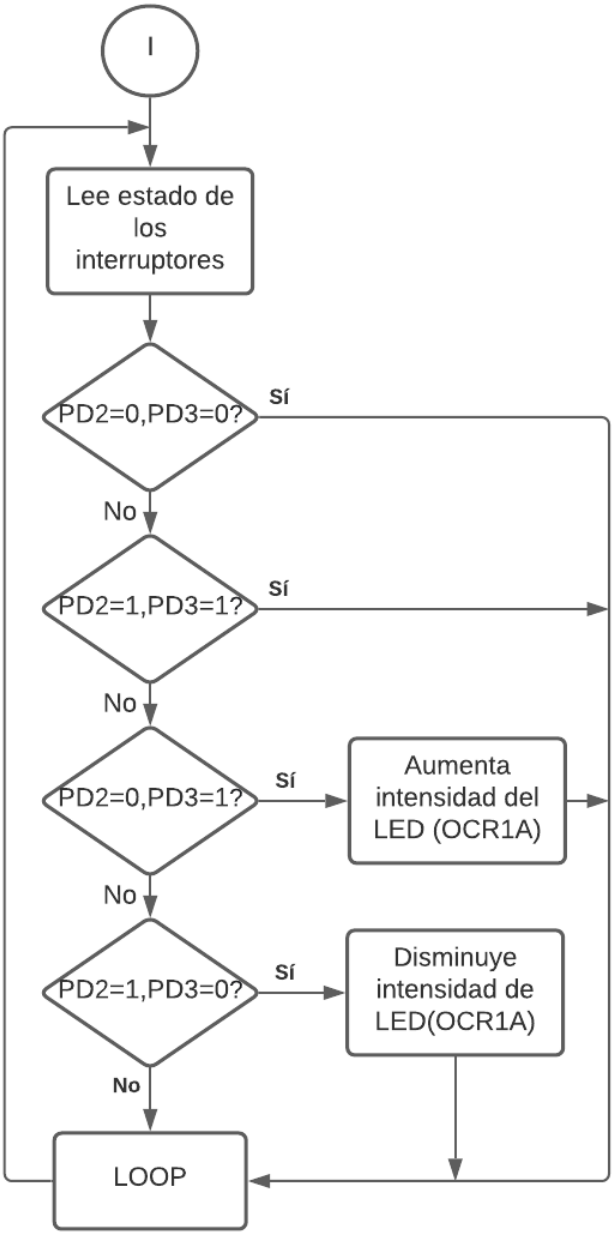

Figura 3: Diagrama de flujo Timers

5. Resultados

Se lograron los resultados deseados, estos se pueden observar en el video del siguiente link: https://youtu.be/kp17dFNMTdA

6. Conclusiones

Se puede concluir que este trabajo práctico permitió conocer el manejo de Timers, en específico por desborde,y el PWM a partir de la variación del duty-cycle de una señal generada. Observando el comportamiento resultante del circuito, la respuesta obtenida coincidió con la esperada. Además, el uso de PWM permite generar de manera más precisa una señal cuadrada, a diferencia de la situación que se daría al configurar un oscilador analógico, dado que el mismo introduciría ruido al sistema dada la cantidad de conexiones agregadas.

7. Anexo

7.1. Código fuente Timers

```
;
; Laboratorio de microprocesadores (86.07)
; Nahila Lutzelschwab
; Padron : 100686
; 1er cuatrimestre 2021
; Turno Martes 19hs
;

/*
  PB1 -> salida (LED)
  PD2,1 -> entradas

  PD2   |   PD3
  0   |   0   -> encendido fijo
  0   |   1   -> parpadeo con prescaler clk/64
  1   |   0   -> parpadeo con prescaler clk/256
  1   |   1   -> parpadeo con prescaler clk/1024

*/

.INCLUDE "m328pdef.inc"

.CSEG       ;Segmento de codigo


; Redefine registros con nombres significativo
.DEF TEMP = R16
.DEF PIND_REG = R17
.DEF PINB_REG = R18

.EQU PIND_MSK = 0b00001100
.EQU PINB_MSK = 0b00000001

.ORG 0
  RJMP START

.ORG OVF1addr ; Vector de interrupcion por overflow
  RJMP ISR_OVERFLOW

START:
  RCALL STACK_INITIALIZE
  RCALL PORTS_CONF
  RCALL TIMER1_CONF
  RCALL INTERRUPTS_CONF


LOOP:
  IN PIND_REG, PIND     ; Lee y carga la informacion del registro PIND
  ANDI PIND_REG, PIND_MSK   ; Aplica una mascara para quedarse solo los estados de las
    entradas (PD2 y PD3)

  CPI PIND_REG, 0 ; Si PD2 = 0 y PD3 = 0 -> encendido fijo del LED y apaga el timer
  BREQ LED_STEADY_ON

