Universidad de Buenos Aires

Facultad de Ingeniería

2do Cuatrimestre 2020

[86.07] Laboratorio de Microprocesadores

Trabajo Práctico Nº 4

Alumno: Agustín Miguel Flouret

Padrón: 102298

Turno: Martes

Docentes:

- Gerardo Stola
- Guido Salaya
- Fernando Cofman

Fecha de entrega: 24 de noviembre de 2020

Índice

1. Objetivo del proyecto

En este proyecto se buscará introducirse en la utilización de los timers, haciendo uso de las interrupciones por evento de timer, y del modo PWM para controlar el brillo de un LED.

2. Descripción del proyecto

Para realizar el proyecto se utilizará la placa Arduino UNO, que incluye un microcontrolador ATmega328P. La programación del microcontrolador se realizará a través del Arduino, utilizando el software AVRDUDE. El lenguaje de programación que se usará es el Assembly de AVR, y se ensamblará con Atmel Studio. El hardware externo se detallará en la siguiente sección.

Las tareas a realizar son las siguientes:

- En primer lugar se hará un programa que haga parpadear un LED conectado en el pin PB0, en 3 frecuencias distintas o que lo deje encendido fijo, según los valores que haya en las entradas PD2 y PD3 como se indica en la Tabla 1. Se utilizará el timer 1 y la interrupción por overflow.

PD2	PD3	Estado del LED
0	0	Encendido fijo
0	1	Parpadea con prescaler CLK/64
1	0	Parpadea con prescaler CLK/256
1	1	Parpadea con prescaler CLK/1024

Tabla 1: Modos de parpadeo del LED

- En el segundo programa se controlará el brillo de un LED usando dos pulsadores. Los pulsadores variarán el ancho de pulso de una señal que alimenta al LED, mediante la funcionalidad PWM del timer 1.

3. Listado de componentes y gastos

- 1 Arduino UNO - $1009
- 1 protoboard - $300
- 1 LED rojo - $10
- 2 pulsadores - $54
- 1 resistor 220Ω - $7
- 2 resistores $10k\Omega$ - $14
- **Gasto total: $1394**

4. Circuito esquemático de Arduino UNO

En la figura 1 se muestra el circuito esquemático completo de la placa Arduino UNO que se usará en este trabajo. Para poder observar con mayor detalle las conexiones con otros componentes, en las secciones que siguen se mostrará solo la parte inferior derecha del esquemático en forma ampliada.

