Fiuba
Facultad de Ingeniería
Universidad de Buenos Aires

Laboratorio de Microprocesadores - 86.07

Trabajo Práctico Nº6:

Timers

Profesor:			Ing. Guillermo Campiglio								
Cuatrimestre/Año:			1º/2020								
Turno de las clases prácticas			Miércoles								
Jefe de trabajos prácticos:			Ing. Pedro Ignacio Martos								
Docentes guía:			Ing. Fabricio Baglivo, Ing. Fernando Pucci								
Fecha de presentación:			29/07/2020								
Alumno			Seguimiento del proyecto								
Nombre	Apellido	Padrón									
Maximiliano Daniel	Reigada	100565									

Observaciones:

.....
.....
.....
.....
.....
.....
.....
.....
.....

Fecha de aprobación			Firma J.T.P

Índice

1. Objetivo

El presente trabajo práctico tiene como objetivo progresar con el manejo de registros de timers de los que dispone el microcontrolador ATmega328P, generar interrupciones a partir de eventos asociados a estos, e implementar métodos de antirrebote de teclas.

2. Descripción

Mediante la implementación de una placa de microcontrolador de código abierto basado en el microchip ATmega328P, denominada Arduino Uno, se busca desarrollar un programa para hacer parpadear un LED conectado al pin `PB0`, en 3 frecuencias distintas o que lo deje encendido de manera fija, según los valores que haya en los pines de entrada `PD0` y `PD1` a partir las indicaciones de la siguiente tabla:

`PD0`	`PD1`	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: Valores de referencia.

Para resolver esta práctica, se usa el `timer1` por medio de la interrupción por `overflow`. Cada vez que se produce un `overflow` se cambia el estado del LED mediante la rutina asociada a dicha interrupción, es decir, si está prendido se apaga y viceversa.

Cuando el LED se encuentra encendido de manera fija, el timer se encuentra apagado poniendo a cero los bits `CS12/1/0` del registro de configuración `TCCR1B`. En los otros tres casos, el timer cuenta los pulsos de clock divididos por prescaler 64 (`CS12/1/0=011`), 256 (`CS12/1/0=100`) y 1024 (`CS12/1/0=101`).

Por otra parte, en las entradas `PD0` y `PD1` se conectan 2 switches, que como producen rebotes al ser presionados, hacen necesario implementar un método antirrebote para detectar correctamente los valores de entrada.

3. Diagrama de conexiones en bloques

A continuación, puede verse el diagrama de conexiones en bloques correspondiente a este trabajo práctico:

Figura 1: Diagrama de conexiones en bloques.

4. Circuito esquemático

Se procede a realizar el esquemático del circuito a implementar. Cabe destacar que, si bien en la realidad los componentes se encuentran conectado al Arduino Uno y este a su vez a una fuente de tensión externa, resulta más útil representar en el esquema solo los componentes electrónicos esenciales del sistema, conectados a referencias de tensión de un determinado valor:

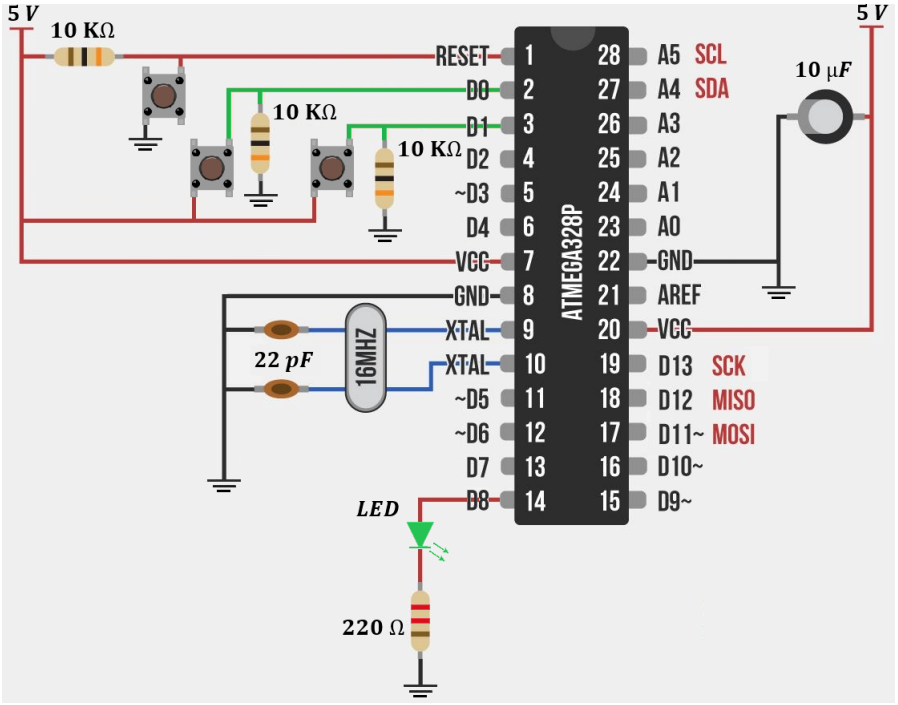

Figura 2: Circuito esquemático.

5. Listado de componentes

En el desarrollo de este trabajo práctico se utilizan los siguientes componentes:

- 1 LED verde de 5mm
- 1 Resistor de 220 Ω
- 2 Resistores de 10 kΩ
- 2 Pulsadores Dip Tact Switch 6x6x5mm
- 6 Cables de conexión
- 1 Protoboard
- 1 Arduino Uno (con microcontrolador ATmega328P incluido)

6. Diagrama de flujo del Software

A continuación, se explicitan los diagramas de flujo correspondientes al código desarrollado:

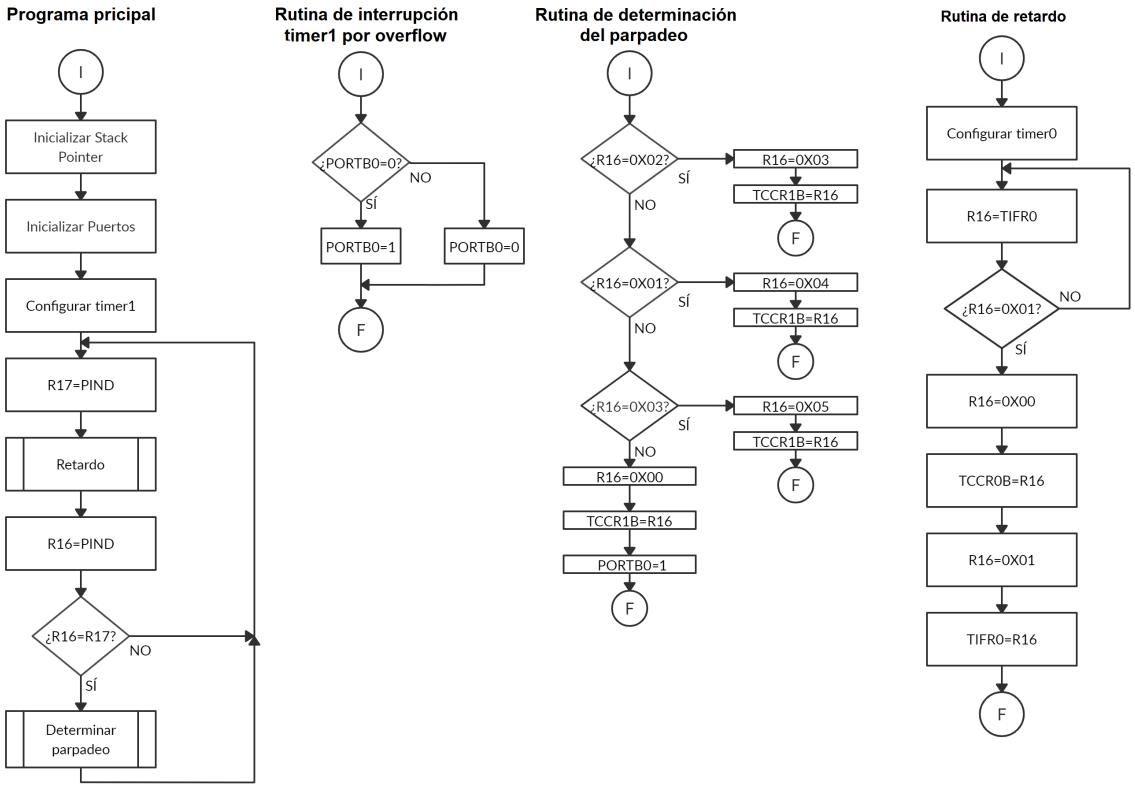

Figura 3: Diagramas de flujo.

