

(6609) Laboratorio de Microcomputadoras

Proyecto:  
( *tp7 PWM* )

<b>Profesor:</b>	<b>Ing. Guillermo Campiglio</b>
<b>Cuatrimestre / Año:</b>	1ro/2020
<b>Turno de clases prácticas:</b>	Miércoles
<b>Jefe de Trabajos Prácticos:</b>	Pedro Martos
<b>Docente guía:</b>	

Autores			Seguimiento del proyecto									
Nombre	Apellido	Padrón										
Cristian	Simonelli	87879										

**Observaciones:**

---

---

---

---

---

---

---

---

---

---

<b>Fecha de aprobación</b>

<b>Firma J.T.P.</b>

<b>COLOQUIO</b>	
<b>Nota final</b>	
<b>Firma Profesor</b>	

<b>Objetivo:</b>	<b>2</b>
<b>Desarrollo.</b>	<b>2</b>
PWM	2
Modulación por ancho de pulso	2
TCCR0A – Timer/Counter Control Register A:	2
TCCR0B – Timer/Counter Control Register B .	3
<b>Listado de componentes:</b>	<b>5</b>
<b>Diagrama en bloques:</b>	<b>6</b>
<b>Circuito esquemático:</b>	<b>6</b>
<b>Diagrama de flujos:</b>	<b>7</b>
<b>Rutina Timer overflow.</b>	<b>7</b>
<b>Código:</b>	<b>8</b>
<b>Resultado:</b>	<b>10</b>
<b>Conclusiones:</b>	<b>10</b>

## Objetivo:

Controlar el brillo de un led a través del uso de un PWM.

## Desarrollo.

Utilizando 2 switches se modifica el duty cycle del pwm logrando así controlar el brillo de un led.

## PWM

### Modulación por ancho de pulso

Consiste en modificar el ciclo útil de una señal, sin variar su frecuencia, solo cambiando el % en estado alto/estado bajo.

De esta forma se logra modificar la potencia entregada a un dispositivo, ya sea para efectuar un trabajo o alguna comunicación.

Se utiliza en casos en donde la respuesta del dispositivo dependa de la tensión aplicada, por ejemplo el par motor de un motor eléctrico.

En este caso la potencia dependerá del valor medio de las señal enviada por el pwm.

### TCCR0A – Timer/Counter Control Register A:

En este registro se elige la forma de activación de la interrupción.

En este caso se utiliza interrupción por flanco, pero podría ser por cambio de valor.

Bit	7	6	5	4	3	2	1	0	
0x24 (0x44)	COM0A1	COM0A0	COM0B1	COM0B0	–	–	WGM01	WGM00	TCCR0A
Read/Write	R/W	R/W	R/W	R/W	R	R	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

## Fast pwm

Mode	WGM02	WGM01	WGM00	Timer/Counter Mode of Operation	TOP	Update of OCRx at	TOV Flag Set on <sup>(1)(2)</sup>
0	0	0	0	Normal	0xFF	Immediate	MAX
1	0	0	1	PWM, phase correct	0xFF	TOP	BOTTOM
2	0	1	0	CTC	OCRA	Immediate	MAX
3	0	1	1	Fast PWM	0xFF	BOTTOM	MAX
4	1	0	0	Reserved	–	–	–
5	1	0	1	PWM, phase correct	OCRA	TOP	BOTTOM
6	1	1	0	Reserved	–	–	–
7	1	1	1	Fast PWM	OCRA	BOTTOM	TOP

WGM01 = WGM02 = 1

## Clear OC0B on compare match

COM0B1	COM0B0	Description
0	0	Normal port operation, OC0B disconnected.
0	1	Toggle OC0B on compare match
1	0	Clear OC0B on compare match
1	1	Set OC0B on compare match

COM0B1 = 1,

## TCCR0B – Timer/Counter Control Register B.

Bit	7	6	5	4	3	2	1	0	
0x25 (0x45)	FOC0A	FOC0B	–	–	WGM02	CS02	CS01	CS00	TCCR0B
Read/Write	W	W	R	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

