Fiuba
Facultad de Ingeniería
Universidad de Buenos Aires

Laboratorio de Microprocesadores (86.07)

Proyecto:

Trabajo Practico Nº 4

Timers y PWM

Profesor:	Ing. Guillermo Campiglio
Cuatrimestre / Año:	1 er cuatrimestre 2021
Turno de clases prácticas:	Miércoles
Jefe de Trabajos Prácticos:	Ing. Pedro Ignacio Martos
Docente guía:	Ing. Fabricio Baglivo

Autores			**Seguimiento del proyecto**								
Nombre	**Apellido**	**Padrón**									
Gonzalo	Puy	99784									

Observaciones:

Fecha de aprobación		

Firma J.T.P

COLOQUIO	
Nota Final	
Firma Profesor	

Índice

1. Introducción

El presente trabajo comprende la configuración y uso de los *Timers/Counters* disponibles en los microcontroladores de la familia AVR. Además, se busca comprender las interrupciones por eventos del timer y la función para generar ondas del timer.

Para lograr los objetivos se realizaran 2 programas distintos, con los cuales se podrán entender los conceptos propuestos por el enunciado.

2. Desarrollo

2.1. Banco de mediciones

Para el trabajo se utilizara el siguiente banco de mediciones, compuesto por

- Placa de desarrollo Arduino "UNO" y su respectivo cable para conectar la placa a la PC.
- El microcontrolador a usar, es el que viene integrado en la placa Arduino: `Atmega328P`.
- Protoboard
- Cables macho-macho para conexión del protoboard y de la placa Arduino.
- 2 pulsadores
- 1 LEDs (Rojo).
- 1 resistencias de 220 Ω.
- 2 resistencias de 10 kΩ.

Si bien los programas son distintos, los elementos utilizados son los mismos para ambos.

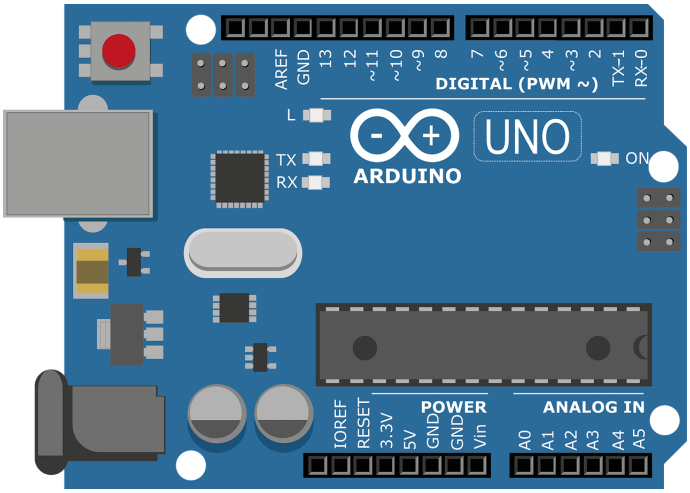

Figura 1: Placa de desarrollo Arduino UNO.

El diagrama de bloques simplificados para ambos programas es el siguiente

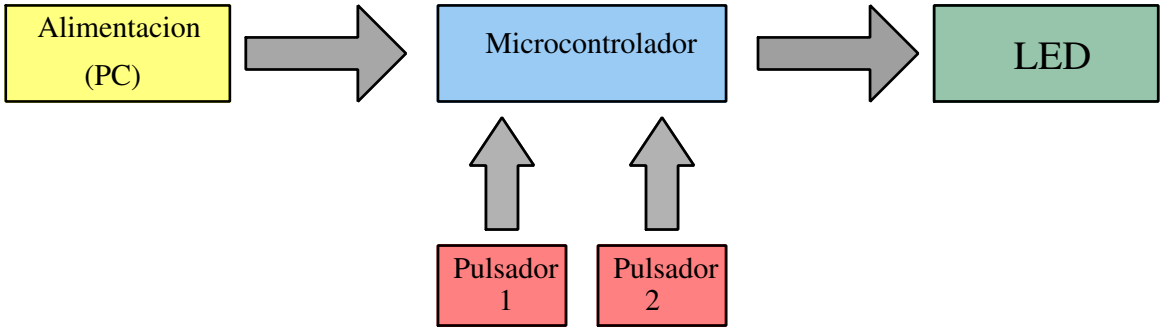

Figura 2: Diagrama de bloques.

