Laboratorio de Microprocesadores - 86.07

Trabajo Práctico Obligatorio N°1

Parpadeo de un LED

Profesor:			Ing. Guillermo Campiglio									
Cuatrimestre/Año:			1°/2020									
Turno de las clases prácticas			Miércoles									
Jefe de trabajos prácticos:			Ing. Pedro Ignacio Martos									
Docente guía:			Ing. Fabricio Baglivo, Ing. Fernando Pucci									
Autores			Seguimiento del proyecto									
Nombre	Apellido	Padrón										
Santiago	López	100566										

Observaciones:

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

Fecha de aprobación				Firma J.T.P

Coloquio	
Nota final	
Firma profesor	

Índice

1. Objetivo

Proveer de las directivas adecuadas a un microcontrolador, con el fin de que un LED se encienda y se apague de forma intermitente.

2. Descripción

En el presente trabajo práctico se comenzó a familiarizarse algunas instrucciones del microcontrolador ATmega328P, utilizando el software *AVR Studio*, con el cual se efectuó un programa que controla el estado de un pin del microcontrolador. Una vez que el programa se *assembló* y *debuggeó*, se lo probó de forma experimental, cargando el programa al microcontrolador con la herramienta *avrdude*.

Para interactuar con el LED, se utilizaron los puertos B del microcontrolador. En un caso se seteó el puerto entero en modo de salida de datos, y luego se interactuó directamente con el pin deseado; en otro caso simplemente se seteó en modo de salida al biestable asociado al pin en cuestión.

3. Diagrama de conexiones en bloque

Las conexiones siguieron el esquema de la Figura 1.

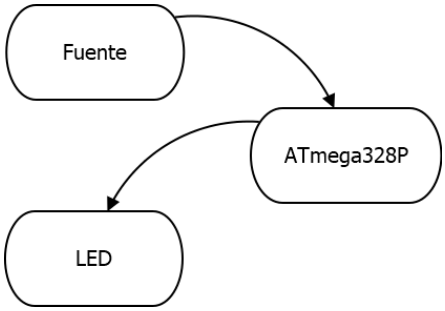

Figura 1: Diagrama de conexiones en bloque.

4. Esquemático

La Figura 2 muestra las conexiones eléctricas efectuadas en el práctico. Los componentes utilizados se encuentran en la sección siguiente.

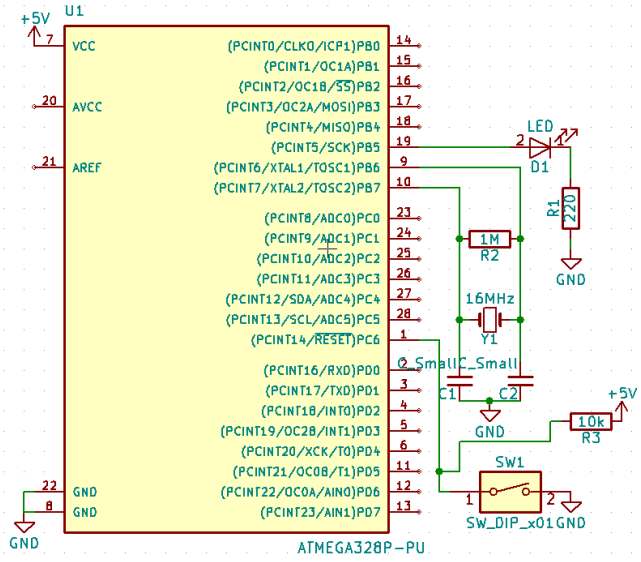

Figura 2: Circuito esquemático.

5. Listado de componentes

Los componentes utilizados fueron los listados a continuación:

- Placa de desarrollo Arduino UNO
- Resistor de 220Ω
- LED de color rojo
- Protoboard
- Cables unipolares

6. Diagrama de flujo

En la Figura 3 se presentan los pasos a seguir por el programa. Ha de notarse que no hay un fin en el diagrama, dado que el programa no termina nunca.