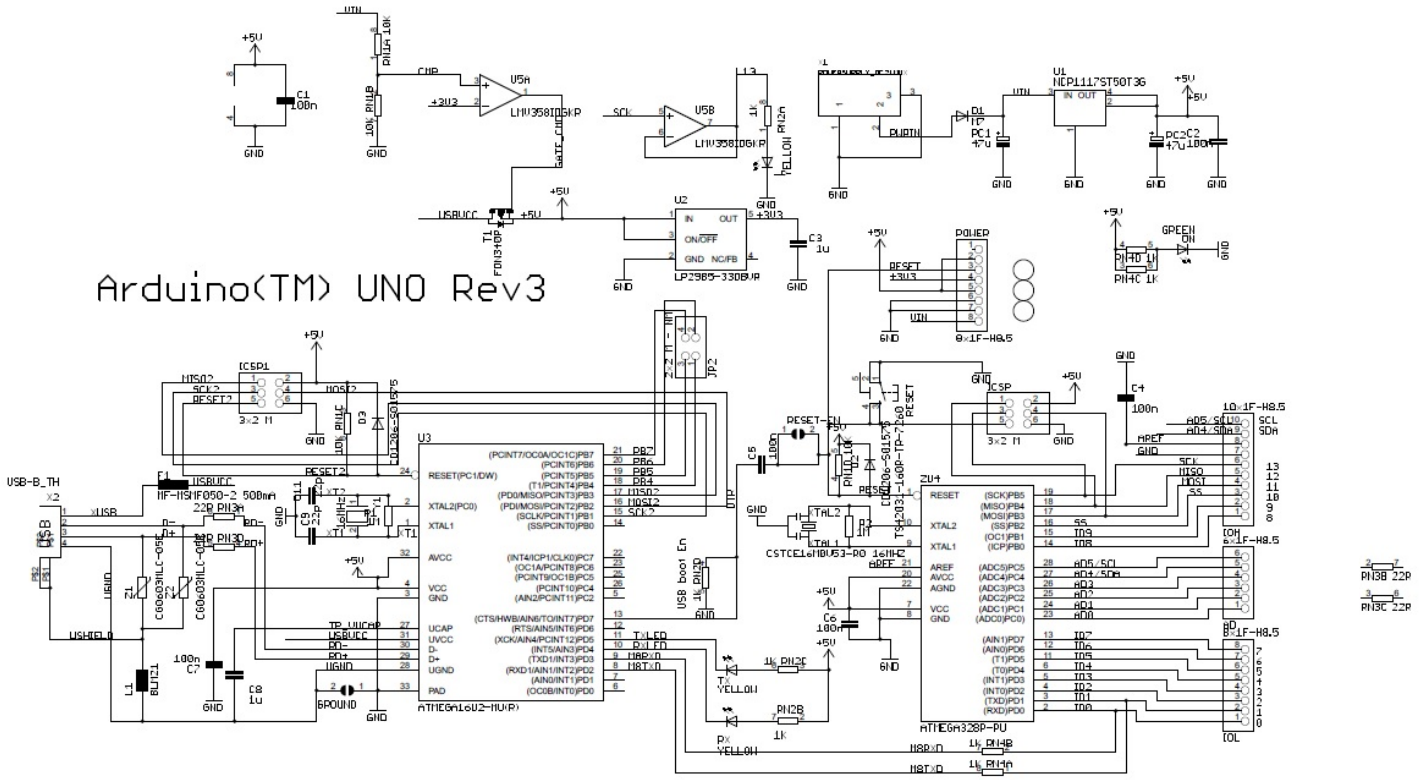

Figura 1: Circuito esquemático de la placa Arduino UNO.

5. Parte 1 - Timers

Se conectó un LED al pin PB0 y dos pulsadores (con resistores de pull-down de $10k\Omega$) a los pines PD2 y PD3. A continuación se muestra el diagrama de conexiones en bloques y el circuito esquemático de esta etapa del proyecto.

5.1. Diagrama de conexiones en bloques

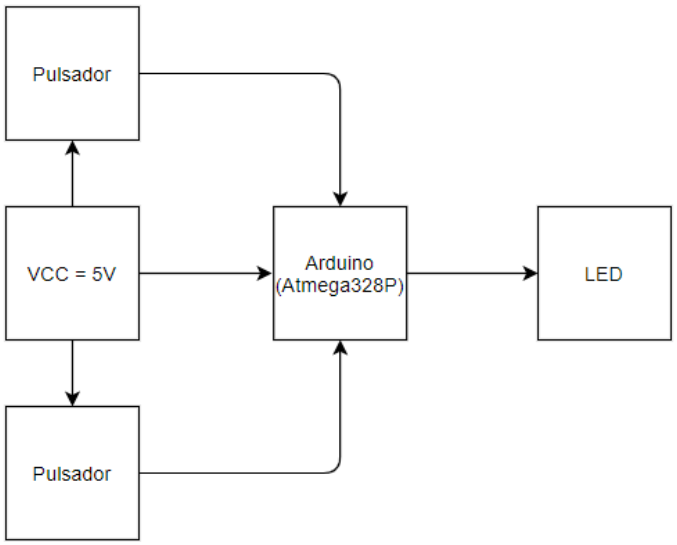

Figura 2: Diagrama de bloques del proyecto.