2.2. Programa 1: Timers

Para el primer programa se utiliza una conexión como se muestra a continuación

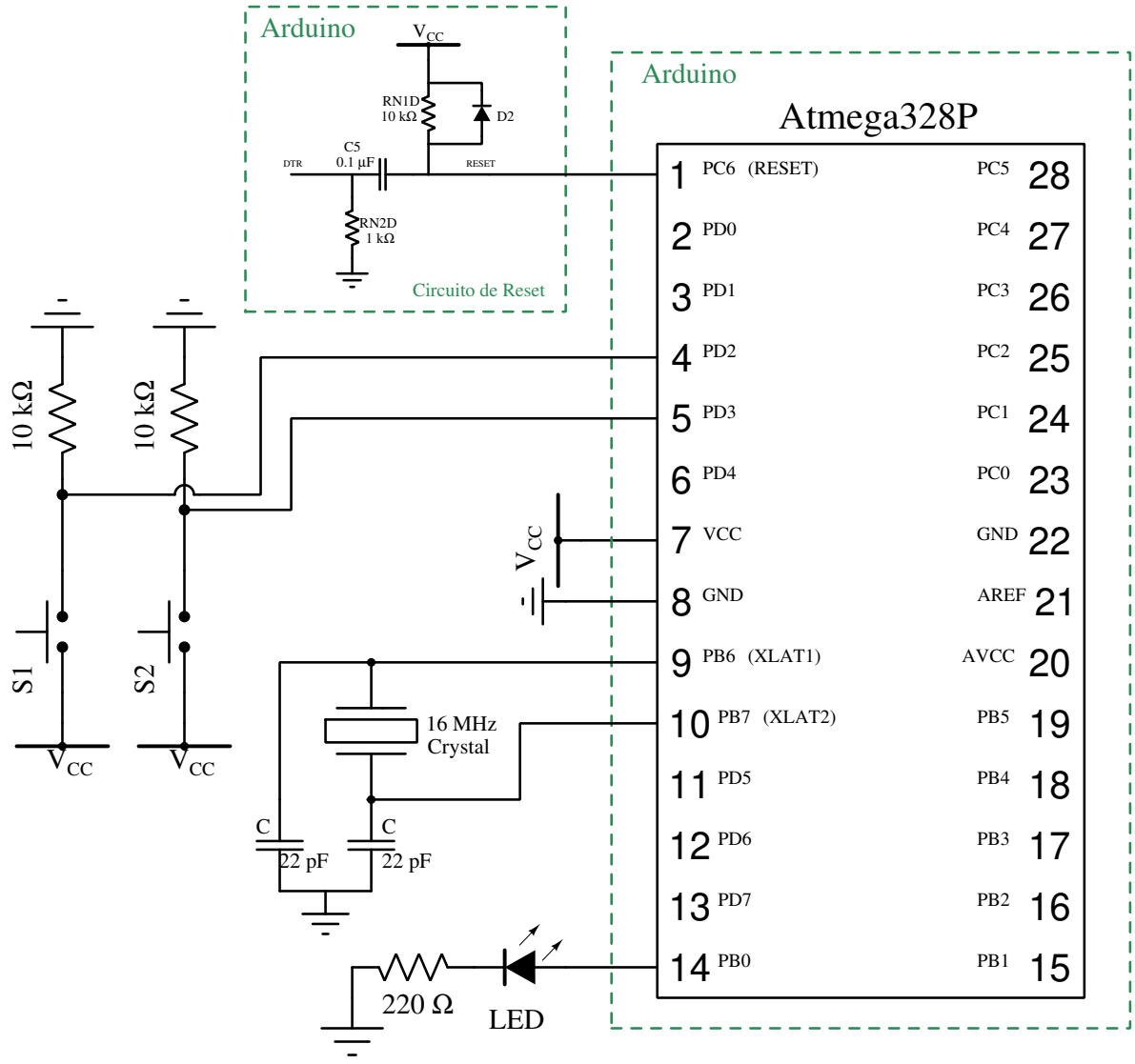

Figura 3: Conexión para el programa 1.

Como puede observarse en la figura 3, se tiene una conexión del tipo *pull-down* para los pulsadores. Esto es, cuando no están presionados se fuerza un '0' logico en el pin, mientras que al pulsarlos se fuerza un '1'logico.