No prescaling. (es necesario setearlo para prender el pwm)

**Table 14-9. Clock Select Bit Description**

CS02	CS01	CS00	Description
0	0	0	No clock source (Timer/Counter stopped)
0	0	1	clk <sub>IO</sub> /(no prescaling)
0	1	0	clk <sub>IO</sub> /8 (from prescaler)
0	1	1	clk <sub>IO</sub> /64 (from prescaler)
1	0	0	clk <sub>IO</sub> /256 (from prescaler)
1	0	1	clk <sub>IO</sub> /1024 (from prescaler)
1	1	0	External clock source on T0 pin. Clock on falling edge.
1	1	1	External clock source on T0 pin. Clock on rising edge.

Como rutina de anti bounce se utilizó el siguiente método.

Se hace polling de los switches cada vez que hay overflow del timer 1.

El valor de prescaler se fue modificando hasta que la experiencia de uso parecía orgánica. Esto es, el tiempo que hay que mantener el botón apretado para que se detecte el dato parece cómodo para el uso y no se detectan rebotes.

Cada vez que se detecta una señal de switch se modifica el valor del registro OCR0B (Output Compare Unit).

Este registro es usado por el pwm para encender o apagar la salida. Cuando el valor del del timer matchea con el valor OCR0B, la salida se pone en 0 u el contador se resetea.

Cuanto mayor sea el valor de OCR0B mayor el estado alto en la salida del pwm.

## Listado de componentes:

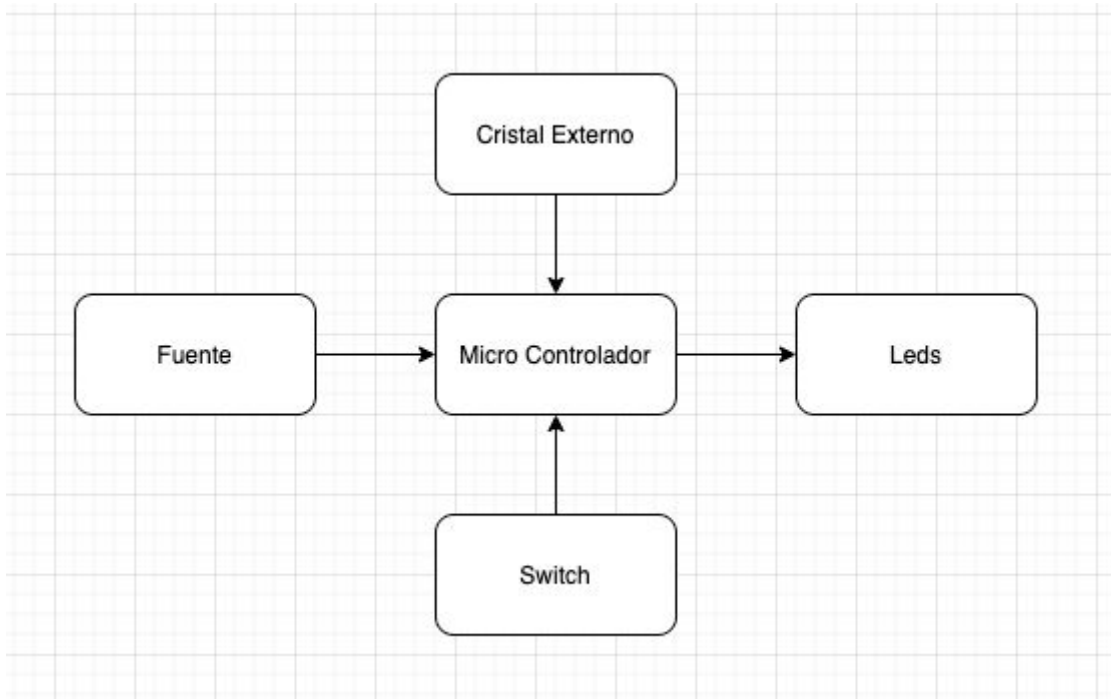
Placa arduino UNO Atmega 328p \$659 aprox 10 usd.