5.2. Circuito esquemático

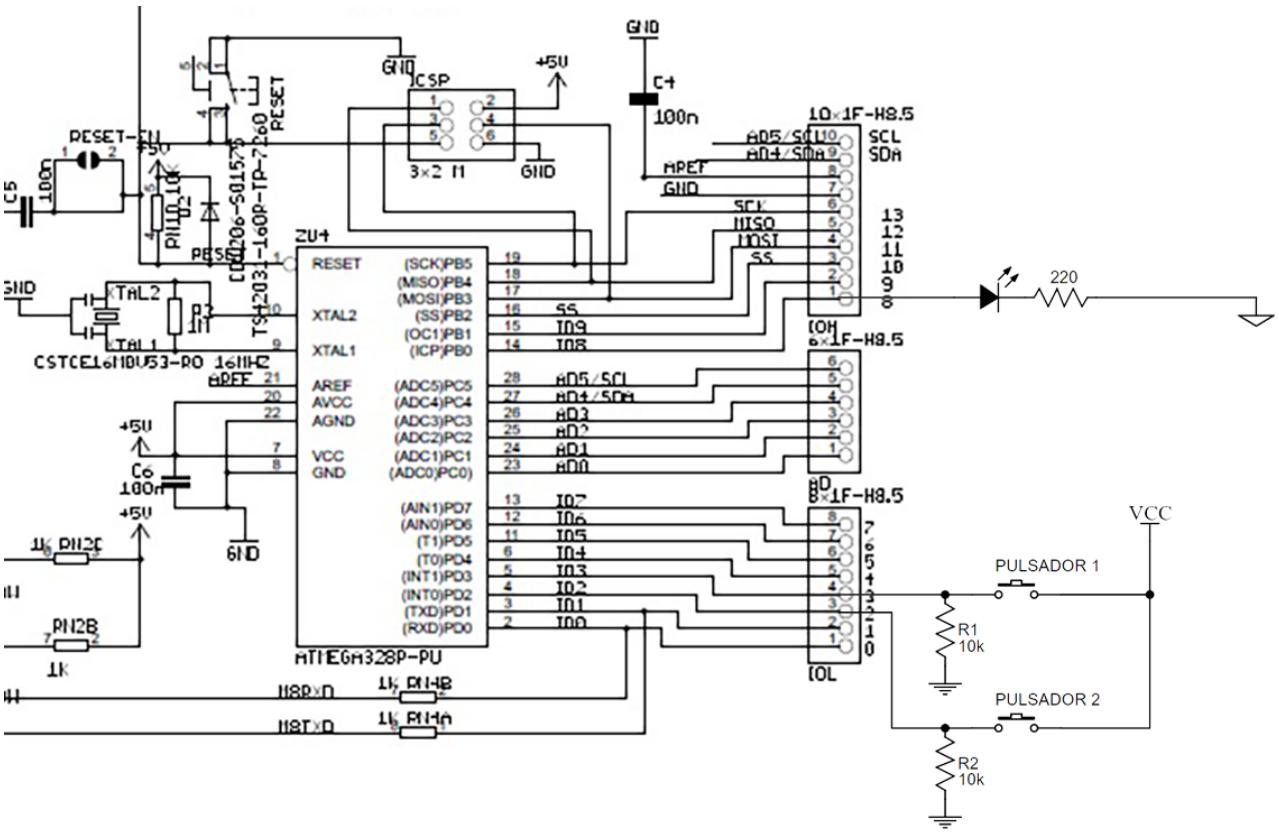

Figura 3: Circuito esquematico del caso 1.

5.3. Software

En este programa se utilizó el timer 1. Al iniciar el programa se habilita la interrupción por overflow del timer 1 (TOIE1) y se configura como salida el pin PB0, donde está conectado el LED.

En el loop principal del programa, se conmuta el estado del LED, y se inicializa (o no) el timer 1 según el estado de las entradas PD2 y PD3, controladas por los pulsadores. Al producirse la interrupción por overflow del contador, se repite este proceso, generando así un parpadeo del led.

La frecuencia de este parpadeo depende del valor que se carga en el registro TCNT1 y de la configuración del prescaler. El valor que se carga en el registro es -16000. Sin prescaler, el overflow se produce luego de 16000 ciclos de clock. A una frecuencia de clock de 16 MHz, esto corresponde a un delay de 1 ms, y, por lo tanto, una frecuencia de parpadeo de 500 Hz. Dividiendo por el valor del prescaler se obtienen las frecuencias de parpadeo para cada caso:

PD2	PD3	Estado del LED	Frecuencia
0	0	Encendido fijo	-
0	1	Parpadea con prescaler CLK/64	7.81 Hz
1	0	Parpadea con prescaler CLK/256	1.95 Hz
1	1	Parpadea con prescaler CLK/1024	0.49 Hz

Tabla 2: Frecuencias de parpadeo del LED

A continuación se muestra el diagrama de flujo y el código del programa.