Por otro lado, debido a la conexión del LED, este se encenderá al colocar un '1' logico en el pin correspondiente.

2.2.1. Programa a implementar

Para este programa se hará parpadear el LED conectado a PB0, en 3 frecuencias distintas o quedará encendido permanentemente según los valores que haya en las entradas PD2 y PD3 (pulsadores) haciendo uso del Timer1 con interrupción por *overflow*.

Cuando el led este en modo encendido fijo, el timer estará apagado. En los otros casos, el timer contara los pulsos de clock divididos por *prescaler*. Al producirse un desborde (*overflow*), se cambiará el estado del LED, esto es, si esta apagado se encenderá y viceversa.

Se resumen los modos del LED en la siguiente tabla

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 $CLK/1024$

En cuanto al Timer1 se configurará en modo normal, y solo se seteará el *prescaler* de acuerdo al estado de PD2 y PD3.

Para calcular la frecuencia a la que parpadea el LED con cada *prescaler* se debe tener en cuenta la frecuencia del oscilador que posee el Arduino UNO es de 16 MHz y que el timer toma esta como referencia. Además, se sabe que el timer1 es un timer de 16 bits por lo que cuenta desde 0 (ya que no se seteo ningún valor inicial) hasta 2^{16}. Entonces se calcularon los valores con la siguiente fórmula.

$$f_{blink} = \frac{f_{oscilador}/N_{prescaler}}{2 \cdot 2^{16}} \tag{2.1}$$

Los resultados obtenidos se muestran en la siguiente tabla

$N_{prescaler}$	f_{blink} [Hz]	T [s]
64	1,91	0,52
256	0,48	2,1
1024	0,12	8,39

Para el efecto anti-rebote del pulsador, se utilizó un pequeño *delay* después de definir el parpadeo para esperar hasta la siguiente detección.

A continuación, se muestra el diagrama de flujo del programa que explica de forma simplificada el funcionamiento del programa.

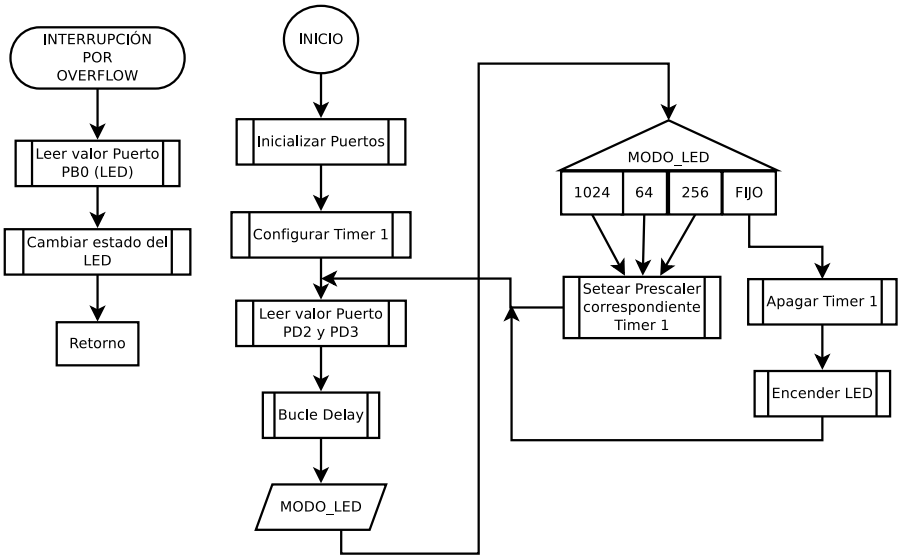

Figura 4: Diagrama de flujo.

2.2.2. Código utilizado

El código utilizado para el programa se adjunta en esta sección.

```
;
; TP4 - Timers
;
; Created: 3/7/2021 20:57:45
; Autor : Puy Gonzalo
; Padron : 99784
;

.INCLUDE "m328pdef.inc"
.INCLUDE "MACROS.inc"
.INCLUDE "DEFINES.inc"

.CSEG
.ORG 0x0000
        JMP     CONFIG
.ORG OVF1addr
        JMP     ISR_OVERFLOW

.ORG INT_VECTORS_SIZE

CONFIG:
        initSP
        initPORTS
        configTimer1

MAIN:
        CALL    READ_VALUE
        CALL    SELEC_MODO

        JMP     MAIN

.INCLUDE "LED_TIMER_CONTROL.inc"
.INCLUDE "READ.inc"
```

