



# FACULTAD DE INGENIERIA

Universidad de Buenos Aires

(6609) LABORATORIO DE MICROCOMPUTADORAS

Proyecto:  
*TP3 - Timers y PWM*

<b>Profesor:</b>	<b>Ing. Guillermo Campiglio</b>
<b>Cuatrimestre / Año:</b>	1er cuatrimestre - 2022
<b>Turno de clases prácticas:</b>	Jueves - 19 a 22
<b>Jefe de Trabajos Prácticos:</b>	Graciela Ratto
<b>Docente guía:</b>	Gavinowich Gabriel

Autores			Seguimiento del proyecto									
Nombre	Apellido	Padrón										
Violeta	Perez Andrade	101456										

Observaciones:

---

---

---

---

---

---

---

---

---

---

<b>Fecha de aprobación</b>

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

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

# Índice

<b>1. Objetivo del proyecto</b>	<b>3</b>
<b>2. Descripción del proyecto</b>	<b>3</b>
2.1. Parte 1 - Timers . . . . .	3
2.2. Parte 2 - PWM . . . . .	3
<b>3. Diagrama de conexiones en bloques</b>	<b>3</b>
3.1. Parte 1 - Timers . . . . .	3
3.2. Parte 2 - PWM . . . . .	4
<b>4. Circuito esquemático</b>	<b>4</b>
4.1. Parte 1 - Timers . . . . .	4
4.2. Parte 2 - PWM . . . . .	5
<b>5. Listado de componentes y tabla de gastos</b>	<b>5</b>
<b>6. Diagrama de flujo</b>	<b>6</b>
6.1. Parte 1 - Timers . . . . .	6
6.2. Parte 2 - PWM . . . . .	7
<b>7. Resultados</b>	<b>7</b>
7.1. Parte 1 - Timers . . . . .	7
7.2. Parte 2 - PWM . . . . .	7
<b>8. Conclusiones</b>	<b>8</b>
<b>9. Apéndice: Código del programa</b>	<b>9</b>
9.1. Parte 1 - Timers . . . . .	9
9.2. Parte 2 - PWM . . . . .	13

## 1. Objetivo del proyecto

El objetivo del presente trabajo es familiarizarse con el manejo de, por una parte, los distintos timers del microcontrolador y su funcionamiento y, por otra, su capacidad para generar eventos cada un determinado tiempo o señales cuadradas moduladas en ancho de pulso (PWM).

## 2. Descripción del proyecto

En este trabajo práctico se desarrollaron dos programas en lenguaje ensamblador.

### 2.1. Parte 1 - Timers

En este primer programa los temporizadores son utilizados en modo normal. El programa consiste en generar un evento cada cierto determinado tiempo. El evento particularmente es parpadear un LED. El mismo es detectado mediante interrupciones, que al detectarse se encenderá o apagará el LED dependiendo de su estado anterior.

Por otra parte, se cuenta con dos pulsadores los cuales son utilizados para cambiar la configuración del prescaler del timer para dependiendo como se presionen modificar la frecuencia de parpadeo del LED.

Para este propósito se utilizó el temporizador uno mientras que el 0 y el 2 fueron utilizados para contar el tiempo de antirrebote al presionar los pulsadores.

### 2.2. Parte 2 - PWM

En este segundo programa, se utilizan los pulsadores para mover un servomotor entre  $0^\circ$  y  $180^\circ$  aumentando o decrementando el ángulo dependiendo de que pulsador se presione.

Para esto, el timer 1 genera una señal cuadrada a la cual se le varía el ancho de pulso dependiendo de que pulsador se presione mientras que los timers 0 y 2 también fueron utilizados para antirrebote.

## 3. Diagrama de conexiones en bloques

En el siguiente diagrama se puede observar cómo se conectaron los distintos bloques del trabajo