7. Código de programa

A continuación, puede verse el código desarrollado para este proyecto:

```
;***************************************************************************
; Código correspondiente al ejercicio planteado en el Trabajo Práctico 6.
;
; Alumno: Reigada Maximiliano Daniel
; Padrón: 100565
;***************************************************************************

.INCLUDE "m328pdef.inc"

.DEF AUX=R16
.DEF REG_CMP=R17

.CSEG
.ORG 0X0000
      RJMP  config

.ORG OVF1addr
      RJMP  isr_timer1

.ORG INT_VECTORS_SIZE

config:
      LDI   AUX, HIGH(RAMEND)                 ;Inicializo el SP al final de la RAM.
      OUT   SPH, AUX
      LDI   AUX, LOW(RAMEND)
      OUT   SPL, AUX

      LDI   AUX, 0X00                         ;Declaro al puerto D como entrada.
      OUT   DDRD, AUX

      LDI   AUX, 0XFF                         ;Declaro al puerto B como salida.
      OUT   DDRB, AUX

      LDI   AUX, (1 << TOIE1)                 ;Habilito interrupción de timer1 por overflow.
      STS   TIMSK1, AUX

      SEI                                     ;Habilito interrupciones globales.

main:
      IN    REG_CMP, PIND
      ANDI  REG_CMP, (1 << PIND1 | 1 << PIND0) ;Guardo en REG_CMP el valor de PIND0 y PIND1.

      RCALL retardo

      IN    AUX, PIND                         ;Guardo en AUX el valor de PIND0 y PIND1 luego
      ANDI  AUX, (1 << PIND1 | 1 << PIND0)    ;de esperar un tiempo tras el último guardado.

      CP    REG_CMP, AUX                      ;Verifico que las dos muestras sean iguales.
      BRNE  main

      RCALL determinar_parpadeo
      RJMP  main


determinar_parpadeo:
      CPI   AUX, (1 << PIND1 | 0 << PIND0)
      BREQ  clock_64

      CPI   AUX, (0 << PIND1 | 1 << PIND0)
      BREQ  clock_256

      CPI   AUX, (1 << PIND1 | 1 << PIND0)
      BREQ  clock_1024

      LDI   AUX, 0X00                         ;Como PIND0=0 y PIND1=0, detengo el timer1
      STS   TCCR1B, AUX                       ;y dejo PB0 encendido.
      SBI   PORTB, 0
      RET
```

```
69
70  clock_64:
71        LDI   AUX, 0X03                       ;Como PIND0=0 y PIND1=1, configuro el timer1
72        STS   TCCR1B, AUX                     ;para que cuente los pulsos de clock divididos
73        RET                                   ;por prescaler 64.
74
75  clock_256:
76        LDI   AUX, 0X04                       ;Como PIND0=1 y PIND1=0, configuro el timer1
77        STS   TCCR1B, AUX                     ;para que cuente los pulsos de clock divididos
78        RET                                   ;por prescaler 256.
79
80  clock_1024:
81        LDI   AUX, 0X05                       ;Como PIND0=1 y PIND1=1, configuro el timer1
82        STS   TCCR1B, AUX                     ;para que cuente los pulsos de clock divididos
83        RET                                   ;por prescaler 1024.
84
85  isr_timer1:
86        SBIC  PORTB, 0                        ;Si PORTB0=0, salto la próxima instrucción.
87        RJMP  apagar_led
88
89        SBI   PORTB, 0                        ;Pongo PORTB0 en estado lógico alto.
90        RETI
91
92  apagar_led:
93        CBI   PORTB, 0                        ;Pongo PORTB0 en estado lógico bajo.
94        RETI
95
96
97  retardo:
98        LDI   AUX, 0X00                       ;Me aseguro de iniciar el registro TCNT0 en 0.
99        OUT   TCNT0, AUX
100
101       LDI   AUX, (1 << CS02 | 1 << CS00)    ;Configuro el timer0 en modo normal
102       OUT   TCCR0B, AUX                     ;y seteo al clock con prescaler 1024.
103
104 loop_retardo:
105       IN    AUX, TIFR0                      ;En caso de que se active el flag de overflow
106       SBRS  AUX, TOV0                       ;del timer0, esquivo la próxima instrucción.
107       RJMP  loop_retardo
108
109       LDI   AUX, 0X00
110       OUT   TCCR0B, AUX                     ;Desactivo el timer0.
111       LDI   AUX, (1<<TOV0)
112       OUT   TIFR0, AUX                      ;Limpio el flag de overflow del timer0.
113
114       RET
```