Listing 1: Código del programa

```
;
; DEFINES.inc
;
; Created: 6/7/2021 00:40:03
; Autor : Puy Gonzalo
; Padron : 99784
;
; Este archivo contiene .DEF y .EQU utiles para el programa.

.DEF    dummyreg = r16
.DEF    MODO_LED = r17
.DEF    CONTROL_LED = r18
```

```
13
14 .EQU     SWITCH_MASK = 0b00001100
15 .EQU     LED_MASK = 0b00000001
16 .EQU     LED_PIN = 0
17 .EQU     MODO_FIJO = 0b00000000
18 .EQU     MODO_64 = 0b00001000
19 .EQU     MODO_256 = 0b00000100
20 .EQU     MODO_1024 = 0b00001100
```

```
1  ;
2  ; MACROS.inc
3  ;
4  ; Created: 6/7/2021 00:40:03
5  ; Autor : Puy Gonzalo
6  ; Padron : 99784
7  ;
8  ; Este archivo contiene macros utiles para el programa.
9
10 ; Esta macro inicializa el stack pointer
11 ; Uso:      initSP
12 .MACRO  initSP
13             LDI dummyreg,LOW(RAMEND)
14             OUT spl,dummyreg
15             LDI dummyreg,HIGH(RAMEND)
16             OUT sph,dummyreg
17 .ENDMACRO
18
19 ; Esta macro inicializa los puertos correspondientes
20 ; Uso:      initPORTS
21 .MACRO  initPORTS
22 ; Puerto PB0 como salida
23         LDI dummyreg, (1<<PB0)
24         OUT DDRB,dummyreg
25 ; Puerto D como entrada
26         LDI dummyreg,0
27         OUT DDRD,dummyreg
28 .ENDMACRO
29
30 ; Esta macro configura la interrupcion por overflow
31 ; del Timer/Counter 1
32 ; Uso: configTimer1
33 .MACRO  configTimer1
34         LDI dummyreg,(1<<TOIE1)
35         STS TIMSK1,dummyreg
36         SEI
37 .ENDMACRO
```

```
1 ;
2 ; READ.inc
3 ;
4 ; Created: 6/7/2021 00:40:03
```

```
; Autor : Puy Gonzalo
; Padron : 99784
;

READ_VALUE:
        IN      MODO_LED,PIND
        ANDI    MODO_LED,SWITCH_MASK
        CALL    DELAY
        IN      dummyreg,PIND
        ANDI    dummyreg,SWITCH_MASK
        CP      MODO_LED,dummyreg
        BRNE    READ_VALUE
        RET

; Delay hecho con Timer0 de 10ms.
DELAY:
        LDI R20,0x64
        OUT TCNT0,R20 ;load Timer0
        LDI R20,0X05
        OUT TCCR0B,R20 ;Timer0, Normal mode, int clk, 1024
   prescaler
AGAIN:
        IN R20,TIFR0 ;read TIFR
        SBRS R20,TOV0 ;if TOV0 is set skip next instruction
        RJMP AGAIN
        LDI R20,0x00
        OUT TCCR0B,R20 ;stop Timer0
        LDI R20,(1<<TOV0)
        OUT TIFR0,R20 ;clear TOV0 flag
        RET
```

```
;
; LED_TIMER_CONTROL.inc
;
; Created: 7/7/2021 02:14:22
; Autor : Puy Gonzalo
; Padron : 99784
;

 SELEC_MODO:
        CPI     MODO_LED,MODO_FIJO
        BREQ    FIJO

        CPI     MODO_LED,MODO_64
        BREQ    M_64

        CPI     MODO_LED,MODO_256
        BREQ    M_256

        CPI     MODO_LED,MODO_1024
        BREQ    M_1024
```

```
21
22 FIJO:
23          LDI     dummyreg,0
24          STS     TCCR1B,dummyreg
25          SBI     PORTB,LED_PIN
26          JMP     RETORNO
27 M_64:
28          LDI     dummyreg, (0<<CS12) | (1<<CS11) | (1<<CS10)
29          STS     TCCR1B,dummyreg
30          JMP     RETORNO
31 M_256:
32          LDI     dummyreg, (1<<CS12) | (0<<CS11) | (0<<CS10)
33          STS     TCCR1B,dummyreg
34          JMP     RETORNO
35 M_1024:
36          LDI     dummyreg, (1<<CS12) | (0<<CS11) | (1<<CS10)
37          STS     TCCR1B,dummyreg
38          JMP     RETORNO
39
40 RETORNO:
41          RET
42
43 ; Rutina de interrupcion
44 ISR_OVERFLOW:
45          IN      CONTROL_LED,PORTB
46          LDI     dummyreg,LED_MASK
47          EOR     CONTROL_LED,dummyreg
48          OUT     PORTB,CONTROL_LED
49          RETI
```

