



Laboratorio de Microprocesadores - 86.07

Trabajo Práctico Obligatorio N°1

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	
Mariano Federico	Guglieri 99573
Seguimiento del proyecto	

Observaciones:

.....

.....

.....

.....

.....

.....

.....

.....

Fecha de aprobación		Firma J.T.P

Coloquio	
Nota final	
Firma profesor	

Índice

1. Objetivos	2
2. Desarrollo	2
3. Diagrama en bloques	2
4. Esquemático y listado de componentes	2
5. Diagrama de flujos	4
6. Códigos	5
7. Resultados	6
8. Conclusiones	6

1. Objetivos

El objetivo de este trabajo consiste en implementar un programa el cual, mediante una interrupción externa, modifique su comportamiento. La tarea principal del programa deberá mantener encendido un led. Cuando se presione un botón, dicho led se apagará, se encenderá otro parpadeando cinco veces y luego se volverá a encender el led principal. El led secundario permanecerá apagado luego de haber parpadeado.

2. Desarrollo

Para la realización del trabajo se utilizó la plataforma de desarrollo de Atmel, Atmel Studio versión 7.0, en donde se implementó el Software. Para la parte física se requirió de un Arduino UNO, el cual sirvió como programador para el integrado ATMEGA328P, además de suplirlo con la energía necesaria para su funcionamiento. Se utilizaron dos resistores junto a dos leds de color verde conectados al puerto b, un pulsador conectado al pin de interrupción, un resistor de $1k\Omega$ y tres cables para unir el circuito.

El código del programa consta de una parte principal, la cual se encarga de que el LED principal (LED0) esté prendido en todo momento, y una subrutina de interrupción. En esta subrutina se encuentran las instrucciones de la interrupción.

La primera parte del código está dedicada a la configuración básica de los puertos y la inicialización del Stack Pointer. Se configura el puerto B como salida y el puerto D como entrada; además de configurar la interrupción para que se accione por flanco ascendente. Cabe aclarar, que en la configuración no se activo la resistencia de pull-up del puerto D. Sin embargo, se explicará dentro del código cómo se podría hacer esto, además de explicar como estaría configurado el esquemático del circuito de haber utilizado pull-up.

Se utilizó una subrutina de delay para poder apreciar visualmente el parpadeo.

3. Diagrama en bloques

El diagrama de conexiones en bloque del trabajo se presenta a continuación:

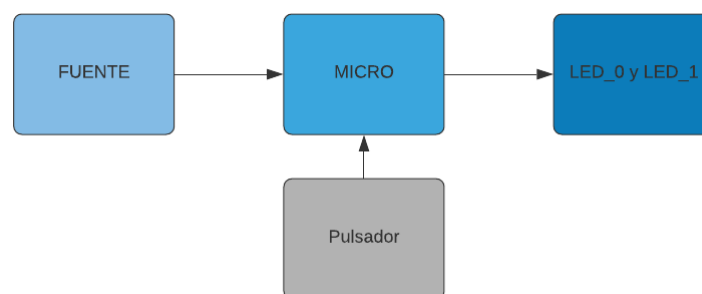


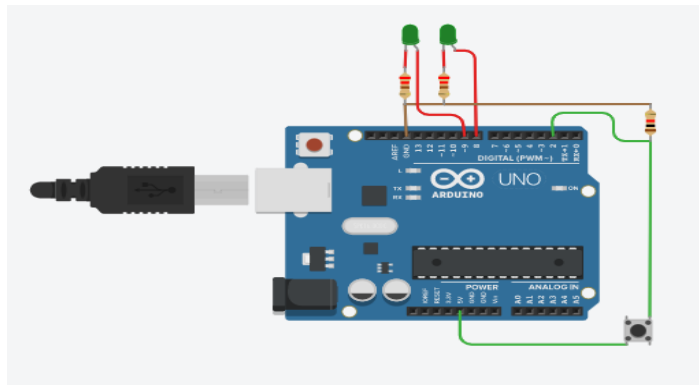
Figura 1: Diagrama de conexiones en bloque

Se observa que el funcionamiento surge primero de la pc donde se encuentra el código que será descargado en el micro y a su vez sirve como una fuente. El micro recibe información del estado del botón. En base a si se pulso o no, el micro ejecutará la rutina de interrupción encendiendo el LED1 cinco veces y apagando el LED0. Esto es lo que se ve en el diagrama.