5.3.1. Diagrama de flujo

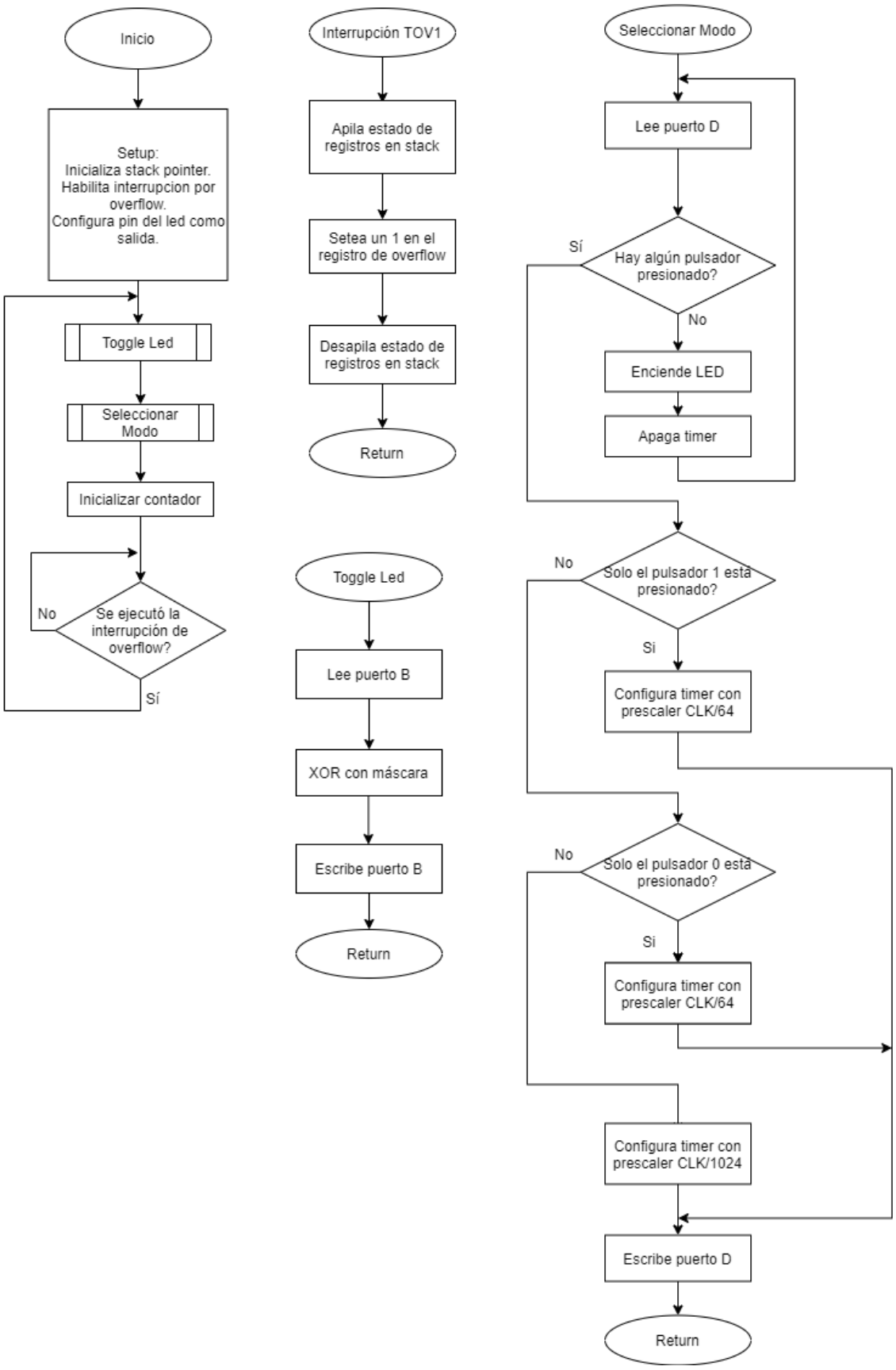

Figura 4: Diagrama de flujo del programa de timers.