2.3. Programa 2: PWM

Para el segundo programa se utilizará una conexión como se muestra en la figura 5. Como puede verse, la conexión es análoga al programa 1, salvo que esta vez se conecta el LED al puerto PD5.

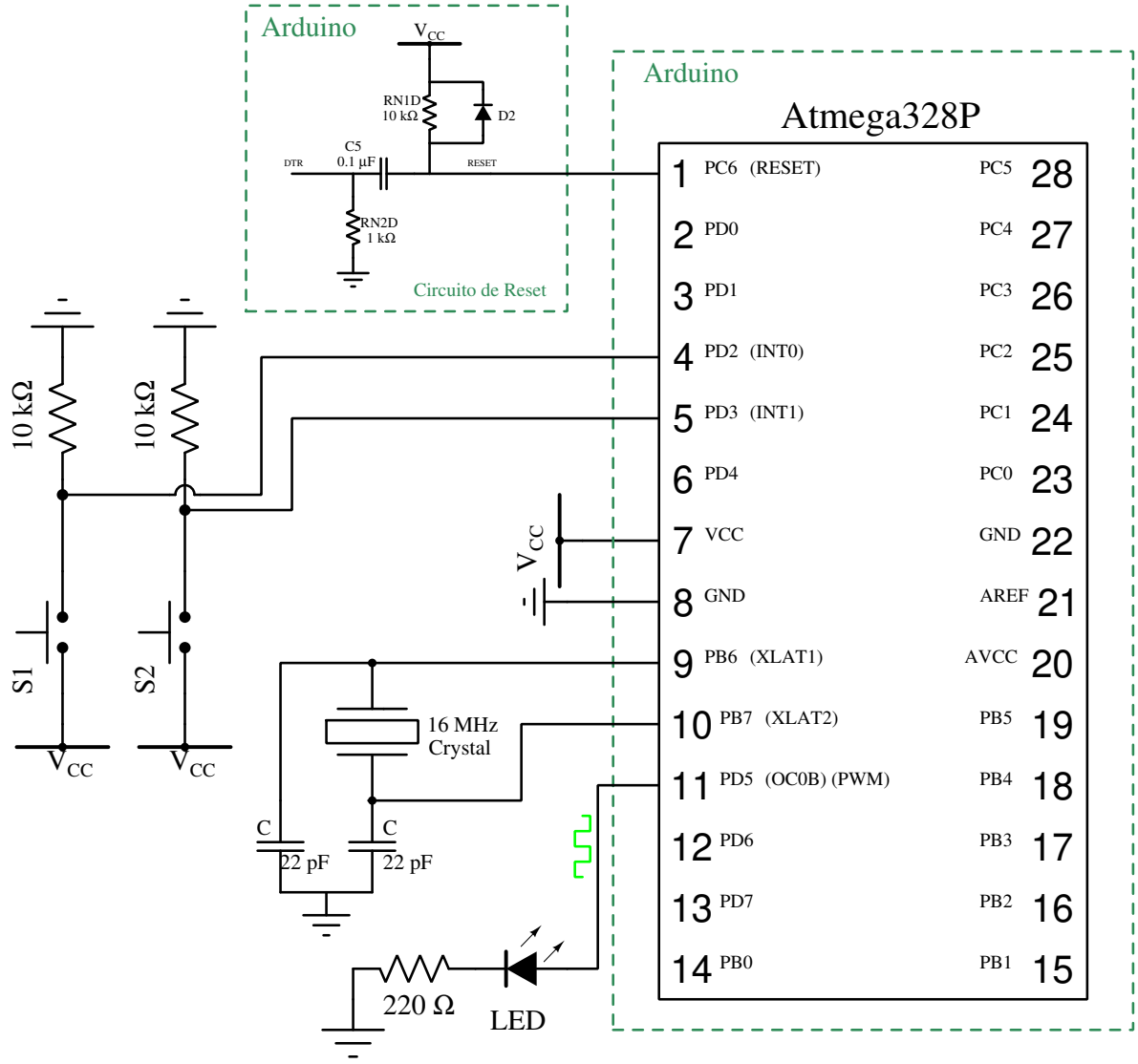

Figura 5: Conexión para el programa 2.