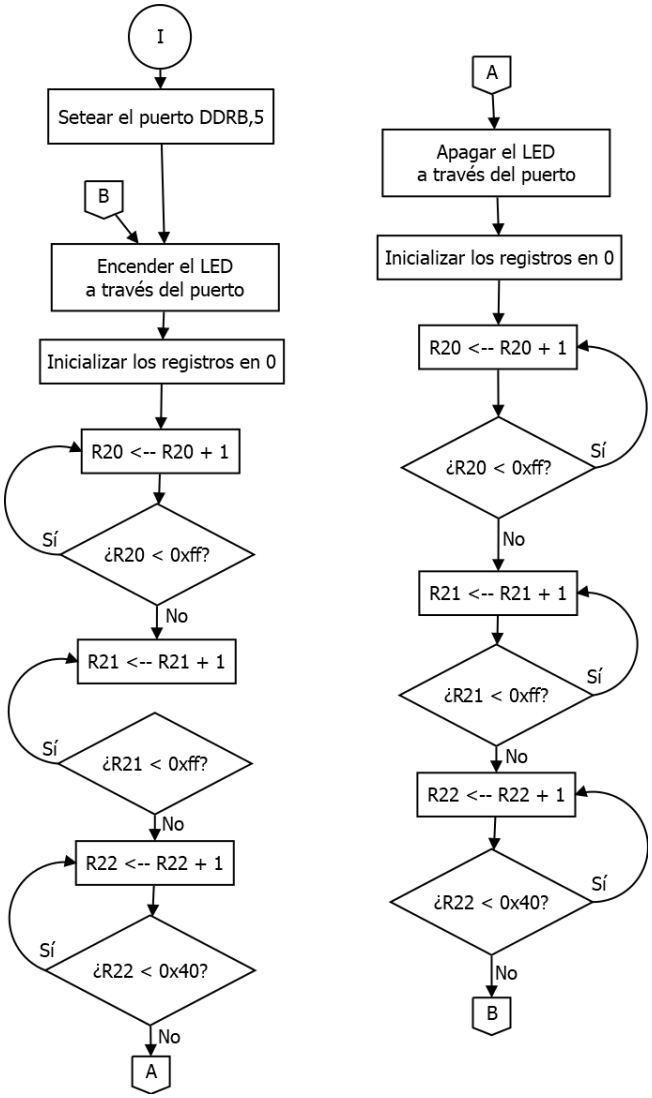

Figura 3: Diagrama de flujo.