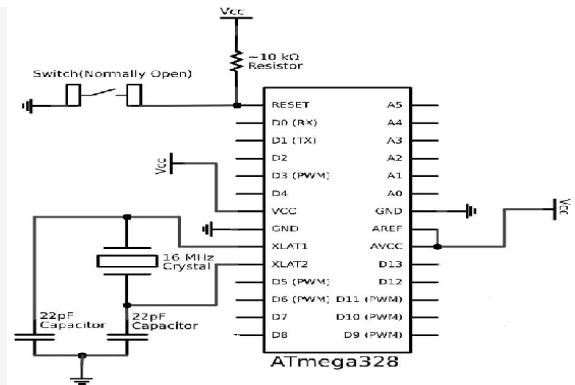
4. Esquemático y listado de componentes

Para el circuito se utilizaron los siguientes componentes:

- 2 resistores de 220Ω
- 1 resistor de $1k\Omega$
- 2 leds de color verde
- 1 pulsador
- 3 cables para protoboard
- 1 placa arduino con atmega328p
- 1 protoboard



(a) Esquemático del circuito implementado



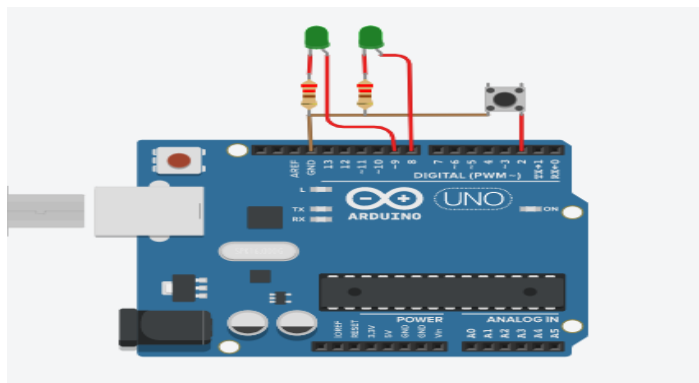
(b) Circuito con cristal de 16MHz

Figura 2: Esquemático del circuito

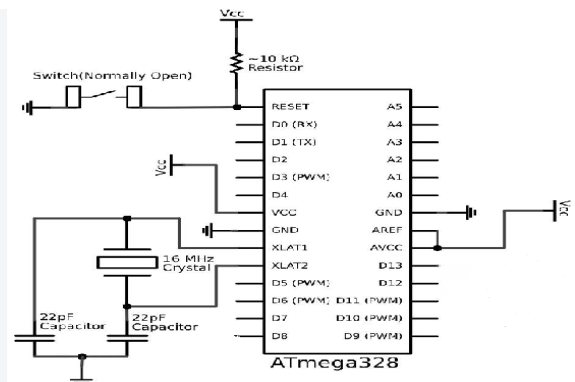
Para esta configuración, la resistencia de pull-up no se utilizó:

Si se hubiera utilizado la resistencia de pull-up, se utilizaría un componente menos, y el circuito sería más sencillo. A continuación, como hubiese quedado con pull-up:

- 2 resistores de 220Ω
- 2 leds de color verde
- 1 pulsador
- 3 cables para protoboard
- 1 placa arduino con atmega328p
- 1 protoboard



(a) Esquemático del circuito implementado



(b) Circuito con cristal de 16MHz

Figura 3: Esquemático del circuito

En total se habrá gastado unos 50 pesos para el circuito. Sin contar la plataforma arduino.

La plataforma arduino le agrega al microcontrolador un cristal de 16Mhz y una resistencia de $10k\Omega$, conectados de la siguiente manera.

5. Diagrama de flujos

A continuación se presentan los diagramas de flujos. El primero consiste en el programa principal más la configuración previa. La configuración de puertos y de la interrupción se puso todo en un mismo bloque ya que esto ya se vió en los trabajos anteriores. La rutina de interrupción se encuentra al lado.