2.3.1. Programa a implementar

En este caso, se realizará un programa que aumente y disminuya el brillo de un LED. Para eso se usarán 2 pulsadores (UP,DOWN) y el modo PWM o modulación por ancho de pulso, que consiste en modificar el ciclo de trabajo de una señal (sin modificar su frecuencia). Con esta señal se alimentara el LED, de forma que el valor medio de la señal será proporcional al brillo del LED. A mayor ancho de pulso, más brillo.

Para llevar acabo este programa, se eligió usar el Timer0, en modo *FAST PWM* no-invertido.

Se tomará una señal con frecuencia de 60 Hz, para lograr esto se utilizo la siguiente formula

$$f = \frac{f_{oscilador}}{256 \cdot N_{prescaler}} \tag{2.2}$$

Como se eligió una frecuencia de 60 Hz, el valor de N que resulto en una frecuencia aproximada a 60 Hz fue $N = 1024$, que resulto en una frecuencia de $f = 61{,}04$ Hz. Por lo tanto se seteo este prescaler en el Timer0.

Con respecto al ciclo de trabajo, se comenzará el programa con un ciclo de trabajo de aproximadamente un 0 % (el valor mas bajo que se colocará en el registro OCR0 será 0). Luego

se incrementará (en caso de presionar el pulsador UP) de a 25 % hasta llegar aproximadamente a un 100 % (El valor que tendrá el registro OCR0 será de 252). En el caso contrario en el que se desee disminuir (presionando el pulsador DOWN) el ciclo de trabajo, se restará un 25 % al valor actual. Y así sucesivamente

Para encontrar el valor que se debe colocar en OCR0 para obtener un ciclo de trabajo del 25 % se utilizo la siguiente formula

$$\text{Duty Cycle} = \frac{\text{OCR0} + 1}{256} \cdot 100 \tag{2.3}$$

En donde reemplazando "Duty Cycle" por 25, se obtiene que OCR0 debe valer 63.

Por ultimo, se aclara que los pulsadores funcionarán por medio de interrupciones externas. Para solucionar el problema del rebote de los pulsadores, se utilizo un pequeño bucle de *delay* al comienzo de las rutinas de interrupciones.

