Universidad de Buenos Aires
Facultad de Ingeniería

86.07 - Laboratorio de microprocesadores

TP4: PWM

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 aumente/disminuya el brillo de un LED.

Índice

1. PWM: Control de brillo

1.1. Introducción

Modulación por ancho de pulso o PWM (Pulse Width Modulation) de una señal es cuando se modifica el ciclo de trabajo o el ancho del pulso de una señal periódica. Uno de los usos del PWM, entre muchos otros, es controlar la cantidad de energía, en este caso el voltaje promedio es mayor conforme aumenta el ciclo de trabajo. En la imagen se puede observar, que el período de la señal permanece fijo, por lo tanto, la frecuencia también, solamente cambia el ciclo de trabajo, en la primera se observa que el ciclo de trabajo es de aproximadamente 50 % lo cual nos indica que es el porcentaje de voltaje promedio entregado a la carga. El PWM se puede utilizar en varias cosas, como el control de la velocidad de motores de DC, la posición de un servomotor, fuentes conmutadas, entre otras cosas más.

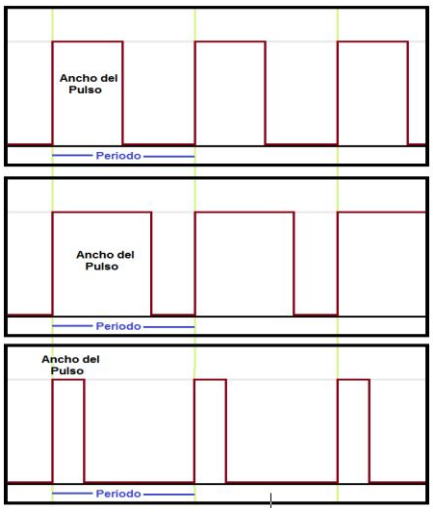

Figura 1: Modulación por ancho de pulso - PWM

El siguiente programa consiste en aumentar y disminuir el brillo de un LED. Para eso se dispone de 2 pulsadores (UP, DOWN) y un LED como indica el esquema del programa anterior (Timers). Para la variación de brillo se usa PWM. Con esta señal se alimenta el LED, de forma que el valor medio de la señal es proporcional al brillo del LED, a mayor ancho de pulso, más brillo.

Para la configuración física se necesitaron 2 resistencias de 10k Ω, 1 de 220 Ω, 2 pulsadores, un LED, cables para el conexionado, un protoboard y un Arduino UNO. Se conecto de la siguiente manera:

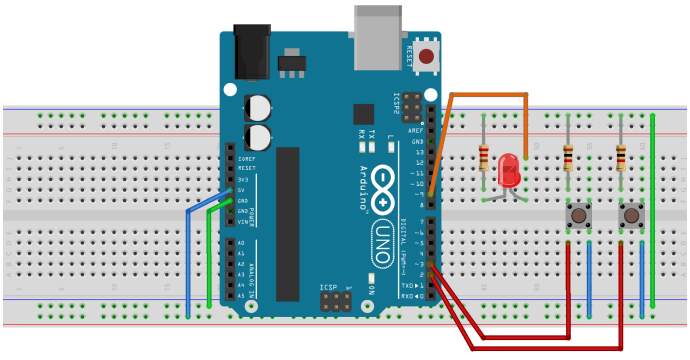

Figura 2: Diagrama esquemático

1.2. Desarrollo

En cuanto al diagrama de bloques es igual al del ejercicio anterior, se utiliza la computadora como fuente y para programar el algoritmo, se utilizan los switches para las interupciones y ambos componentes interactuan con el arduino y ese con el LED.

En cuanto al diagrama de flujos se utilizó la siguiente lógica. El programa principal es solo un loop del main que espera que se presione alguna de las interrupciones para modificar el brillo:

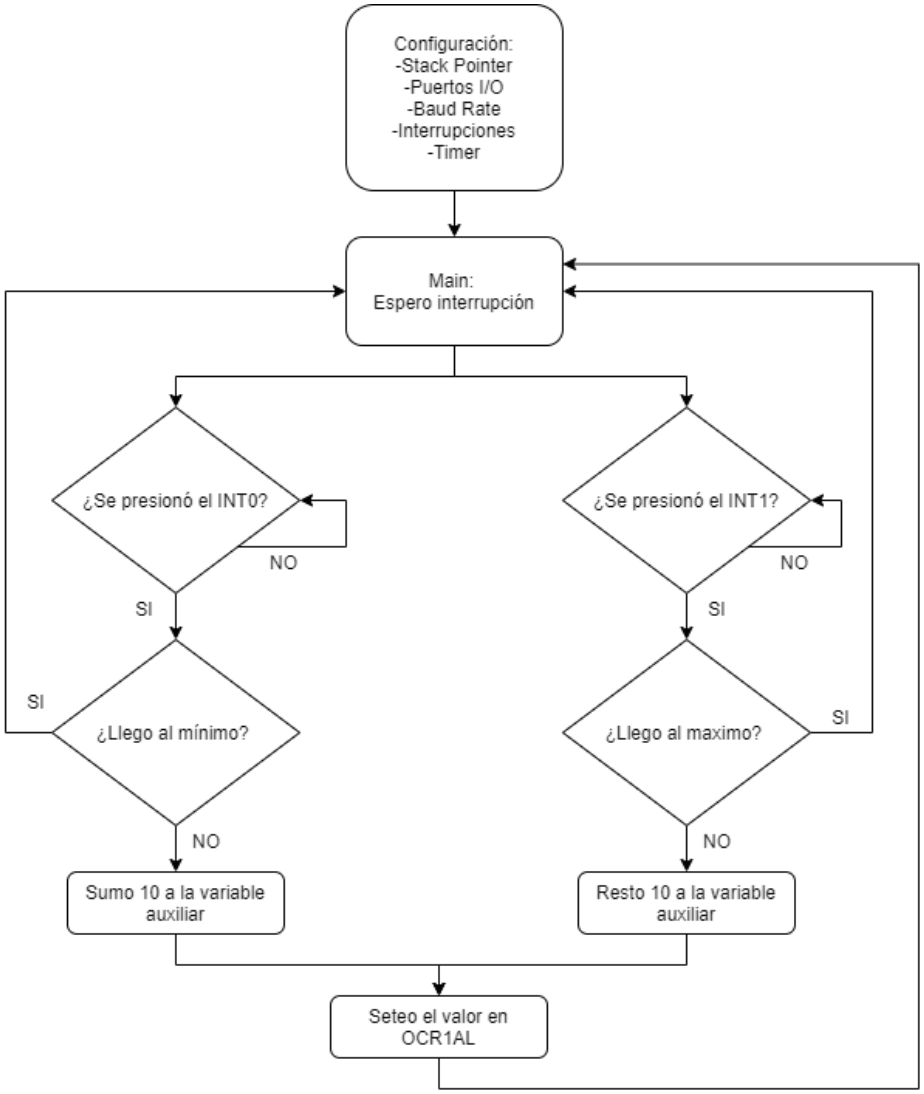

Figura 3: Diagrama de flujo

1.2.1. Macros utilizadas

```
;-----------------------------------------------------------------------------------------------
;                                                                   Constantes y registros
;-----------------------------------------------------------------------------------------------

;Registros auxiliares
.def        aux1 = R17
.def        aux2 = R18


;-----------------------------------------------------------------------------------------------
;                                                                   Configuraciones puertos
;-----------------------------------------------------------------------------------------------

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

;Configuracion de las interrupciones --------------------------------------------------------
.macro configINT
        clr                 aux1
        ldi                 aux1,(1 << ISC11 | 0 << ISC10 | 1 << ISC01 | 0 << ISC00 )
        sts                 EICRA,aux1

        ldi                 aux1,  (1<<INT0)|(1<<INT1)
        out                 EIMSK,aux1

        sei
.endmacro

;Configuracion del timer --------------------------------------------------------------------
.macro configtimer
        ldi                 aux1, ( 1<<CS10 | 1<<WGM12)
        sts                 TCCR1B, aux1
        ldi                 aux1, ( 1<<COM1A1 |1<<WGM10)
        sts                 TCCR1A, aux1
.endmacro
;-----------------------------------------------------------------------------------------------
;                                                                   Inicializaciones
;-----------------------------------------------------------------------------------------------

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

Listing 1: macros.inc

1.2.2. Código principal

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

;Interrupciones:
.org INT0addr
        rjmp        INT0action
.org INT1addr
        rjmp        INT1action

.cseg
.org 0x0000
        jmp                 config

.org INT_VECTORS_SIZE

;-------------------------------------------------------------------
```

```
;                               Configuracion de los puertos
;-----------------------------------------------------------------------

config:
        ; Inicializo el stack
        initSP

        ; Declaro PortB como salida (LED)
        configport DDRB,0xff

        ;Declaro PortD como entrada (pulsadores)
        configport DDRD,0x00

        ; Configuro interrupciones
        configINT

        ; Configuro timer
        configtimer

;-----------------------------------------------------------------------
;                                                               Main
;-----------------------------------------------------------------------
;El loop principal no hace nada, todo lo hacen las interrupciones
main:
        rjmp        main

;-----------------------------------------------------------------------
;                                     Rutinas de interrupción
;-----------------------------------------------------------------------
;Incrementa
INT0action:
        ;Verifico el maximo
        cpi                 aux1, 0xff
        breq        INT0end
        ;Aumenta en 10
        ldi                 aux2, 10
        add                 aux1, aux2
        sts                 OCR1AL, aux1

        INT0end:
                reti

;Disminuye
INT1action:
        ;Verifico el minimo
        cpi                 aux1, 0x00
        breq        INT1end
        ;Disminuye en 10
        ldi                 aux2, 10
        sub                 aux1, aux2
        sts                 OCR1AL, aux1

        INT1end:
                reti
```

