Fiuba
Facultad de Ingeniería
Universidad de Buenos Aires

Laboratorio de Microprocesadores - 86.07

Trabajo Práctico Nº1:

Parpadeo de un LED

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:			20/05/2020								
Alumno			Seguimiento del proyecto								
Nombre	Apellido	Padrón									
Maximiliano Daniel	Reigada	100565									

Observaciones:

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

Fecha de aprobación			Firma J.T.P

Índice

1. Objetivo **1**

2. Descripción **1**

3. Diagrama de conexiones en bloques **1**

4. Circuito esquemático **2**

5. Listado de componentes **2**

6. Diagrama de flujo del Software **3**

7. Código de programa **3**
7.1. Uso de solo un pin 3
7.2. Uso de todo el puerto 4

8. Resultados **5**

9. Conclusiones **5**

1. Objetivo

El presente trabajo práctico tiene como objetivo poner en práctica el manejo de un puerto de entrada/salida en un microcontrolador para hacer parpadear un LED.

2. Descripción

Mediante la implementación de una placa de microcontrolador de código abierto basado en el microchip ATmega328P, denominada Arduino Uno, se busca controlar el parpadeo de un LED.

Se procede a conectar el LED al pin 0 del puerto B del microcontrolador, colocando un resistor en serie para limitar la corriente de este, y cumplir con el objetivo planteado por medio de un programa cargado en el dispositivo. Este programa escrito originalmente en assembly, se desarrolla y compila mediante el entorno de desarrollo Atmel Studio 7.0 y se graba sobre el microchip mediante el programa AVRDUDE.

Cabe destacar que en este caso se aplican dos métodos diferentes de programación, en el primero solo se alterna el estado lógico del pin al que se encuentra conectado el LED, mientras que, en el segundo, se altera el estado lógico de todo el puerto. Para ambos casos el principio de funcionamiento del ciclo iterativo es el mismo: se prende el LED poniendo al pin o puerto en un estado lógico alto, se realiza un retardo temporal, se apaga el LED poniendo ahora al pin o puerto en un estado lógico bajo, se vuelve a realizar un retardo y se repite todo el proceso.

Por otro lado, a fin de aprender a interpretar y utilizar la relación entre las interrupciones y el tiempo que conlleva la ejecución de estas, se decide por implementar los retardos por medio de 3 subrutinas particulares. La primera subrutina genera un retardo de $10\mu s$, la segunda uno de $10ms$ y la tercera de $1000ms$, y es necesario aclarar que todas son calculadas para el cristal de $16MHz$ del que dispone el Arduino.

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 el LED se encuentra conectado al Arduino Uno y este a su vez a la computadora, resulta más útil representar en el esquema solo los componentes electrónicos esenciales del sistema, conectados a referencias de tensión de un determinado valor:

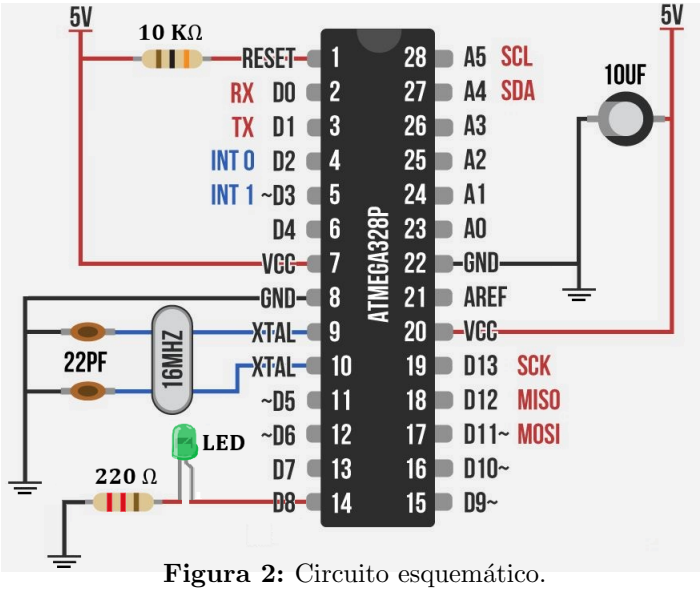

Figura 2: Circuito esquemático.