### 3.1. Parte 1 - Timers

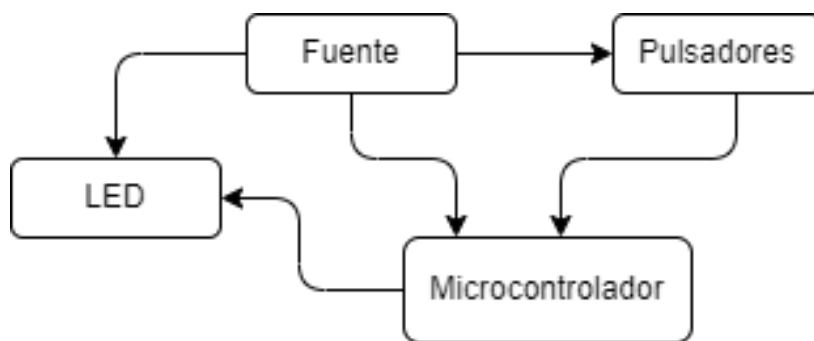


Figura 1: Diagrama de conexión en bloques - Parte 1 - Timers

### 3.2. Parte 2 - PWM

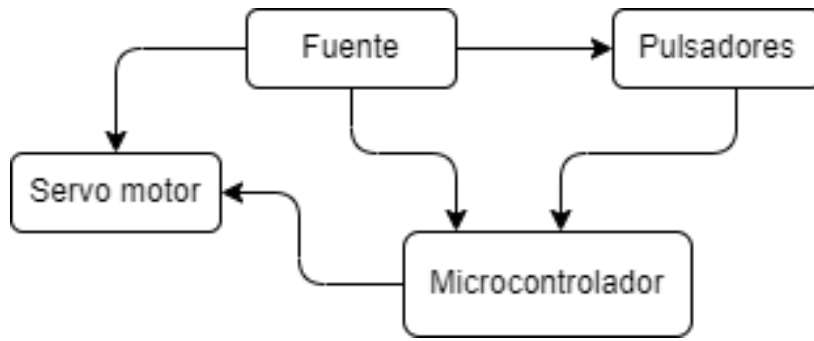


Figura 2: Diagrama de conexión en bloques - Parte 2 - PWM

## 4. Circuito esquemático

Se incluyen los circuitos esquemáticos del trabajo

### 4.1. Parte 1 - Timers

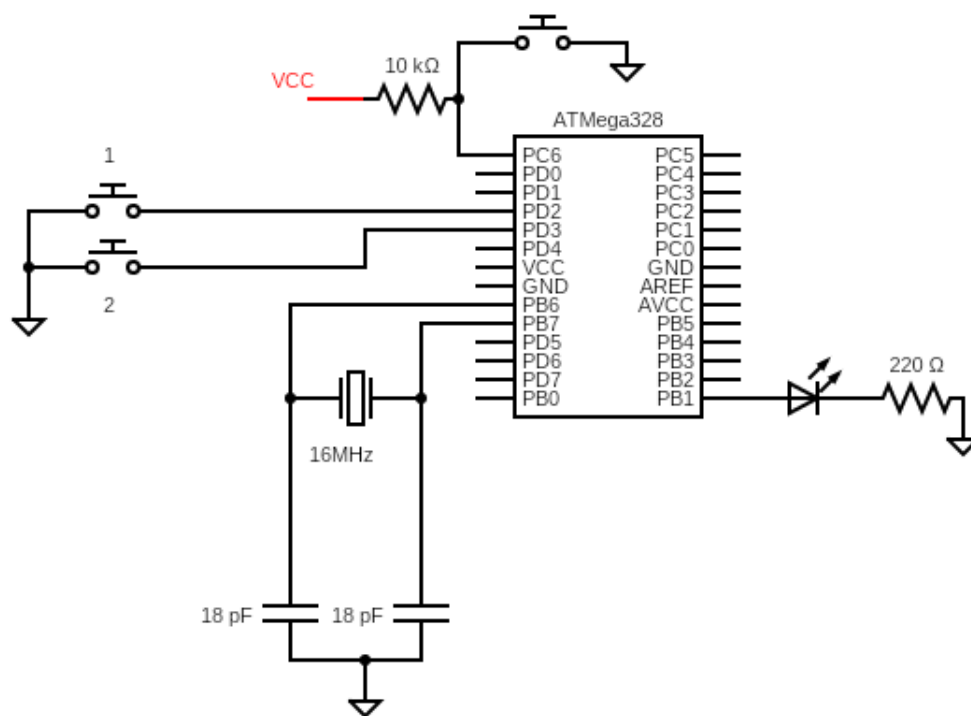


Figura 3: Circuito esquemático - Parte 1 - Timers

## 4.2. Parte 2 - PWM

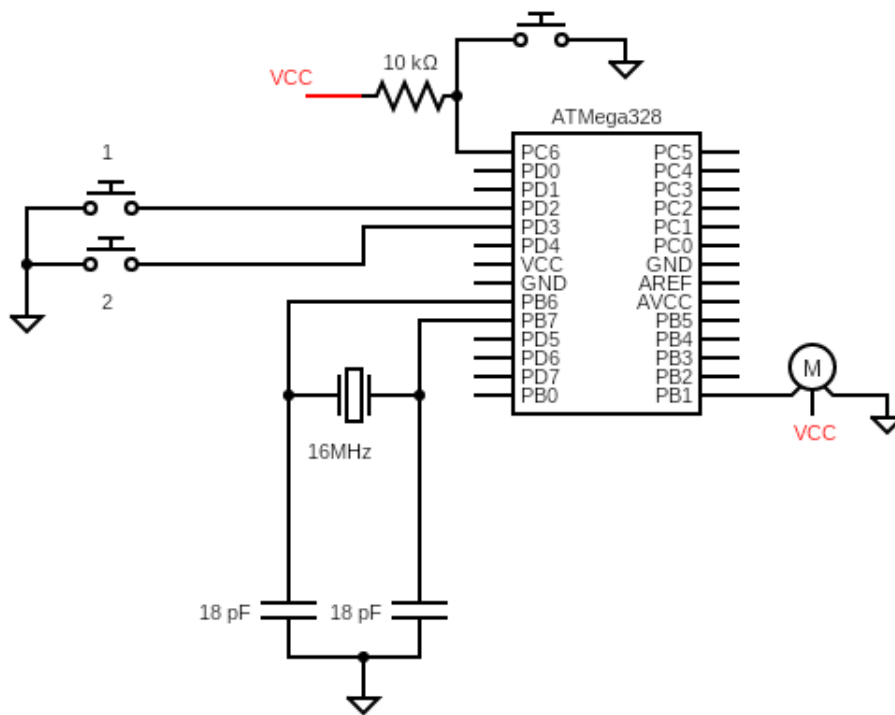


Figura 4: Circuito esquemático - Parte 2 - PWM

## 5. Listado de componentes y tabla de gastos

Listado de componentes		
Componente	Cantidad	Precio uni.
Servomotor	1	\$ 642,20
Touch Switch 5mm	2	\$ 22,28
Resistencia 220 Ω	1	\$ 35,52
LED	1	\$ 9,20
Atmega 328P	1	\$ 878.90
Tira de 10 cables	1	\$ 237,90
Total		\$ 1848,28