5.3.2. Código del programa

```
.include "m328pdef.inc"

.def overflow = r18
.equ valor_inicial_TCNT1 = -16000
.equ pin_led = PB0

.org 0
        rjmp inicio

.org OVF1addr
    rjmp interrupcion_overflow

inicio:
                ;Inicializa stack
        ldi r16, high(RAMEND)
        out SPH, r16
        ldi r16, low(RAMEND)
        out SPL, r16

                ldi r16, (1<<TOIE1)
                sts TIMSK1, r16                         ; Habilita interrupcion por overflow

                sbi DDRB, pin_led               ; Configura el pin del led como salida

                sei


main:        call toggle_led
                call seleccionar_modo
                call inicializar_contador
                clr overflow
loop:        cpi overflow, 1
                brne loop        ; espera que se ejecute la interrupcion por overflow
                rjmp main


; Configura el timer segun los pulsadores presionados
seleccionar_modo:
                in r16, PIND
                andi r16, 0b00001100
caso0:         cpi r16, 0x00000000
                brne caso1
                ; Mientras no haya ningun pulsador presionado, enciende el led y apaga el timer
                sbi PORTB, pin_led
                ldi r20, 0
                sts TCCR1B, r20
                rjmp seleccionar_modo
caso1:         cpi r16, 0b00001000
                brne caso2
                ldi r20, (1<<CS11)|(1<<CS10) ; Prescaler clk/64
                rjmp salida
caso2:         cpi r16, 0b00000100
                brne caso3
                ldi r20, (1<<CS12)        ; Prescaler clk/256
                rjmp salida
caso3:         cpi r16, 0b00001100
                brne salida
                ldi r20, (1<<CS12)|(1<<CS10) ; Prescaler clk/1024
```

```
salida:         sts TCCR1B, r20
                ret

; Conmuta el estado del led
toggle_led:
                in r16, PORTB
                ldi r17, (1<<pin_led)
                eor r16, r17
                out PORTB, r16
                ret

; Inicializa el contador TCNT1 del timer 1
inicializar_contador:
                ldi r16, high(valor_inicial_TCNT1)
                sts TCNT1H, r16
                ldi r16, low(valor_inicial_TCNT1)
                sts TCNT1L, r16
                ret

; Setea el registro de overflow
interrupcion_overflow:
        push r16
        in r16, sreg
        push r16
        ldi overflow, 1
        pop r16
        out sreg, r16
        pop r16
    reti

```

6. Parte 2 - PWM

En este caso las conexiones son las mismas, aunque se conectó el LED al pin PB1, ya que esta es la salida OC1A del timer 1 en modo PWM.

6.1. Diagrama de conexiones en bloques

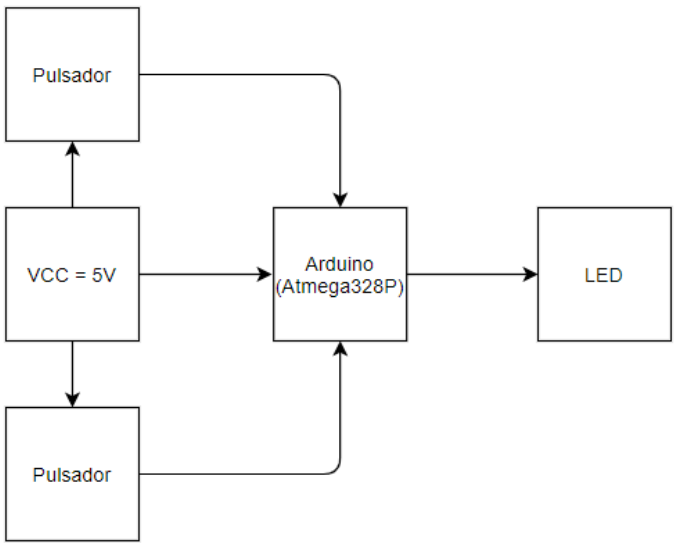

Figura 5: Diagrama de bloques del proyecto.