```

```
54    CPI PIND_REG, 0b00001000  ; Si PD2 = 0 y PD3 = 1 -> parpadea con prescaler clk/64
55    BREQ PRESCALER_64
56
57    CPI PIND_REG, 0b00000100  ; Si PD2 = 1 y PD3 = 0 -> parpadea con prescaler clk/256
58    BREQ PRESCALER_256
59
60    CPI PIND_REG, 0b00001100  ; Si PD2 = 1 y PD3 = 1 -> parpadea con prescaler clk/1024
61    BREQ PRESCALER_1024
62
63    RJMP LOOP     ; Vuelve la rutina
64
65
66  ; Inicia el stack en la parte alta de la memoria
67  STACK_INITIALIZE:
68    LDI TEMP, LOW(RAMEND)
69    OUT SPL, TEMP
70    LDI TEMP, HIGH(RAMEND)
71    OUT SPH, TEMP
72    RET
73
74  ; Configura puertos
75  PORTS_CONF:
76    LDI TEMP, (1<<DDB1)   ; Configura PB1 como salida
77    OUT DDRB, TEMP
78    CLR TEMP
79    OUT DDRD, TEMP      ; Configura el puerto D como entrada
80    RET
81
82  ; Configuracion del Timer1 por overflow y lo inicializa
83  TIMER1_CONF:
84    LDI TEMP, (1<<TOIE1)  ; TOEI1 = 1 del registro de mascara de interrupcion del temporizador(
        TIMSK1)
85                  ; habilita la interrupcion por overflow
86    STS TIMSK1, TEMP
87    RCALL INITIALIZE_TIMER1
88    RET
89
90  ; Incializa en cero el Timer1
91  INITIALIZE_TIMER1:
92    CLR TEMP
93    STS TCNT1H, TEMP
94    STS TCNT1L, TEMP
95    RET
96
97  ; Configura interrupciones
98  INTERRUPTS_CONF:
99    SEI       ; Habilita las interrupciones globales
100   RET
101
102 ; Encendido fijo y apagado del timer
103 LED_STEADY_ON:
104   RCALL TIMER_OFF
105   SBIS PORTB, 1   ; Se fija si esta prendido el LED ignora lo siguiente, sino lo prende
106   SBI PORTB, 1
107   RJMP LOOP     ; Vuelve al loop principal
108
109 ; Apaga el timer
110 TIMER_OFF:
111   LDI TEMP, 0x00
112   STS TCCR1B, TEMP    ; Detiene el timer
113   RET
114
115 ; Configura preescala clk/64
116 PRESCALER_64:
117   LDI TEMP, (1<<CS10)|(1<<CS11) ; Configura la preescala clk/64 en el registro de control B
        del Timer1
118   STS TCCR1B, TEMP
119   RJMP LOOP           ; Vuelve al loop principal
120
```

```
; Configura preescala clk/256
PRESCALER_256:
  LDI TEMP, (1<<CS12) ; Configura la preescala clk/256 en el registro de control B del Timer1
  STS TCCR1B, TEMP
  RJMP LOOP      ; Vuelve al loop principal

; Configura preescala clk/1024
PRESCALER_1024:
  LDI TEMP, (1<<CS10)|(1<<CS12) ; Configura la preescala clk/1024 en el registro de control B
    del Timer1
  STS TCCR1B, TEMP
  RJMP LOOP      ; Vuelve al loop principal

; Rutina de interrupcion por overflow
ISR_OVERFLOW:
  PUSH PIND_REG    ; Guarda en el stack el estado de las entradas PD2 y PD3

  IN TEMP, SREG    ; Guarda en el stack el registro de estados en caso de ser modificado
  PUSH TEMP

  IN PINB_REG, PINB ; Lee y carga la informacion del registro PINB
  LDI TEMP, 0b00000010     ; Aplica una mascara para quedarse solo con el estado del PB1 (LED)
  EOR TEMP, PINB_REG  ; Aplica una Exclusive OR para cambiar el estado del LED
  OUT PORTB, TEMP

  POP TEMP
  OUT SREG, TEMP     ; Devuelve del stack el registro de estados guardado
  POP PIND_REG     ; Devuelve del stack los valores de los registros guardados
  RETI
```

7.2. Código fuente PWM

```
;
; Laboratorio de microprocesadores (86.07)
; Nahila Lutzelschwab
; Padron : 100686
; 1er cuatrimestre 2021
; Turno Martes 19hs
;

/*
  PB1 -> salida (LED)
  PD2,1 -> entradas

  PD2   |   PD3
  0   |   0   -> encendido fijo
  0   |   1   -> parpadeo con prescaler clk/64
  1   |   0   -> parpadeo con prescaler clk/256
  1   |   1   -> parpadeo con prescaler clk/1024

*/


.INCLUDE "m328pdef.inc"

.CSEG        ;Segmento de codigo


; Redefine registros con nombres significativo
.DEF TEMP = R16
.DEF PIND_REG = R17
.DEF PINB_REG = R18

.EQU PIND_MSK = 0b00001100
.EQU PINB_MSK = 0b00000001