## 6. Diagrama de flujo

### 6.1. Parte 1 - Timers

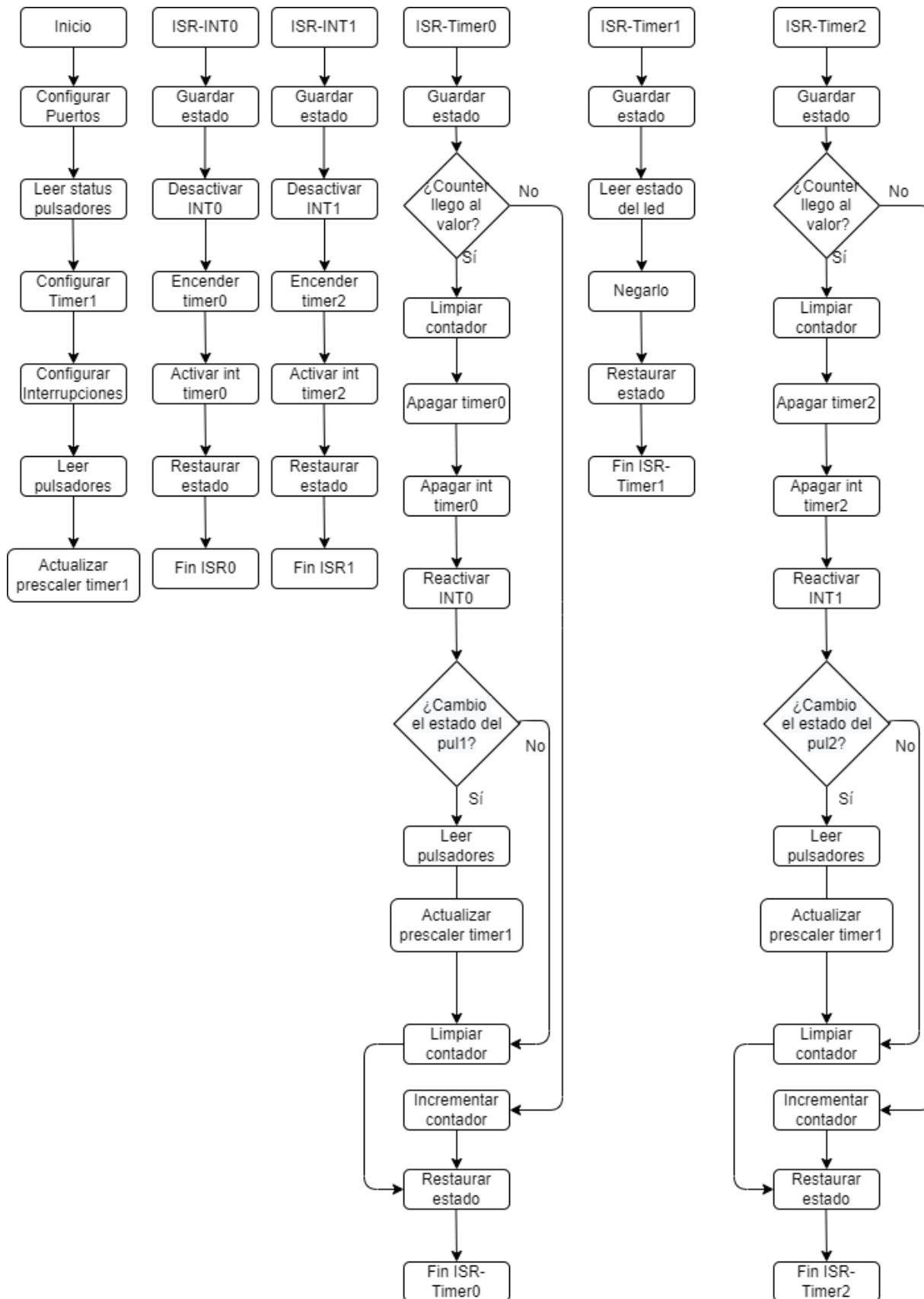


Figura 5: Diagrama de flujo - Parte 1 - Timers

## 6.2. Parte 2 - PWM

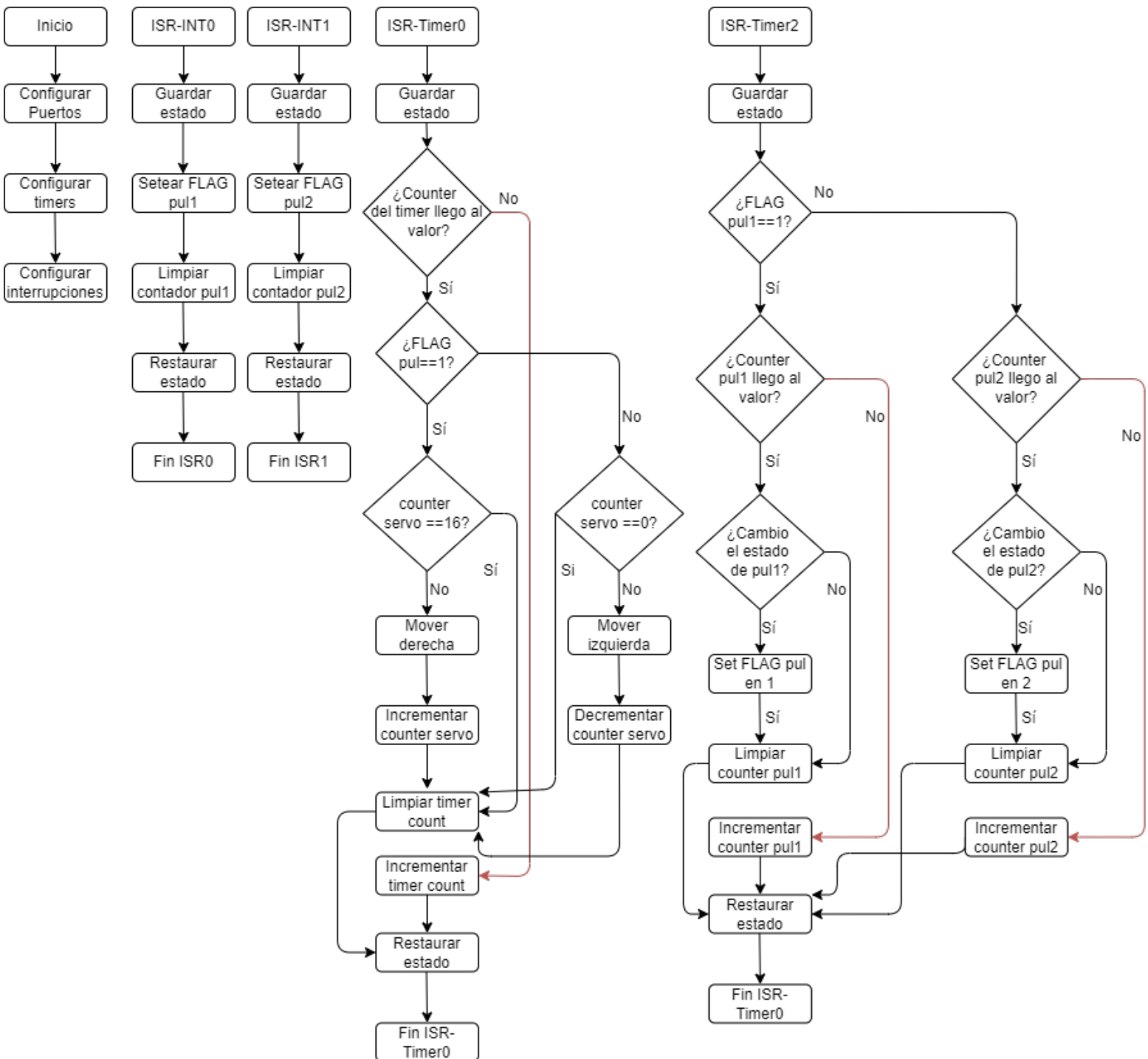


Figura 6: Diagrama de flujo - Parte 2 - PWM

## 7. Resultados

### 7.1. Parte 1 - Timers

Los resultados fueron los esperados, las cuentas dieron bien y se pudo ver esto reflejado en el período que tenía el LED al momento de parpadear con los distintos estados de los pulsadores.