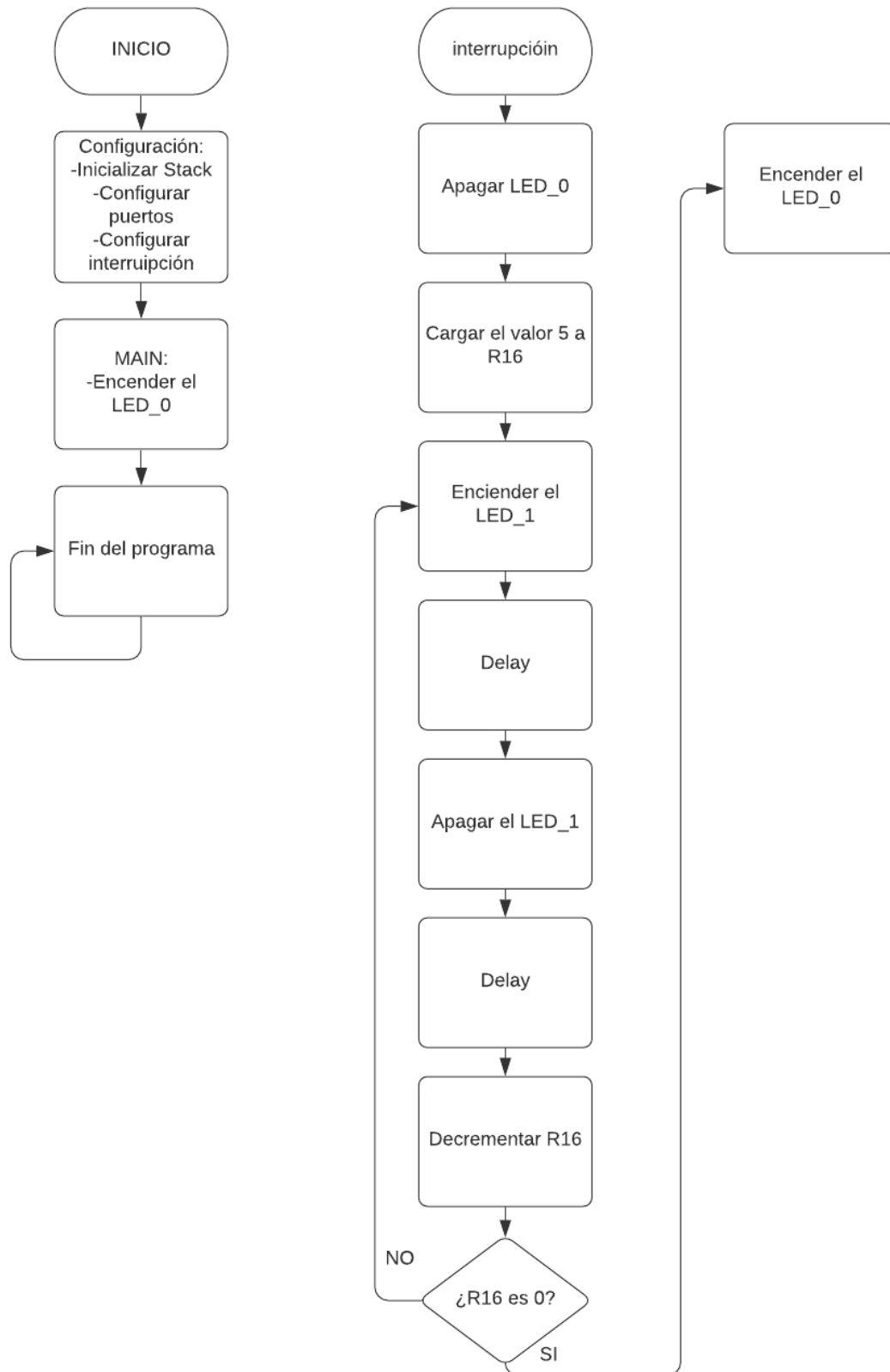


Figura 4: Diagrama de flujos

6. Códigos

A continuación se presenta el código:

```

1  .include "m328pdef.inc"
2
3  .cseg
4  ;direcciones de codigo
5  .org 0x0000
6  jmp     configuracion
7  .org INT0addr
8  jmp     interrupcion_0
9  .org INT_VECTORS_SIZE
10
11 configuracion:
12     ;inicializo el stack
13     ldi     R16, HIGH(RAMEND)
14     out     SPH, R16
15     ldi     R16, LOW(RAMEND)
16     out     SPL, R16
17
18     ;declaro el PORTB como salida
19     ldi     R16, 0xff
20     out     DDRB, R16
21
22     ;declaro el PORTD como entrada activando resistencia de pull-up
23     ldi     R16, 0x00
24     out     DDRD, R16
25     ldi     R16, 0xff
26     out     PORTD, R16
27
28     ;configuro la interrupcion 0 por flanco ascendente
29     ldi     R16, ( 1 << ISC01 | 0 << ISC00 )
30     sts     EICRA, R16
31     ldi     R16, (1 << INT0)
32     out     EIMSK, R16
33
34     sei
35
36     ;programa principal consta de dejar prendido un led
37 main:
38     sbi     PORTB, 0
39 fin:   rjmp  fin
40
41     ;codigo de la interrupcion
42 interrupcion_0:
43
44         cbi     PORTB, 0
45
46         ldi     R16, 5
47
48     loop_int0: sbi     PORTB, 1
49                 rcall delay
50                 cbi     PORTB, 1
51                 rcall delay
52
53                 dec     R16
54                 brne    loop_int0
55                 sbi     PORTB, 0
56 reti
57     ;delay para que sea visible las pulsaciones del LED
58 delay:
59     ldi r23, 32
60     loop3: ldi r24, 255
61     loop2: ldi r22, 255
62     loop1: dec r22
63     brne loop1
64     dec r24
65     brne loop2
66     dec r23
67     brne loop3
68 ret

```

7. Resultados

Si bien se logró el cometido de apagar el LED0 e implementar un parpadeo en el LED1 cuando se acciona el pulsador, a veces el LED1 parpadeaba diez veces en vez de cinco. Fuera de esto no se encontró otro inconveniente en el trabajo.

8. Conclusiones

Nuevamente se observa que al aplicar la resistencia de pull-up se ahorra un componente en el circuito.

Se observa que al utilizar una interrupción se evita usar la lógica de polling. Esto permite hacer un código más legible además de ahorrar ciclos de máquina. También, al ser el polling una lógica que pregunta constantemente si ocurrió una modificación externa, se debe tener en cuenta la frecuencia de muestreo. Es decir, cada cuánto el programa chequea si hubo una interrupción. Todo esto se evita con las interrupciones. Además, permiten una manera más cómoda de decidir el modo de accionamiento de la interrupción, sea por flanco ascendente, descendente, por estado alto o por estado bajo.

Con respecto al error que se observó en los resultados, se llega a la conclusión que el error se encuentra en el accionar mecánico del pulsador. Éste produce rebotes, que provocan que la interrupción se ejecute dos veces en vez de una sola.