Fiuba
Facultad de Ingeniería
Universidad de Buenos Aires

Laboratorio de Microprocesadores - 86.07

Trabajo Práctico Nº4:

Interrupción Externa

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:	08/07/2020

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 puertos utilizando las interrupciones externas de las que dispone el microcontrolador, a fin de manejar los estados lógicos en los que se encuentran los pines de un puerto de salida de éste.

2. Descripción

Mediante la implementación de una placa de microcontrolador de código abierto basado en el microchip ATmega328P, denominada Arduino Uno, se busca controlar dos LEDs conectados en los pines PB0 y PB1 por medio del accionar de un pulsador conectado en el pin PD2 correspondiente a la interrupción externa 0 del microcontrolador (INT0).

El funcionamiento del trabajo a desarrollar radica en:

- Encender el LED_0 (conectado a PB0) mediante el programa principal.
- Detectar el accionar del pulsador por flanco descendente.
- Mediante la rutina de interrupción, apagar el LED_0, hacer parpadear el LED_1 (conectado a PB1) 5 veces a una frecuencia aproximada de $1Hz$ y volver a encender el LED_0.

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:

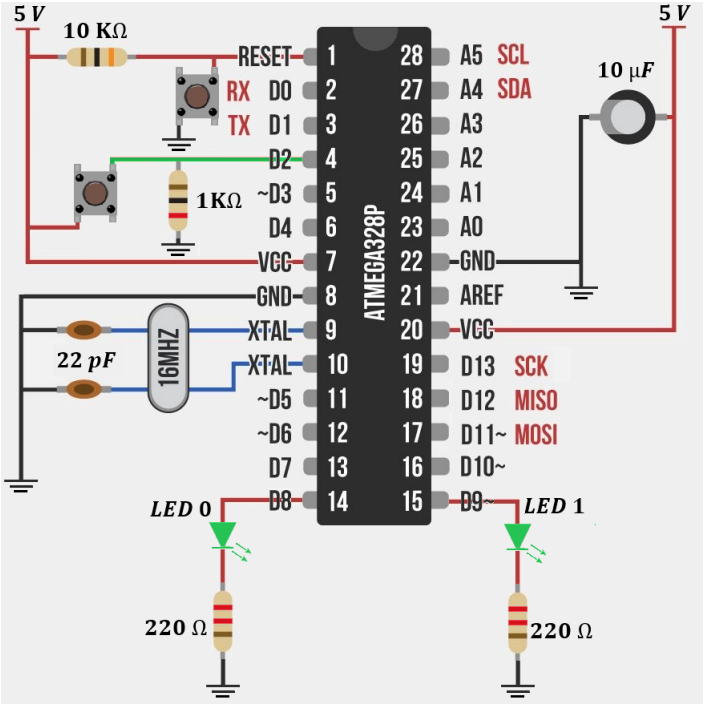

Figura 2: Circuito esquemático.

5. Listado de componentes

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

- 2 LED verde de 5mm
- 2 Resistores de 220 Ω
- 1 Resistor de 1 kΩ
- 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:

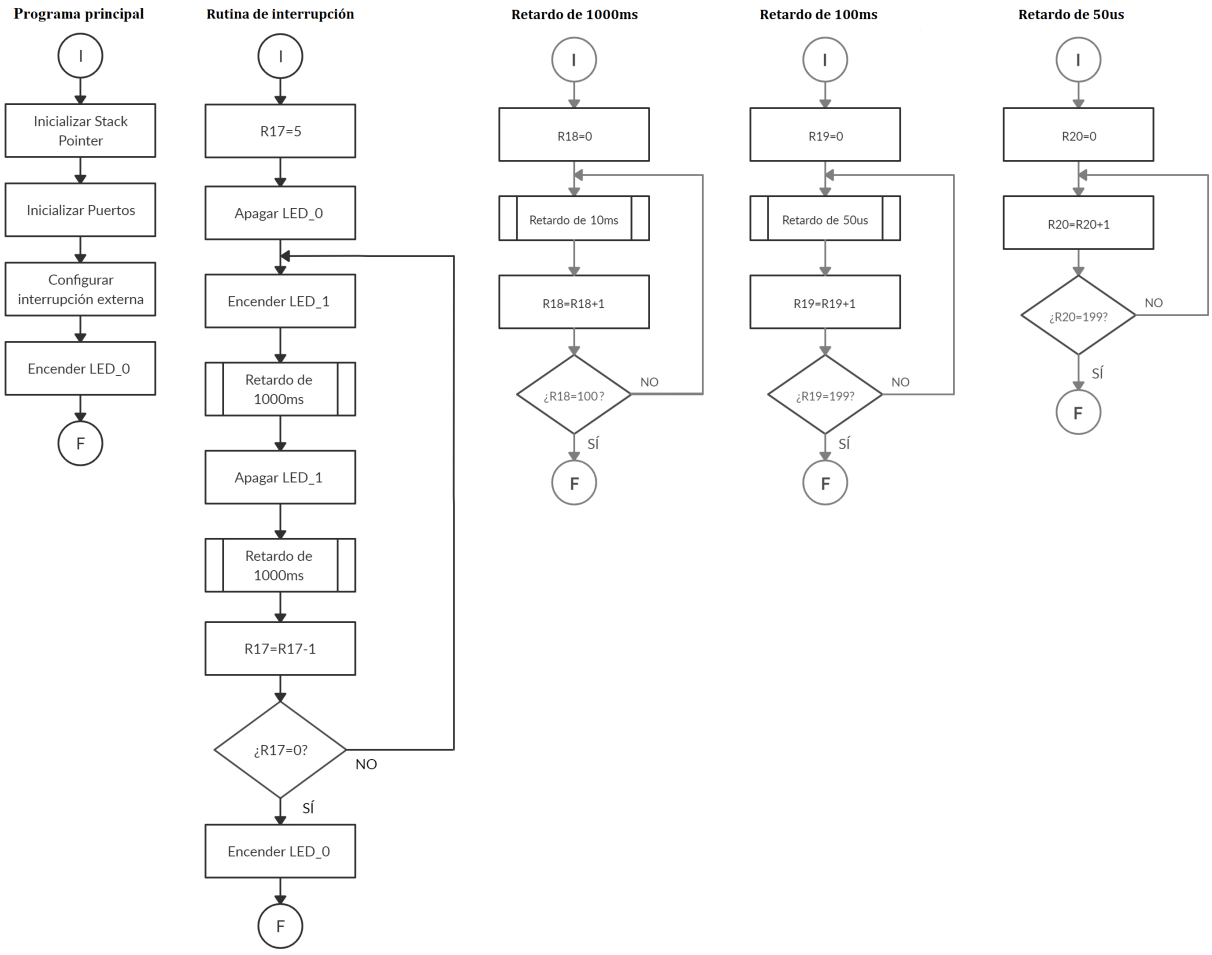

Figura 3: Diagramas de flujo.