A continuación se muestra el diagrama de flujo del programa

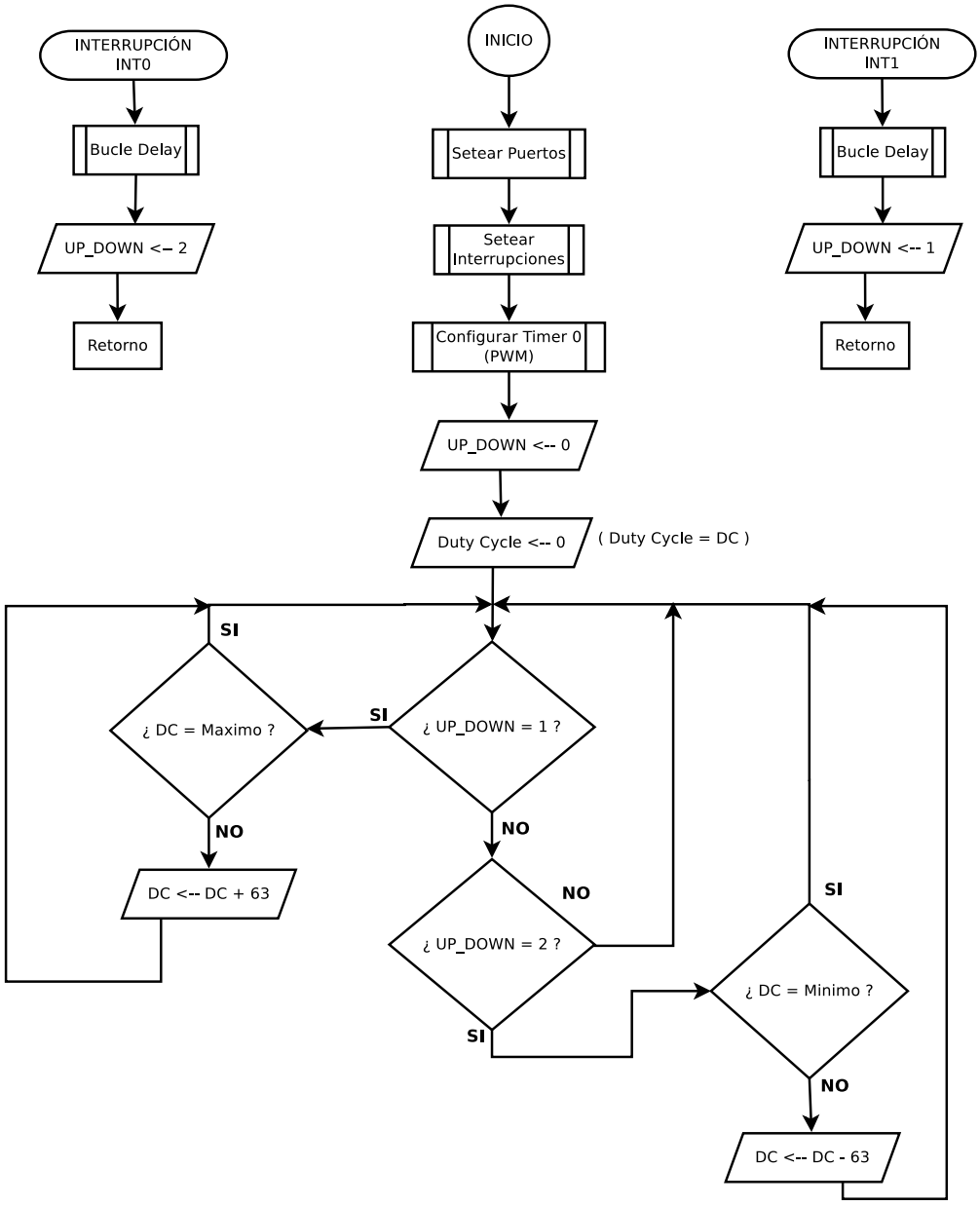

Figura 6: Diagrama de flujo.

2.3.2. Código del programa

El código utilizado para el programa 2 se adjunta a continuación

```
;
; TP4 - PWM
;
; Created: 3/7/2021 20:57:45
; Autor : Puy Gonzalo
; Padron : 99784
;

.include "m328pdef.inc"
.include "MACROS2.inc"
.include "DEFINES2.inc"

.CSEG
.ORG 0x0000
        JMP     CONFIG

.ORG INT0addr
        JMP     isr_DOWN

.ORG INT1addr
        JMP     isr_UP

.ORG INT_VECTORS_SIZE

CONFIG:
        initSP
        initPORTS
        setInt
        configTimer0

MAIN:
        CLR     UP_DOWN
        LDI     DC,0
        OUT     OCR0B,DC
DC_CONTROL:
        SBRC    UP_DOWN,0
        CALL    UP
        SBRC    UP_DOWN,1
        CALL    DOWN
        JMP     DC_CONTROL

.INCLUDE    "INTERRUPCIONES.inc"
.INCLUDE    "DCCONTROL.inc"
```