7. Código fuente

En el primer programa se controla el estado del LED mediante el manejo del biestable correspondiente, sin afectar el estado del resto, mientras que en el segundo programa se controla el estado del LED habiendo seteado al puerto B completo en formato de salida de datos.

```
 1: .include "m328pdef.inc"
 2:
 3: .dseg
 4:
 5: .EQU LED_PORT = 5               ; led en PORTB 5
 6:                                 ; se que el led esta ahi por la hoja de datos
 7:
 8: .cseg                           ; todo lo que viene a cont es codigo ejecutable
 9:                                 ; (va en la flash program memory)
10:
11: .org 0x0000                     ; escribo a continuacion de 0x0000
12:             jmp     main       ; el programa se va a donde arranca mi codigo
13:
14: .org INT_VECTORS_SIZE
15: main:
16:             sbi     DDRB,LED_PORT   ; pongo en modo salida el FF que se comunica
17:                                     ; con el puerto del led
18:
19: led_on:     sbi     PORTB,LED_PORT  ; enciendo el led
20:
21: // inicializo los contadores        ; cuento los ciclos de maquina que quiero
22:             ldi     r20,0x00        ;  4 * 255
23:             ldi     r21,0x00        ; (4 * 255 + 5) * 255
24:             ldi     r22,0x00        ; (4 * 255 + 5) * 255 + 5) * 64 * 1/f = 1s
25:
26: ciclo_encendido:                    ; chequeo los ciclos de maquina paso a paso
27:             inc     r20             ;((( 1 +
28:             cpi     r20,0xff        ;    1 +
29:             brlo    ciclo_encendido ;    2  )  * 255 +
30:
31:             ldi     r20,0x00        ;  ( 1 +
32:             inc     r21             ;    1 +
33:             cpi     r21,0xff        ;    1 +
34:             brlo    ciclo_encendido ;    2  )) * 255 +
35:
36:             ldi     r21,0x00        ;  ( 1 +
37:             inc     r22             ;    1 +
38:             cpi     r22,0x40        ;    1 +
39:             brlo    ciclo_encendido ;    2  )) * 64 ; 64 pues voy hasta 0x40
40:
41:
42:             cbi     PORTB,5         ; apago el led
43:
44: // inicializo los contadores        ; cuento los ciclos de maquina que quiero
45:             ldi     r20,0x00        ;  4 * 255
46:             ldi     r21,0x00        ; (4 * 255 + 5) * 255
47:             ldi     r22,0x00        ; (4 * 255 + 5) * 255 + 5) * 64 * 1/f = 1s
48:
49: ciclo_apagado:                      ; cheque los ciclos de maquina paso a paso
50:             inc     r20             ;((( 1 +
51:             cpi     r20,0xff        ;    1 +
52:             brlo    ciclo_apagado   ;    2  )  * 255 +
53:
54:             ldi     r20,0x00        ;  ( 1 +
55:             inc     r21             ;    1 +
56:             cpi     r21,0xff        ;    1 +
57:             brlo    ciclo_apagado   ;    2  )) * 255 +
58:
59:             ldi     r21,0x00        ;  ( 1 +
60:             inc     r22             ;    1 +
61:             cpi     r22,0x40        ;    1 +
62:             brlo    ciclo_apagado   ;    2  )) * 64
63:
64:
65:             RJMP    led_on          ; vuelvo a arrancar
```

```
    1: .include "m328pdef.inc"
    2:
    3:
    4: .cseg                          ; todo lo que viene a cont es codigo ejecutable (va
en la flash program memory)
    5: .org 0x0000                    ; todo lo que viene a cont ponelo a partir de 0x0000
    6:              jmp     main      ; main es una etiqueta
    7:                                ; al hacer eso estoy pisando el reset con el jump
    8:
    9: .org INT_VECTORS_SIZE          ; los perifericos tienen asociada una dir de memoria
 a partir de la cual se ejecuta codigo especifico
   10: main:                          ; INT_VECTOR_SIZE calcula la cantidad de memoria que
 hay que dejar para esos perifericos
   11:                                ; la etiqueta esta definida en el include
   12:              ldi     r23,0xff
   13:              out     DDRB,r23  ; pongo el puerto en modo salida
   14:
   15: led_on:      ldi     r23,0xff
   16:              out     PORTB,r23 ; enciendo el led
   17:
   18:
   19: // inicializo los contadores
   20:              ldi     r20,0x00  ;  4 * 255
   21:              ldi     r21,0x00  ; (4 * 255 + 5) * 255
   22:              ldi     r22,0x00  ; (4 * 255 + 5) * 255 + 5) * 64 * 1/f = 1,5s
   23:
   24: ciclo_encendido:
   25:              inc     r20             ;((( 1 +
   26:              cpi     r20,0xff        ;    1 +
   27:              brlo    ciclo_encendido ;    2  )  * 255 +
   28:
   29:              ldi     r20,0x00        ;  ( 1 +
   30:              inc     r21             ;    1 +
   31:              cpi     r21,0xff        ;    1 +
   32:              brlo    ciclo_encendido ;    2  )) * 255 +
   33:
   34:              ldi     r21,0x00        ;  ( 1 +
   35:              inc     r22             ;    1 +
   36:              cpi     r22,0x40        ;    1 +
   37:              brlo    ciclo_encendido ;    2  )) * 64
   38:
   39:              clr     r23
   40:              out     PORTB,r23       ; apago el led
   41:
   42: // inicializo los contadores
   43:              ldi     r20,0x00  ;  4 * 255
   44:              ldi     r21,0x00  ; (4 * 255 + 5) * 255
   45:              ldi     r22,0x00  ; (4 * 255 + 5) * 255 + 5) * 64 * 1/f = 1,5s
   46:
   47: ciclo_apagado:
   48:              inc     r20             ;( 1 +
   49:              cpi     r20,0xff        ;  1 +
   50:              brlo    ciclo_apagado   ;  2  ) * 255
   51:
   52:              ldi     r20,0x00        ;( 1 +
   53:              inc     r21             ;  1 +
   54:              cpi     r21,0xff        ;  1 +
   55:              brlo    ciclo_apagado   ;  2  )* 255
   56:
   57:              ldi     r21,0x00        ;( 1 +
   58:              inc     r22             ;  1 +
   59:              cpi     r22,0x40        ;  1 +
   60:              brlo    ciclo_apagado   ;  2  )* 64
   61:
   62:
   63:              RJMP    led_on      ; vuelvo a arrancar
```

8. Costos

A continuación se presenta un listado de los costos de los componentes utilizados en el práctico.

- Arduino UNO - $850
- Resistores - $20
- LED rojo - $20
- Protoboard - $240
- Cables unipolares - $150

Sumando un costo total de $1280.

9. Conclusiones

El manejo de los puertos es importante, por eso se programó y testeó la respuesta del microcontrolador ante el manejo de un solo bit, como del byte entero. En este caso no pareció haber diferencia alguna, pero para casos futuros esto parece de gran importancia, si se llegara querer conocer el estado de los puertos en una zona más avanzada del código, con el fin de operar con la tranquilidad de que no va a suceder ningún imprevisto.

Como última observación, cabe aclarar que el uso de los registros como contadores no pareció la forma más óptima de llevar a cabo los *delays*, pero demostraron la importancia de prestarle atención a los ciclos de máquina que toma cada operación realizada para que no hubiera imprevistos con la frecuencia de oscilación.