```

```
.ORG 0
  RJMP START


START:
  RCALL STACK_INITIALIZE
  RCALL PORTS_CONF
  RCALL TIMER1_CONF
  RCALL INTERRUPTS_CONF


LOOP:
  RCALL DELAY        ; Rutina de retardo mediante PWM

  IN PIND_REG, PIND    ; Lee el registro PIND
  ANDI PIND_REG, PIND_MSK ; Aplica una mascara para quedarse solo los estados de las entradas
    (PD2 y PD3)

  CPI PIND_REG, 0      ; Si PD2 = 0 y PD3 = 0 -> vuelve al loop
  BREQ LOOP_RETURN

  CPI PIND_REG, 0b00001100 ; Si ambos pulsadores estan presionados -> vuelve al loop
  BREQ LOOP_RETURN

  CPI PIND_REG, 0b00001000     ; Si PD2 = 0 y PD3 = 1 -> aumenta intensidad de LED
  BREQ BRIGHTEN_LED

  CPI PIND_REG, 0b00000100     ; Si PD2 = 1 y PD3 = 0 -> disminuye intensidad de LED
  BREQ DARKEN_LED

LOOP_RETURN:
  RJMP LOOP        ; Repite la rutina



; Inicia el stack en la parte alta de la memoria
STACK_INITIALIZE:
  LDI TEMP, LOW(RAMEND)
  OUT SPL, TEMP
  LDI TEMP, HIGH(RAMEND)
  OUT SPH, TEMP
  RET

; Configura puertos
PORTS_CONF:
  LDI TEMP, (1<<DDB1)    ; Configura PB1 como salida
  OUT DDRB, TEMP
  CLR TEMP
  OUT DDRD, TEMP       ; Configura el puerto D como entrada
  RET

; Configuracion del Timer1 PWM
TIMER1_CONF:
  LDI TEMP, (1<<COM1A1) | (1<<WGM10)     ; Configura como Fast PWM de 8 bits
  STS TCCR1A, TEMP

  LDI TEMP, (1<<WGM12) | (1<<CS11)
  STS TCCR1B, TEMP

  RCALL INITIALIZE_TIMER1
  RCALL SET_LED
  RET

; Inicializa en cero el Timer1
INITIALIZE_TIMER1:
  CLR TEMP
  STS TCNT1H, TEMP
  STS TCNT1L, TEMP
  RET
```

```
104
105 ; Inicializa el LED con el minimo brillo de PWM
106 SET_LED:
107   STS OCR1AH, TEMP
108   STS OCR1AL, TEMP ; posible utilizando PWM
109   RET
110
111 ; Configura interrupciones
112 INTERRUPTS_CONF:
113   SEI        ; Habilita las interrupciones globales
114   RET
115
116 ; Incrementa la intensidad del LED
117 BRIGHTEN_LED:
118   LDS TEMP, OCR1AL  ; Lee y carga la parte baja del registro OCR1A
119   CPI TEMP, 0xFF    ; Si alcanza su maximo valor no realiza nada sino incrementa el registro
120   BREQ LOOP_RETURN
121   INC TEMP
122   STS OCR1AL, TEMP  ; Carga el valor incrementado a la parte baja del registro OCR1A
123   RJMP LOOP      ; Vuelve al loop principal
124
125 ; Disminuye la intensidad del LED
126 DARKEN_LED:
127   LDS TEMP, OCR1AL  ; Lee y carga la parte baja del registro OCR1A
128   CPI TEMP, 0       ; Si alcanza su minimo valor no realiza nada sino decrementa el registro
129   BREQ LOOP_RETURN
130   DEC TEMP
131   STS OCR1AL, TEMP  ; Carga el valor decrementado a la parte baja del registro OCR1A
132   RJMP LOOP      ; Vuelve al loop principal
133
134
135 ; Rutina de retardo entre incrementos y decrementos
136 DELAY:
137   LDI T1, 200
138 LOOP0:
139   LDI T2, 50
140 LOOP1:
141   LDI T3, 23
142 LOOP2:
143   DEC T3
144   BRNE LOOP2
145   DEC T2
146   BRNE LOOP1
147   DEC T1
148   BRNE LOOP0
149   RET
```

8. Bibliografía

- Mazidi, M. A., Naimi, S., Naimi, S. (2010). AVR Microcontroller and Embedded Systems: Using Assembly and C (Pearson Custom Electronics Technology). New Jersey, United States of America: Pearson
- ATMEGA328P Datasheet (PDF) - ATMEL.http://ww1.microchip.com/downloads/ Corporation. en/DeviceDoc/Atmel-7810-Automotive-Microcontrollers-ATmega328P$_{D}atasheet.pdf$