Listing 2: Código del programa

```
;
; Defines2.inc
```

```
;
; Created: 6/7/2021 00:40:03
; Autor : Puy Gonzalo
; Padron : 99784
;
; Este archivo contiene .DEF y .EQU utiles para el programa.

.EQU    MAXIMO_DC = 252
.EQU    MINIMO_DC = 0
.EQU    DC_25 = 63
.EQU    DC_UP = 1
.EQU    DC_DOWN = 2
.equ    PIN_SDOWN = 2
.equ    PIN_SUP = 3

.DEF    dummyreg = r16
.DEF    DC = r17
.DEF    UP_DOWN = r18
```

```
;
; MACROS2.inc
;
; Created: 6/7/2021 00:40:03
; Autor : Puy Gonzalo
; Padron : 99784
;
; Este archivo contiene macros utiles para el programa.

; Esta macro inicializa el stack pointer
; Uso:      initSP
.MACRO  initSP
            LDI dummyreg,LOW(RAMEND)
            OUT spl,dummyreg
            LDI dummyreg,HIGH(RAMEND)
            OUT sph,dummyreg
.ENDMACRO

; Esta macro inicializa los puertos correspondientes
; Uso:      initPORTS
.MACRO  initPORTS
; Puerto PD2, PD3 como entrada (interrupcion) Puerto PD5 como salida (OC0B).
        LDI dummyreg,(0<<PD2) | (0<<PD3) | (1<<PD5)
        OUT DDRD,dummyreg
.ENDMACRO

; Esta macro setea la interrupcion externa en INT0 e INT1
; Uso: setINT0
.macro setInt
        LDI dummyreg, (1<<ISC11) | (1<<ISC10) | (1<<ISC01) | (1<<ISC00)
```

```
31          STS EICRA,dummyreg
32          LDI dummyreg, (1<<INT1) | (1<<INT0)
33          OUT EIMSK,dummyreg
34          SEI
35  .endmacro
36
37  ; Esta macro configura el Timer/Counter 0
38  ; Modo Fast PWM no-invertido con prescaler de 1024
39  ; Uso: configTimer0
40  .MACRO  configTimer0
41          LDI dummyreg, (1<<COM0B1) | (0<<COM0B0) | (1<<WGM01) |
        (1<<WGM00)
42          OUT TCCR0A,dummyreg
43          LDI dummyreg, (0<<WGM02) | (1<<CS02) | (0<<CS01) | (1<<
        CS00)
44          OUT TCCR0B,dummyreg
45  .ENDMACRO
```

```
1   ;
2   ; DCCONTROL.inc
3   ;
4   ; Created: 7/7/2021 22:47:28
5   ; Autor : Puy Gonzalo
6   ; Padron : 99784
7   ;
8
9    UP:
10          CPI     DC,MAXIMO_DC
11          BREQ    UP_RETORNO
12
13          LDI     dummyreg, DC_25
14          ADD     DC,dummyreg
15          OUT     OCR0B,DC
16          CLR     UP_DOWN
17
18  UP_RETORNO:
19          RET
20
21  DOWN:
22          CPI     DC,MINIMO_DC
23          BREQ    DOWN_RETORNO
24
25          LDI     dummyreg,DC_25
26          SUB     DC,dummyreg
27          OUT     OCR0B,DC
28          CLR     UP_DOWN
29
30  DOWN_RETORNO:
31          RET
```

```
;
; INTERRUPCIONES.inc
;
; Created: 7/7/2021 22:47:55
; Autor : Puy Gonzalo
; Padron : 99784
;



 isr_DOWN:
        NOP
        CALL    DELAY
        NOP
        SBIS    PIND,PIN_SDOWN
        JMP     RETORNO_isrDOWN

        LDI UP_DOWN,DC_DOWN

RETORNO_isrDOWN:
        SBI     EIFR,0
        RETI

isr_UP:
        NOP
        CALL    DELAY
        NOP
        SBIS    PIND,PIN_SUP
        JMP     RETORNO_isrUP

        LDI UP_DOWN,DC_UP

RETORNO_isrUP:
        SBI     EIFR,0
        RETI


;Delay de 10 ms
DELAY:
    ldi  r21, 208
    ldi  r22, 202
L1:
    dec  r22
    brne L1
    dec  r21
    brne L1
    nop
    ret
```

3. Resultados

Finalmente, se logró que ambos programas funcionaran correctamente y sin complicaciones. Para mostrar esto, se adjunta un enlace a un vídeo, en donde se muestra el funcionamiento de los programas.

4. Conclusiones

Como conclusión, se puede decir que se logró entender el manejo de Timers/Counters que se ofrece en los microcontroladores de la familia AVR, las diferentes configuraciones que estos tienen y las interrupciones por overflow. Gracias a esto, se logró hacer parpadear un LED sin la necesidad de generar retardos como se hizo en el primer trabajo.

En cuanto al programa donde se utilizo PWM, se entendió el manejo de la energía utilizada. Si bien en este caso solo se hizo brillar con mas o menos intensidad un LED, es posible ver que PWM tiene diversas utilidades, como por ejemplo en el control de motores de corriente continua.

Por ultimo, se puede destacar que se reforzó conocimiento sobre los problemas que pueden generar el efecto rebote de los pulsadores mecánicos y como este problema puede solucionarse por software, sin la necesidad de añadir componentes extras a la conexión.