### 7.2. Parte 2 - PWM

En esta parte los resultados también fueron los esperados, el servo motor se movía hacia ambos lados frenando en los extremos.

## 8. Conclusiones

El desarrollo del trabajo permitió aprender a utilizar los timers del microcontrolador y sus distintas funciones. Como lo puede ser meramente de contador (como se utilizó para contar el debounce) y también para poder generar ondas cuadradas de distinto pulso.

Por otra parte, también se pudo observar que el tiempo entre distintos eventos depende de la frecuencia a la que este funcionando el microcontrolador (en caso del TP 16 MHz), por lo que resulta un parámetro muy importante a tener en cuenta al momento de diseñar un circuito.



## 9. Apéndice: Código del programa

### 9.1. Parte 1 - Timers

```
1 ;
2 ; TP3.asm
3 ;
4 ; Author : Violeta Perez Andrade 101456
5 ;
6
7 .equ LED_PORT_DIR = DDRB
8 .equ LED_PORT = PORTB
9 .equ LED_PIN_NUM = 1
10 .equ LED_PIN = PINB
11 .equ PUL1_PORT_DIR = DDRD
12 .equ PUL1_PIN = PIND
13 .equ PUL1_PIN_NUM = 2
14 .equ PUL2_PORT_DIR = DDRD
15 .equ PUL2_PIN = PIND
16 .equ PUL2_PIN_NUM = 3
17 .def pull_status = r16
18 .def pull_counter = r18
19 .def pull_actual = r20
20 .def pul2_status = r17
21 .def pul2_counter = r19
22 .def pul2_actual = r21
23
24
25 ; Replace with your application code
26 .include "m328Pdef.inc"
27 ;Inicio del codigo
28 .org 0x0000
29     rjmp inicio
30 .org INT0Addr
31     rjmp isr_int0
32 .org INT1Addr
33     rjmp isr_int1
34 .org OVF0addr
35     rjmp OVF0
36 .org OVF1addr
37     rjmp OVF1
38 .org OVF2addr
39     rjmp OVF2
40
41 .org INT_VECTORS_SIZE
42
43 inicio:
44 ; Se inicializa el Stack Pointer al final de la RAM utilizando la definicion global
45 ; RAMEND
46     ldi r16,HIGH(RAMEND)
47     out sph,r16
48     ldi r16,LOW(RAMEND)
49     out spl,r16
50
51
52     rcall configure_ports
53
54     //el registro pull_status lo voy a usar para salvar
55     //los estados de pulsador uno
56     in pull_status, PUL1_PIN
57     //este and es para dejar en uno el bit 2 que es el que me importa
58     andi pull_status, 0b00000100
59     //shifteo para que el valor me quede al final y sean mas facil las comparaciones
60     lsr pull_status
61     lsr pull_status
62
```

```

63 //lo mismo para el pulsador dos que ahora es el bit3
64 in pul2_status, PUL2_PIN
65 lsr pul2_status
66 lsr pul2_status
67 lsr pul2_status
68
69 //configuro timers e interrupciones
70 //rcall configure_timer0
71 rcall configure_timer1
72 //rcall configure_timer2
73 rcall configurar_interrupciones
74 rcall handle_led
75 sei
76
77 main_loop:
78 jmp main_loop
79
80 isr_int0:
81 in r24, SREG ; Guardar registro de estado
82 push r24
83 push r16
84
85 ;desactivo la interupcion externa para que no suceda
86 ;la interrupcion mientras estoy contando el debounce
87 cbi EIMSK, INTO
88
89 ;activo el timer 0 que lo voy a estar usando para contar
90 ;el debounce del pulsador uno
91 ldi r16, (1 << CS02) | (0 << CS01) | (1 << CS00) ;setea el prescaler en 1024
92 out TCCR0B, r16
93
94 ;activo la interrupcion del timer 0
95 ldi r16, 1<<TOIE0
96 sts TIMSK0, r16
97
98 pop r16
99 pop r24
100 reti
101
102 isr_int1:
103 in r24, SREG ; Guardar registro de estado
104 push r24
105 push r16
106
107 ;desactivo la interupcion externa para que no suceda
108 ;la interrupcion mientras estoy contando el debounce
109 cbi EIMSK, INT1
110
111 ;activo el timer2 que lo voy a estar usando para contar
112 ;el debounce del pulsador uno
113 ldi r16, (1 << CS22) | (1 << CS21) | (1 << CS20)
114 sts TCCR2B, r16
115
116 ;activo la interrupcion del timer 2
117 ldi r16, 1<<TOIE2
118 sts TIMSK2, r16
119
120 pop r16
121 pop r24
122 reti
123
124 OVF0:
125 in r24, SREG ; Guardar registro de estado
126 push r24
127
128 ;chequeo si el counter llego a lo necesario