6.2. Circuito esquemático

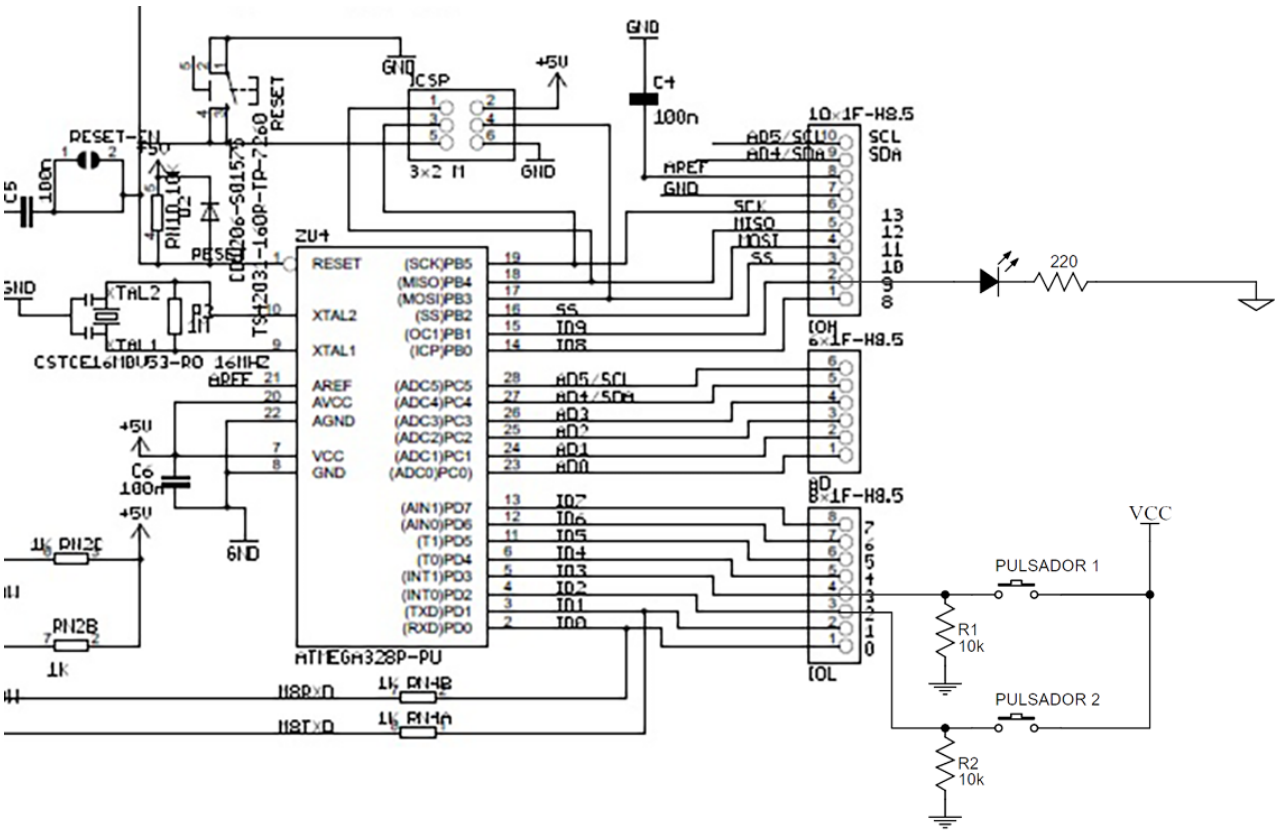

Figura 6: Circuito esquematico del caso 2.