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 Cables de conexión
- 1 Protoboard
- 1 Arduino Uno (con microcontrolador ATmega328P incluido)

6. Diagrama de flujo del Software

Se desarrolla el diagrama de flujo correspondiente al programa principal diseñado y a los retardos utilizados en este:

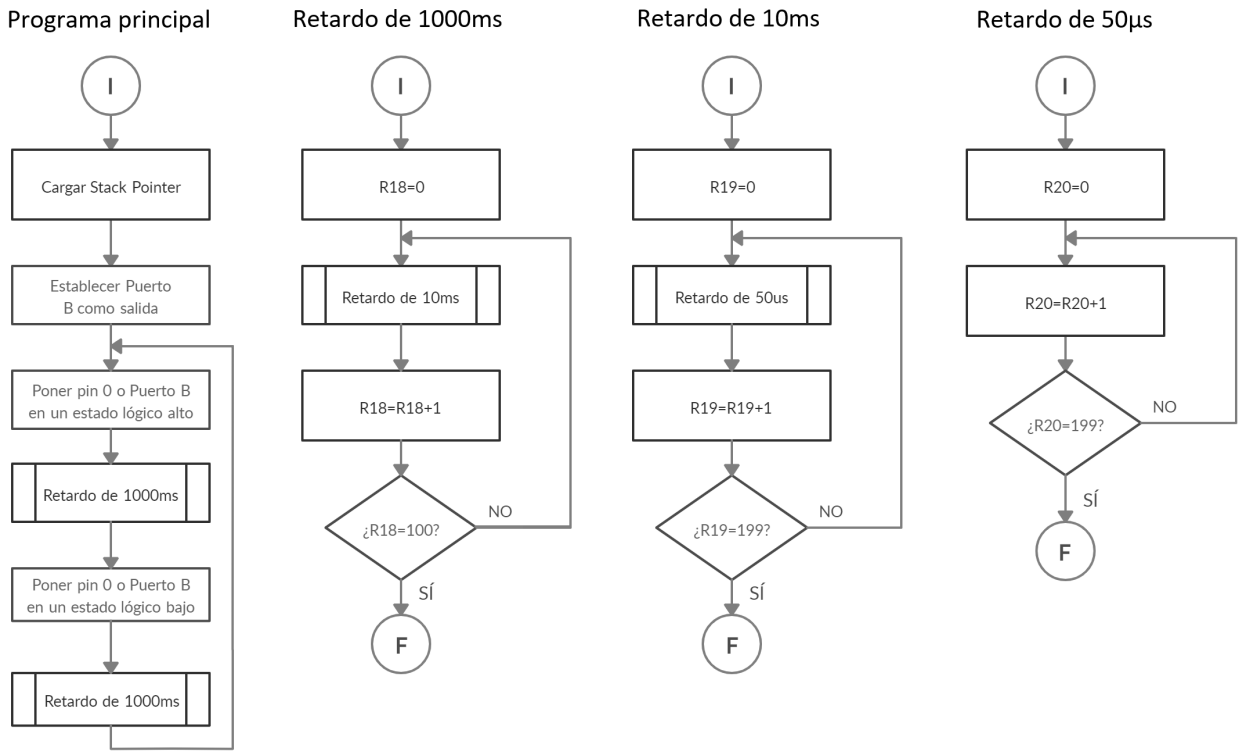

Figura 3: Diagrama de flujo.

7. Código de programa

A continuación, puede verse el código desarrollado para este proyecto, implementando los dos métodos descritos:

7.1. Uso de solo un pin

```
.include "m328pdef.inc"

.cseg
.org 0x0000
          rjmp main

.org INT_VECTORS_SIZE

main:
                                ;Se inicializa el Stack Pointer al final de la RAM
    ldi R16, HIGH(RAMEND)       ;Carga el SPH
    out SPH, R16
    ldi R16, LOW(RAMEND)        ;Carga el SPL
    out SPL, R16

                                ;Se configura el puerto B
   ldi R16, 0xFF
   out DDRB, R16                ;Configura al puerto B como salida
```

```
19
20                          ;Comienza el ciclo de encendido y apagado
21  main_loop:
22     sbi PORTB, 0         ;Pone el pin 0 del puerto B en un estado lógico alto
23     rcall retardo_1000ms
24     cbi PORTB, 0         ;Pone el pin 0 del puerto B en un estado lógico bajo
25     rcall retardo_1000ms
26     rjmp main_loop       ;Reinicio del ciclo
27
28
29
30
31  ;***************************************************************************************************
32  ;                Retardo de 1000ms calculado con un cristal de 16MHz
33  ;El loop interno se debe ejecutar un numero X de veces para que el tiempo total sea 1000ms.
34  ;Considerando los ciclos de máquina que conlleva cada instrucción ejecutada:
35  ;1000ms=(1/16MHz)*(1+X*160004-1+4) ----> X=100
36  ;***************************************************************************************************
37  retardo_1000ms:
38      eor     R18, R18                ;1 CM
39  loop_retardo_1000ms:
40      rcall   retardo_10ms            ;160000 CM
41      inc     R18                     ;1 CM
42      cpi     R18,100                 ;1 CM
43      brne    loop_retardo_1000ms     ;2 CM
44                                      ;1 CM (brne conlleva 1CM cuando R18=100)
45      ret                             ;4 CM
46
47
48
49  ;***************************************************************************************************
50  ;                Retardo de 10ms calculado con un cristal de 16MHz
51  ;El loop interno se debe ejecutar un numero X de veces para que el tiempo total sea 10ms.
52  ;Considerando los ciclos de máquina que conlleva cada instrucción ejecutada:
53  ;10ms=(1/16MHz)*(1+X*804-1+4) ----> X=199
54  ;***************************************************************************************************
55  retardo_10ms:
56      eor     R19, R19                ;1 CM
57  loop_retardo_10ms:
58      rcall   retardo_50us            ;800 CM
59      inc     R19                     ;1 CM
60      cpi     R19,199                 ;1 CM
61      brne    loop_retardo_10ms       ;2 CM
62                                      ;1 CM (brne conlleva 1CM cuando R19=199)
63      ret                             ;4 CM
64
65
66
67  ;***************************************************************************************************
68  ;                Retardo de 50us calculado con un cristal de 16MHz
69  ;El loop interno se debe ejecutar un numero X de veces para que el tiempo total sea 50us.
70  ;Considerando los ciclos de máquina que conlleva cada instrucción ejecutada:
71  ;50us=(1/16MHz)*(1+X*4-1+4) ----> X=199
72  ;***************************************************************************************************
73  retardo_50us:
74      eor     R20, R20                ;1 CM
75  loop_retardo_50us:
76      inc     R20                     ;1 CM
77      cpi     R20,199                 ;1 CM
78      brne    loop_retardo_50us       ;2 CM
79                                      ;1 CM (brne conlleva 1CM cuando R20=199)
80      ret                             ;4 CM
```

7.2. Uso de todo el puerto

```
1   .include "m328pdef.inc"
2
3   .cseg
4   .org 0x0000
5            rjmp main
6
7   .org INT_VECTORS_SIZE
8
9   main:
10                              ;Se inicializa el Stack Pointer al final de la RAM
11     ldi R16, HIGH(RAMEND)    ;Carga el SPH
12     out SPH, R16
13     ldi R16, LOW(RAMEND)     ;Carga el SPL
14     out SPL, R16
15                              ;Se configura el puerto B
16     ldi R16, 0xFF
17     ldi R17, 0X00
18     out DDRB, R16            ;Configura al puerto B como salida
19
20
21                              ;Comienza el ciclo de encendido y apagado
22  main_loop:
23     out PORTB, R16           ;Pone al puerto B en un estado lógico alto
24     rcall retardo_1000ms
25     out PORTB, R17           ;Pone al puerto B en un estado lógico bajo
26     rcall retardo_1000ms
27     rjmp main_loop           ;Reinicio del ciclo
```

8. Resultados

Luego de desarrollar el código descripto en las secciones anteriores y armar el circuito dispuesto, se logra controlar el encendido del led a través de un pin del microcontrolador.

A continuación, puede verse un listado de los componentes utilizados y sus costos:

Componentes	Costos
Led verde de 5mm	$7,00
Resistor de 220 Ω	$5,50
Cables de conexión	$4,00
Protoboard	$232,00
Arduino Uno	$720,00
TOTAL	$968,50

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. En primera instancia la idea de poner en práctica el manejo de registros, instrucciones y puertos de un microcontrolador con una tarea simple como el encendido y apagado de un LED, funciona notablemente como una introducción a la resolución de sistemas controlados mediante microprocesadores. A su vez, sirve como una clara iniciación en la programación en un lenguaje de bajo nivel como lo es el Assembler, estableciendo relaciones directas o cuasi-directas entre el software y el hardware utilizado.

Por último, como apreciación subjetiva, es necesario destacar lo relativamente problemático que es generar un retardo temporal. Es comprensible que esto sea así, debido a las limitaciones que se tienen por el momento en cuanto al conocimiento del manejo del microcontrolador. Sin embargo, a futuro sería de gran utilidad encontrar una manera más compacta y accesible de implementar este tipo de rutinas.