1 led (pack de 10) \$70.

1 resistencias 220 ohm 1/8w 1% \$50.

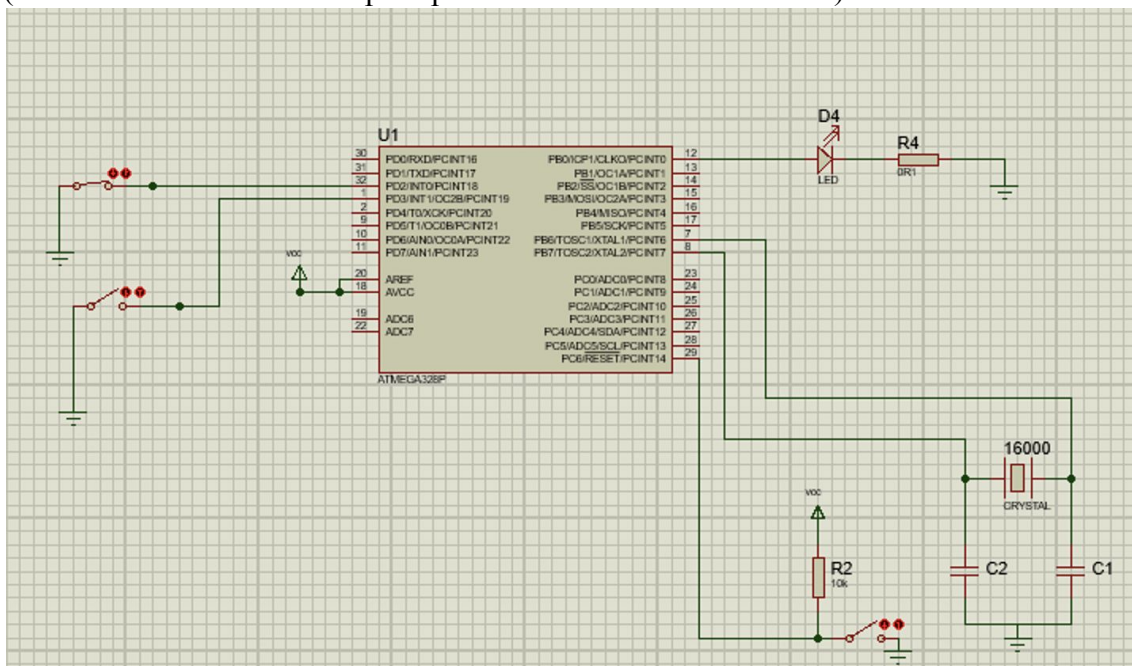
2 switch \$10.

## Diagrama en bloques:



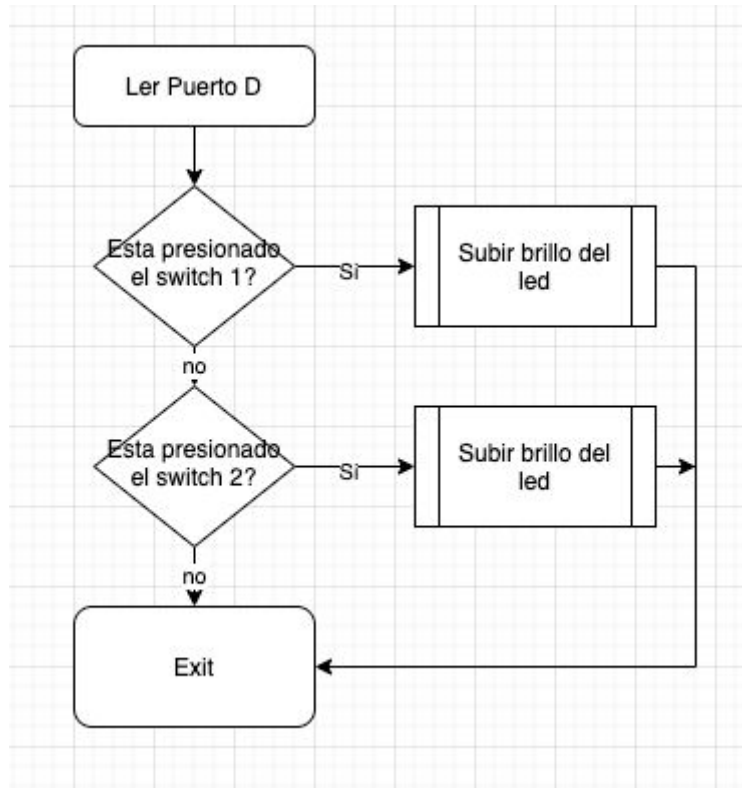
## Circuito esquemático:

(Se utilizaron resistencia de pullup internas del microcontrolador)



## Diagrama de flujos:

### Rutina Timer overflow.





## Código:

```
.include "m328pdef.inc"

.def dummyreg = r21

.cseg
.org 0x0000
        jmp     configuracion
.org OVFladdr
        jmp     isr_timovfl

.org INT_VECTORS_SIZE
configuracion:
; ram init
        ldi     dummyreg, low(RAMEND)
        out     spl, dummyreg
        ldi     dummyreg, high(RAMEND)
        out     sph, dummyreg

; port init
        ldi     dummyreg, 0xf3                ; Port D 2/3
        out     DDRD, dummyreg
        ldi     dummyreg, 0x0c                ; Pullups 2/3
        out     PORTD, dummyreg

; timer init
        ldi     dummyreg, (1 << TOIE1)        ; set overflow
        sts     TIMSK1, dummyreg
        ldi     dummyreg, (1 << CS10 | 1 << CS11)
        sts     TCCR1B, dummyreg

; pwm init
        ldi     dummyreg, (1 << COM0B1 | 1 << WGM01 | 1 << WGM00) ; fast pwm non
        out     TCCR0A, dummyreg
        ldi     dummyreg, (1 << CS00)          ; Timer 0 on,
        out     TCCR0B, dummyreg
        ldi     dummyreg, 0x01                ; compare
        out     OCR0B, dummyreg

        sei

main:
```

```

        jmp main

; Polling every that timer1 overflows to prevent switch bounces
ISR_TIMOVF1:
        in     r25, PIND                ; get PIND
        mov    r26, r25                ; preserv PIND
        value
        andi   r26, 0x04                ; is portd.2
        pressed ?
        breq   led_up                  ; is so,
        increase led's brigh
        mov    r26, r25                ; portd.2 was
        no pressed
        andi   r26, 0x08                ; is portd.3
        pressed ?
        breq   led_down                ; is so,
        decrease led's brigh
exit_isr:    reti

; The idea behind this is to change the value that tcntc in compared with
led_up:
        in     r24, OCR0B                ; get OCR0B
        original value
        lsl    r24                      ; multiply
        this by 2
        brcs   exit_isr                ; is carry
        seted? if this is the case, dont update the value
        out    OCR0B, r24              ; if not, set
        the new OCR0B value
        reti                            ;

led_down:
        in     r24, OCR0B                ; get OCR0B
        original value
        lsr    r24                      ; divided this
        by 2
        breq   exit_isr                ; if this is 0
        dont update the value
        out    OCR0B, r24              ; update the
        value
        reti

```

## **Resultado:**

Se controla la el brillo del led con el pwm

## **Conclusiones:**

Se podría lograr el mismo efecto controlando la salida de un puerto utilizando valores de algún registro y aplicando una lógica dentro del programa para controlar el ciclo de carga.

Pero la facilidad del uso del pwm lo hace muy útil, para esta aplicación.

Solo hace falta cambiar el valor umbral, con el cual se compara el ciclo de carga.