6.3. Software

En este programa se utiliza el Timer 1, pero en modo PWM. En este caso se configuró en el modo 5 (Fast PWM, 8-bit), non-inverting. Para variar el ancho de pulso de la señal del PWM se utilizaron dos pulsadores, que al ser presionados activan las interrupciones externas por flanco ascendente. Al presionar el pulsador conectado al pin PD2, aumenta el valor del registro OCR1A, y entonces aumenta el ancho de pulso de la señal. Al presionar el pulsador conectado al pin PD3, disminuye. Los valores que se cargan en el registro OCR1A son 0b00000000, 0b00000001, 0b00000011, 0b00000111, ... , 0b11111111. De esta forma, al presionar los pulsadores se duplica o se divide por 2 (aproximadamente) el ancho de pulso, y, por consiguiente, el brillo del LED.

Como las interrupciones tienen una duración muy corta y se activan por flanco, pueden ser activadas más de una vez por el rebote del pulsador. Para solucionar esto se incluye un delay de 200 ms dentro de la rutina de delay, y se limpia el flag de la interrupción.

6.3.1. Diagrama de flujo

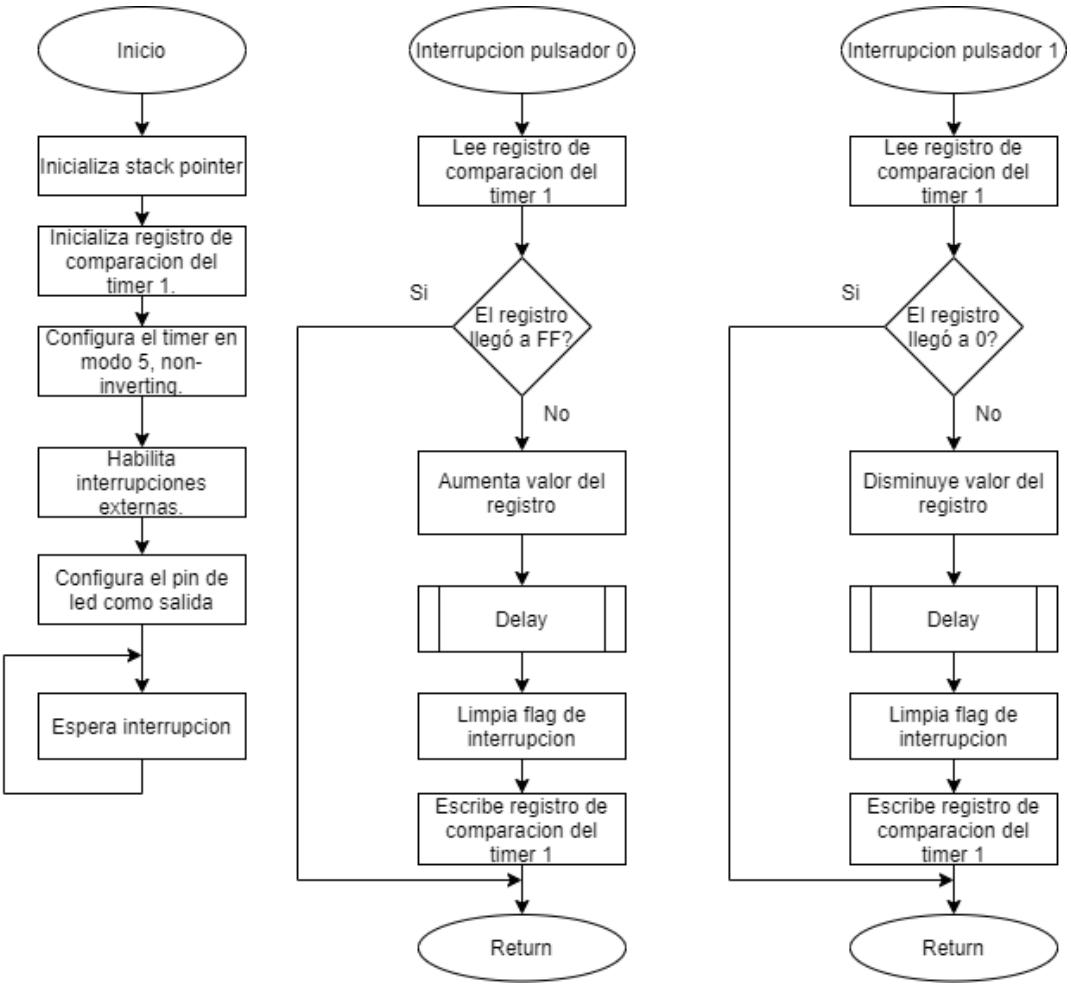

Figura 7: Diagrama de flujo del programa de PWM.