8. Resultados

Luego de desarrollar el código descripto en las secciones anteriores y armar el circuito especificado, se logra controlar el parpadeo o estado de encendido fijo del LED conectado al pin `PB0` a partir de los pulsadores dispuestos en los pines `PD0` y `PD1`. Cabe destacar, que para cargar el programa en el microcontrolador se debe desconectar momentáneamente los pines del puerto D del resto del circuito, debido a que durante este proceso dichos pines son utilizados como medio de comunicación entre los dos microcontroladores que posee el Arduino Uno.

Por otra parte, considerando que en la placa de Arduino Uno el oscilador de cristal es de $16MHz$, y que el Atmega328p toma como referencia de clock dicha frecuencia, se puede a calcular la frecuencia de oscilación en cada tipo de parpadeo a partir de:

$$f_n = \frac{16MHz}{2 \cdot 2^{16} \cdot n} \tag{1}$$

donde n es el valor del prescaler correspondiente a cada tipo de parpadeo, y 2^{16} es la cantidad maxima de valores que se pueden representar mediante los dos registros `TCNT1L` y `TCNT1H`.

Con esto último, se tiene que:

PD0	PD1	n	T_n $[s]$	f_n $[Hz]$
0	1	64	0,524	1,9
1	0	256	2,097	0,477
1	1	1024	8,389	0,120

Tabla 2: Frecuencias de parpadeo.

En cuanto al método antirrebote pedido en el enunciado, se decide por guardar en un registro el estado en los pines `PD0` y `PD1`, realizar una espera de aproximadamente $16ms$ mediante la implementación del `timer0`, y guardar en otro registro el estado actual de los pines. Tras esto se procede a comparar ambos valores, y en caso de que no sean compatibles se procede a reiniciar la lectura. En caso de que sean iguales, se selecciona el tipo de parpadeo a realizar.

Finalmente, se procede a realizar un listado de los componentes utilizados y sus costos:

Componentes	Costos
Led verde de 5mm	$7,00
Pulsadores Dip Tact Switch	$44,00
Resistor de 220 Ω	$5,50
Resistores de 10kΩ	$11,00
Cables de conexión	$12,00
Protoboard	$232,00
Arduino Uno	$720,00
TOTAL	$1031,50

9. Conclusiones

Tras haber realizado todos los pasos pedidos en el enunciado de este trabajo práctico, resta destacar las conclusiones que la experiencia ha aportado.

A partir de este trabajo se logra afianzar el manejo de timers del que puede disponer un microcontrolador, poniendo en práctica el cambio de estado lógico en un pin de salida mediante la interrupción asociada a un evento interno del dispositivo.

Es necesario hacer énfasis en todos los parámetros configurables del timer1 de los que se tuvo que tomar nota y especificar para esta práctica en particular a fin de hacerlo funcionar en modo normal. Así mismo, aquellas opciones de uso que no fueron implementadas, como por ejemplo las posibles configuraciones de los registros `TCCR1A`, `TCCR1C` e `ICR1`, quedan como buenas herramientas a tener en cuenta en futuras aplicaciones.

Por otra parte, el uso del polling como método para evitar efectos no deseados asociados a rebotes en pulsadores, se toma como un ejemplo de caso de aplicación en el que implementar esta metodología en vez del uso de interrupciones externas, simplifica la resolución del problema planteado.