1.3. Video funcionando

https://drive.google.com/file/d/1Y3N3f0O32t8UWrjf8A2XG4U9hZPev18c/view?usp=sharing

1.4. Repositorio Github

https://github.com/fiuba-labo-de-micro-miercoles/2021_1c_trabajos_practicos-lmberard/tree/master/TP4_PWM

2. Bibliografía

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

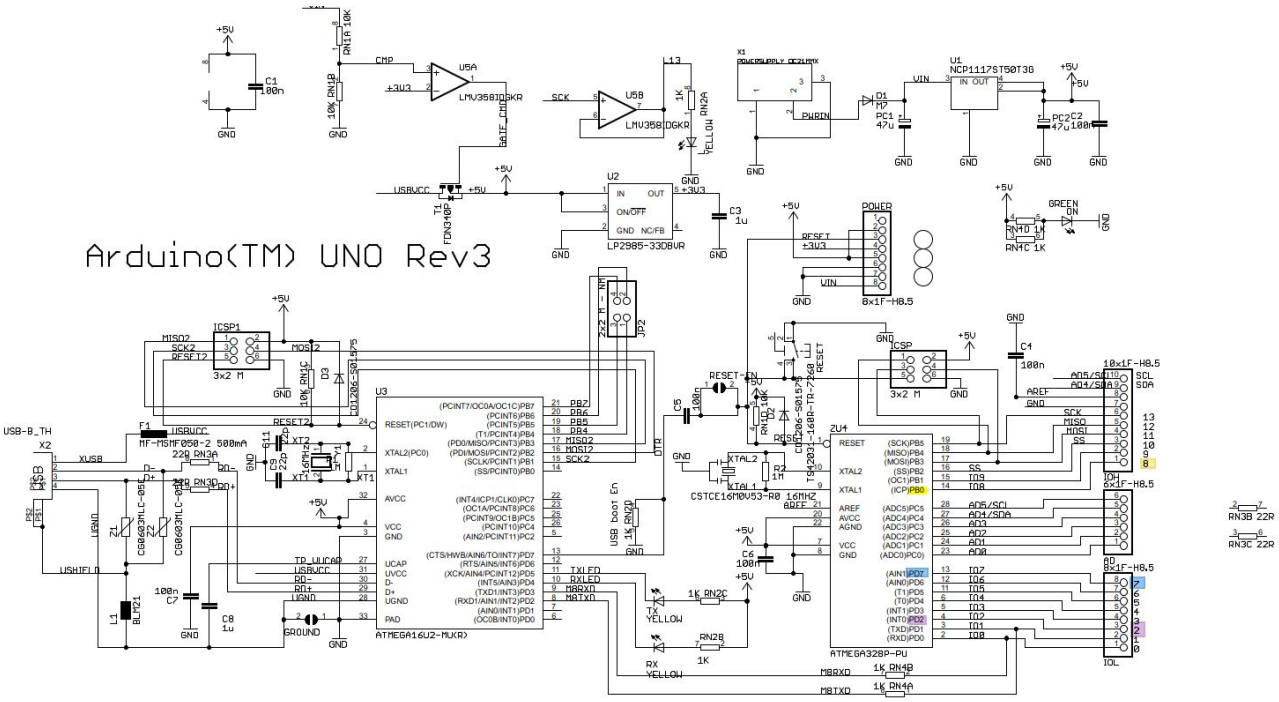

Figura 4: Esquemático Arduino UNO