6.3.2. Código del programa

```
.include "m328pdef.inc"

.equ pin_led = PB1

.org 0
        rjmp inicio


.org 0x02
                rjmp interrupcion_pulsador0

.org 0x04
                rjmp interrupcion_pulsador1


inicio:
        ;Inicializa stack
        ldi r16, high(RAMEND)
        out SPH, r16
        ldi r16, low(RAMEND)
        out SPL, r16

```

```
23                  ; Inicializa el registro de comparacion del timer 1
24                  ldi r21, 0x00
25                  sts OCR1AH, r21
26                  ldi r22, 0x01
27                  sts OCR1AL, r22
28
29                  ; Configura el timer en modo 5 (Fast PWM, 8-bit), non-inverting
30                  ldi r16, (1<<COM1A1)|(1<<WGM10)
31                  sts TCCR1A, r16
32                  ldi r16, (1<<WGM12)|(1<<CS10)
33                  sts TCCR1B, r16
34
35                  ; Habilita INT0 e INT1 como interrupciones por flanco ascendente
36                  ldi r16, (1<<ISC11)|(1<<ISC10)|(1<<ISC01)|(1<<ISC00)
37                  sts EICRA, r16
38                  ldi r16, (1<<INT1)|(1<<INT0)
39                  out EIMSK, r16
40
41                  sbi DDRB, pin_led               ; Configura el pin del led como salida
42
43                  sei                ; Habilita interrupciones globales
44
45  esperar:
46                  rjmp esperar        ; Espera que se presione algun pulsador
47
48  ; Aumenta el valor del registro OCR, si el valor no es el maximo (0xFF)
49  interrupcion_pulsador0:
50                  lds r16, OCR1AL
51                  cpi r16, 0xFF
52                  breq salida_int0
53                  sec
54                  rol r16
55                  sts OCR1AL, r16
56                  ; Llama a la rutina de delay y limpia el flag de interrupcion,
57                  ; para evitar que la interrupcion se ejecute dos veces por el rebote
58                  call delay
59                  sbi EIFR, 0x00
60  salida_int0:
61                  reti
62
63  ; Disminuye el valor del registro OCR, si el valor no es el minimo (0x00)
64  interrupcion_pulsador1:
65                  lds r16, OCR1AL
66                  cpi r16, 0x00
67                  breq salida_int1
68                  lsr r16
69                  sts OCR1AL, r16
70                  ; Llama a la rutina de delay y limpia el flag de interrupcion,
71                  ; para evitar que la interrupcion se ejecute dos veces por el rebote
72                  call delay
73                  sbi EIFR, 0x01
74  salida_int1:
75                  reti
76
77
78
79  ; Delay de 200ms (3200000 ciclos)
80  delay:
81      ldi  r18, 17
82      ldi  r19, 60
```

```
83      ldi  r20, 204
84  L1: dec  r20
85      brne L1
86      dec  r19
87      brne L1
88      dec  r18
89      brne L1
90          ret
91
```

7. Resultados

Se logró obtener los resultados esperados en ambas partes del proyecto: hacer parpadear el LED a distintas frecuencias usando el Timer 1 y las interrupciones por overflow, variar el brillo del LED mediante el PWM, y manejar el rebote de los pulsadores.

8. Conclusiones

En este trabajo se logró implementar con éxito lo requerido, y sirvió como una introducción al manejo de timers en varios modos y del PWM, así como para integrar estos temas con técnicas aprendidas en los otros trabajos, como el uso de interrupciones externas activadas por pulsadores. Además, se vio la importancia de controlar el rebote del pulsador mediante software. Por ejemplo, el rebote podría producir un comportamiento indeseado por el cual el LED disminuye o aumenta su brillo más de lo que corresponde a una sola activación del pulsador, por lo tanto es importante evitar que suceda esto.