```

```

129     cpi pull_counter, 3
130     breq OVFO_limit_reached
131
132     ;si no llego, simplemente lo incremento y salgo
133     inc pull_counter
134     rjmp reti_OVF0
135
136 OVFO_limit_reached:
137     ;en caso que ya haya pasado el tiempo de debounce
138     ;limpio el contador
139     clr pull_counter
140
141     ;paro el timer porque ya no quiero seguir contando
142     clr r0
143     sts TCCR0B, r0
144
145     ;reactivar interrupcion externa
146     sbi EIMSK, INT0
147
148     ;chequear si efectivamente cambio el bit
149     ;comparo el valor leido con el que tenia guardado en pull_status
150     ;si los valores son iguales, solamente fue un falso positivo
151     ;si no, manejo el cambio y actualizo el estado del pulsador
152
153     in pull_actual, PUL1_PIN
154     lsr pull_actual
155     lsr pull_actual
156     andi pull_actual, 0b00000001
157     cp pull_actual, pull_status
158     breq reti_OVF0
159     mov pull_status, pull_actual
160     rcall handle_led
161
162 reti_OVF0:
163     out SREG, r24 ; Restaurar registro de estado
164     pop r24
165     reti
166
167 handle_led:
168     ;en r25 cargo los estados de los pulsadores
169     ;y en base al caso que este steo el prescaler en lo necesario o enciendo el led
170     mov r25, pul2_status
171     lsl r25
172     or r25, pull_status
173     cpi r25, 0b00000001
174     breq handle_led_01
175     cpi r25, 0b00000010
176     breq handle_led_10
177     cpi r25, 0b00000011
178     breq handle_led_11
179 handle_led_00:
180     ;paro el timer uno
181     clr r0
182     sts TCCR1B, r0
183     ;prendo el led
184     sbi LED_PORT, LED_PIN_NUM
185     rjmp hanlde_led_fin
186 handle_led_01:
187     ;acitvo el timer 1 con prescaler 64 (011)
188     ldi r22, (0 << CS12) | (1 << CS11) | (1 << CS10) ;setea el prescaler en 64
189     sts TCCR1B, r22
190     rjmp hanlde_led_fin
191 handle_led_10:
192     ;acitvo el timer 1 con prescaler 256 (100)
193     ldi r22, (1 << CS12) | (0 << CS11) | (0 << CS10) ;setea el prescaler en 256
194     sts TCCR1B, r22

```

```

195     rjmp hanlde_led_fin
196 handle_led_l1:
197     ;acitvo el timer 1 con prescaler 1024 (101)
198     ldi r22, (1 << CS12) | (0 << CS11) | (1 << CS10) ;setea el prescaler en 1024
199     sts TCCR1B, r22
200     rjmp hanlde_led_fin
201 hanlde_led_fin:
202     ret
203
204 OVF1:
205     ;esta interrupcion prende el led cuando llegue a overflow el timer 1
206     ;y el tiempo hasta el overflow va a depender de en cuanto este seteado
207     ;el prescaler, que depende de como esten apretados los pulsadores
208     in r24, SREG ; Guardar registro de estado
209     push r24
210     sbic LED_PORT, LED_PIN_NUM ;skip if bit is cleared
211     rjmp OVF1_turn_off_led
212     sbi LED_PORT, LED_PIN_NUM
213     rjmp OVF1_fin
214 OVF1_turn_off_led:
215     cbi LED_PORT, LED_PIN_NUM
216 OVF1_fin:
217     pop r24
218     reti
219
220 OVF2:
221     ;la misma logica que OVF0 pero para el pulsador 2
222     in r24, SREG ; Guardar registro de estado
223     push r24
224     ;chequeo si el counter llego a lo necesario
225     cpi pul2_counter, 3
226     breq OVF2_limit_reached
227     inc pul2_counter
228     rjmp reti_OVF2
229
230 OVF2_limit_reached:
231     ;limpio el contador
232     clr pul2_counter
233     ;desactivar la interrupcion del timer
234     clr r0
235     sts TIMSK2, r0
236     ;reactivar interrupcion externa
237     sbi EIMSK, INT1
238     ;chequear si efectivamente cambio el bit
239     in pul2_actual, PUL2_PIN
240     lsr pul2_actual
241     lsr pul2_actual
242     lsr pul2_actual
243     andi pul2_actual, 0b00000001
244     cp pul2_actual, pul2_status
245     breq reti_OVF2
246     mov pul2_status, pul2_actual
247     rcall handle_led
248
249 reti_OVF2:
250     out SREG, r24 ; Restaurar registro de estado
251     pop r24
252     reti
253
254 configure_timer1:
255     push r16
256     ldi r16, (0 << CS12) | (0 << CS11) | (0 << CS10)
257     sts TCCR1B, r16
258     ldi r16, 1<<TOIE1
259     sts TIMSK1, r16
260     ldi r16, 1<<TOV1

```

```

261     sts TIFR1, r16
262     pop r16
263     ret
264
265
266
267 ;*****
268 ; Subrutina para configurar las interrupciones
269 ; registros utilizados: r16
270 ;*****
271 configurar_interrupciones:
272     push r16
273     ; INT0 e INT1 responden al cambio de flanco
274     ldi r16, (0 << ISC11) | (1 << ISC10) | (0 << ISC01) | (1 << ISC00)
275     sts EICRA, r16
276     ; Activar interrupciones para INT0 e INT1
277     ldi r16, (1 << INT1) | (1 << INT0)
278     out EIMSK, r16
279     pop r16
280     ret
281
282 ;*****
283 ; Se configuran los puertos del microcontrolador como entrada/salida/otra funcion
284 ;
285 ; En este caso, los puertos de los pulsadores se configuran como input
286 ; y el led como output
287 ;
288 ;Entrada:
289 ;Salida:
290 ;Registros utilizados: R20
291 ;*****
292
293 configure_ports:
294     push r20
295     push r21
296     ldi r21, 0x0
297     ldi r20, 0xFF
298     ;el led esta en el puerto B entonces es output
299     out DDRB, r20
300     ;como en los pulsadores estan en el puerto D
301     ;es de input
302     out DDRD, r21
303     ;pongo en 1 la resistencia de pulldown
304     sbi PORTD, 2
305     sbi PORTD, 3
306     pop r21
307     pop r20
308     ret

```

## 9.2. Parte 2 - PWM

```

1 ;
2 ; TP3p2.asm
3 ;
4 ; Author : Violeta Perez Andrade 101456
5 ;
6
7
8 ; Replace with your application code
9 .include "m328Pdef.inc"
10 .equ TOP_20ms = 40000
11 .equ UPPER_LIMIT = 4000
12 .equ LOWER_LIMIT = 2000
13 .def pul1_counter = r17
14 .def pul2_counter = r18

```

```

15 .def flag_pulsadores = r19
16 .def pull_flag = r20
17 .def pul2_flag = r21
18 .def contador_timer = r22
19 .def contador_servo = r23
20 ;Inicio del codigo
21 .org 0x0000
22     rjmp inicio
23 .org INT0Addr
24     rjmp isr_int0
25 .org INT1Addr
26     rjmp isr_int1
27 .org OVF0addr
28     rjmp OVF0
29 .org OVF2addr
30     rjmp OVF2
31
32 .org INT_VECTORS_SIZE
33
34 inicio:
35 ; Se inicializa el Stack Pointer al final de la RAM utilizando la definicion global
36 ; RAMEND
37     ldi     r16,HIGH(RAMEND)
38     out     sph,r16
39     ldi     r16,LOW(RAMEND)
40     out     spl,r16
41
42 main:
43     clr     contador_timer
44     clr     contador_servo
45     rcall   configure_ports
46     rcall   configure_timer0
47     rcall   configure_timer1
48     rcall   configure_timer2
49     rcall   configurar_interrupciones
50     sei
51 here:
52     rjmp    here
53 ;*****
54 ; Se configuran los puertos del microcontrolador como entrada/salida/otra funcion
55 ;
56 ; En este caso, los puertos de los pulsadores se configuran como input
57 ; y el puerto donde esta el servo como output
58 ;
59 ;Entrada:
60 ;Salida:
61 ;Registros utilizados: R20
62 ;*****
63
64 configure_ports:
65     push    r20
66     push    r21
67     ldi     r21, 0x0
68     ldi     r20, 0xFF
69     ;el servo esta en el puerto B entonces es output
70     out     DDRB, r20
71     ;como en los pulsadores estan en el puerto D
72     ;es de input
73     out     DDRD, r21
74     ;pongo en 1 la resistencia de pullup
75     sbi     PORTD, 2
76     sbi     PORTD, 3
77     pop     r21
78     pop     r20
79     ret
80

```

```

81 configure_timer1:
82     push r16
83     push r17
84     //TCCR1A
85     //dejo el a en modo inversor
86     //y el b en modo no inversor
87     clr r16
88     ori r16, (0 << WGM10) | (1 << WGM11) | (1 << COM1A1) | (0 << COM1A0) | (0 << COM1B1) | (0 << COM1B0)
89     sts TCCR1A, r16
90     //TCCRB
91     clr r16
92     ori r16, (0 << ICNC1) | (0 << ICES1) | (0 << CS12) | (1 << CS11) | (0 << CS10) | (1 << WGM12) | (0 << WGM13)
93     sts TCCR1B, r16
94     ldi r16, LOW(2000)
95     ldi r17, HIGH(2000)
96     sts OCR1AH, r17
97     sts OCR1AL, r16
98     ldi contador_servo, 0
99     //ICR1
100    //necesito que el top sea 40000=0x9C40
101    //para un pulso de 20 ms
102    ldi r16, HIGH(TOP_20ms)
103    sts ICR1H, r16
104    ldi r16, LOW(TOP_20ms)
105    sts ICR1L, r16
106    pop r17
107    pop r16
108    ret
109
110 configure_timer0:
111     push r16
112     ldi r16, (1 << CS02) | (0 << CS01) | (1 << CS00) ;setea el prescaler en 1024
113     out TCCR0B, r16
114     ;activar la interrupcion del timer
115     ldi r16, 1<<TOIE0
116     sts TIMSK0, r16
117     pop r16
118     ret
119
120 configure_timer2:
121     push r16
122     ldi r16, (1 << CS22) | (1 << CS21) | (1 << CS20)
123     sts TCCR2B, r16
124     ldi r16, 1<<TOIE2
125     sts TIMSK2, r16
126     pop r16
127     ret
128
129 OVF0:
130     in r24, SREG ; Guardar registro de estado
131     push r24
132     ;chequeo si el counter llego a lo necesario
133     cpi contador_timer, 13
134     breq OVF0_limit_reached
135     inc contador_timer
136     rjmp reti_OVF0
137 OVF0_limit_reached:
138     cpi flag_pulsadores, (1 << 0)
139     breq mover_izquierda
140     cpi flag_pulsadores, (1 << 1)
141     breq mover_derecha
142     rjmp reti_OVF0
143 mover_derecha:
144     cpi contador_servo, 16
145     breq fin_OVF0
146     //quiero sumar 125 que es el paso

```

```

147 //125=63+62
148 lds r24, OCR1AL
149 lds r25, OCR1AH
150 adiw r24, 63
151 adiw r24, 62
152 sts OCR1AH, r25
153 sts OCR1AL, r24
154 inc contador_servo
155 rjmp fin_OVF0
156 mover_izquierda:
157 cpi contador_servo, 0
158 breq fin_OVF0
159 //quiero sumar 125 que es el paso
160 //125=63+62
161 lds r24, OCR1AL
162 lds r25, OCR1AH
163 sbiw r24, 63
164 sbiw r24, 62
165 sts OCR1AH, r25
166 sts OCR1AL, r24
167 dec contador_servo
168 fin_OVF0:
169 ;limpio el contador
170 clr contador_timer
171 reti_OVF0:
172 pop r24
173 reti
174
175 isr_int0:
176 push r24
177 in r24, SREG ; Guardar registro de estado
178 ldi pull_flag, 1
179 clr pull_counter
180 out SREG, r24 ; Restaurar registro de estado
181 pop r24
182 reti
183
184 isr_int1:
185 push r24
186 in r24, SREG ; Guardar registro de estado
187 ldi pul2_flag, 1
188 clr pul2_counter
189 out SREG, r24 ; Restaurar registro de estado
190 pop r24
191 reti
192
193 OVF2:
194 push r24
195 in r24, SREG ; Guardar registro de estado
196 ;chequeo si el counter1 llego a lo necesario
197 cpi pull_flag, 1
198 breq handle_pull
199 cpi pul2_flag, 1
200 breq handle_pul2
201 rjmp reti_ovf2
202 handle_pull:
203 cpi pull_counter, 4
204 breq pull_debounce
205 inc pull_counter
206 rjmp reti_OVF2
207 pull_debounce:
208 sbic PIND, 2
209 breq pull_falso_positivo
210 ldi flag_pulsadores, (1 << 1)
211 clr pull_counter
212 rjmp reti_OVF2

```



```

213 pull_falso_positivo:
214     ldi pul1_flag, 0
215     ldi flag_pulsadores, 0
216     rjmp reti_OVF2
217
218 handle_pul2:
219     cpi pul2_counter, 4
220     breq pul2_debounce
221     inc pul2_counter
222     rjmp reti_OVF2
223 pul2_debounce:
224     sbic PIND, 3
225     breq pul2_falso_positivo
226     ldi flag_pulsadores, (1 << 0)
227     clr pul2_counter
228     rjmp reti_OVF2
229 pul2_falso_positivo:
230     ldi pul2_flag, 0
231     ldi flag_pulsadores, 0
232     rjmp reti_OVF2
233 reti_OVF2:
234     out SREG, r24
235     pop r24
236     reti
237
238 ;*****
239 ; Subrutina para configurar las interrupciones
240 ; registros utilizados: r16
241 ;*****
242 configurar_interrupciones:
243     push r16
244     ; INT0 e INT1 responden al flanco descendente
245     ldi r16, (1 << ISC11) | (0 << ISC10) | (1 << ISC01) | (0 << ISC00)
246     sts EICRA, r16
247     ; Activar interrupciones para INT0 e INT1
248     ldi r16, (1 << INT1) | (1 << INT0)
249     out EIMSK, r16
250     pop r16
251     ret

```