7. Código de programa

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

7.1. Uso de resistor pull-down externo

```
.INCLUDE "m328pdef.inc"

.DEF AUX=R16
.DEF CONTADOR=R17

.CSEG
.ORG 0X0000
      RJMP  config

.ORG INT0addr
      RJMP  isr_int0

.ORG INT_VECTORS_SIZE

config:
      LDI       AUX, HIGH(RAMEND)          ;Cargo el SPH
      OUT       SPH, AUX
      LDI       AUX, LOW(RAMEND)           ;Cargo el SPL
      OUT       SPL, AUX

      LDI       AUX, 0XFF
      OUT       DDRB, AUX                  ;Declaro al puerto B como salida
      LDI       AUX, 0X00
      OUT       PORTB, AUX                 ;Inicializo al puerto B en 0X00


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

      LDI       AUX, (1 << ISC01)          ;Configuro IE0 por flanco descendente
      STS       EICRA, AUX
      LDI       AUX, (1 << INT0)           ;Habilito IE0
      OUT       EIMSK, AUX

      SEI                                  ;Habilito interrupciones globales

main:
      SBI       PORTB, 0                   ;Enciendo el LED_0 conectado a PB0

loop:
      RJMP  loop

isr_int0:
      LDI       CONTADOR, 5
      CBI       PORTB, 0                   ;Apago el LED_0 conectado a PB0

loop_parpadeo:
      SBI       PORTB, 1                   ;Enciendo el LED_1 conectado a PB1
      RCALL retardo_1000ms
      CBI       PORTB, 1                   ;Apago el LED_1 conectado al PB1
      RCALL retardo_1000ms

      DEC       CONTADOR
      BRNE  loop_parpadeo                  ;Si LED_1 no parpadeo 5 veces, repito el ciclo

      SBI     PORTB, 0                     ;Enciendo el LED_0 conectado a PB0
      RETI

;*******************************************************************************************
;                 Retardo de 1000ms calculado con un cristal de 16MHz
;El loop interno se debe ejecutar un numero X de veces para que el tiempo total sea 1000ms.
;Considerando los ciclos de máquina que conlleva cada instrucción ejecutada:
;1000ms=(1/16MHz)*(1+X*160004-1+4) ----> X=100
;*******************************************************************************************
```

```
retardo_1000ms:
        CLR     R18                             ;1 CM
loop_retardo_1000ms:
      RCALL   retardo_10ms                      ;160000 CM
      INC     R18                               ;1 CM
      CPI     R18, 100                          ;1 CM
      BRNE    loop_retardo_1000ms               ;2 CM
                                                ;1 CM (BRNE conlleva 1CM cuando R18=100)
      RET                                       ;4 CM


;*****************************************************************************************
;                 Retardo de 10ms calculado con un cristal de 16MHz
;El loop interno se debe ejecutar un numero X de veces para que el tiempo total sea 10ms.
;Considerando los ciclos de máquina que conlleva cada instrucción ejecutada:
;10ms=(1/16MHz)*(1+X*804-1+4) ----> X=199
;*****************************************************************************************
retardo_10ms:
      CLR     R19                               ;1 CM
loop_retardo_10ms:
      RCALL   retardo_50us                      ;800 CM
      INC     R19                               ;1 CM
      CPI     R19, 199                          ;1 CM
      BRNE    loop_retardo_10ms                 ;2 CM
                                                ;1 CM (BRNE conlleva 1CM cuando R19=199)
      RET                                       ;4 CM


;*****************************************************************************************
;                 Retardo de 50us calculado con un cristal de 16MHz
;El loop interno se debe ejecutar un numero X de veces para que el tiempo total sea 50us.
;Considerando los ciclos de máquina que conlleva cada instrucción ejecutada:
;50us=(1/16MHz)*(1+X*4-1+4) ----> X=199
;*****************************************************************************************
retardo_50us:
      CLR     R20                               ;1 CM
loop_retardo_50us:
      INC     R20                               ;1 CM
      CPI     R20, 199                          ;1 CM
      BRNE    loop_retardo_50us                 ;2 CM
                                                ;1 CM (BRNE conlleva 1CM cuando R20=199)
      RET                                       ;4 CM
```

7.2. Uso de resistor pull-up interno

```
.INCLUDE "m328pdef.inc"

.DEF AUX=R16
.DEF CONTADOR=R17

.CSEG
.ORG 0X0000
      RJMP  config

.ORG INT0addr
      RJMP  isr_int0

.ORG INT_VECTORS_SIZE

config:
      LDI       AUX, HIGH(RAMEND)           ;Cargo el SPH
      OUT       SPH, AUX
      LDI       AUX, LOW(RAMEND)            ;Cargo el SPL
      OUT       SPL, AUX

      LDI       AUX, 0XFF
      OUT       DDRB, AUX                   ;Declaro al puerto B como salida
      LDI       AUX, 0X00
```

```
24      OUT     PORTB, AUX              ;Inicializo al puerto B en 0X00
25
26
27      LDI     AUX, 0X00
28      OUT     DDRD, AUX               ;Declaro al puerto D como entrada
29      LDI     AUX, 0XFF
30      OUT     PORTD, AUX              ;Activo las resistencias pull-up del puerto D
31
32
33      LDI     AUX, (1 << ISC01)       ;IE0 por flanco descendente
34      STS     EICRA, AUX
35      LDI     AUX, (1 << INT0)
36      OUT     EIMSK, AUX              ;Habilito IE0
37
38      SEI                             ;Habilito interrupciones globales
39
40  main:
41      SBI     PORTB, 0                ;Enciendo el LED_0 conectado a PB0
42
43  loop:
44      RJMP  loop
45
46  isr_int0:
47      LDI     CONTADOR, 5
48      CBI     PORTB, 0                ;Apago el LED_0 conectado a PB0
49
50  loop_parpadeo:
51      SBI     PORTB, 1                ;Enciendo el LED_1 conectado a PB1
52      RCALL retardo_1000ms
53      CBI     PORTB, 1                ;Apago el LED_1 conectado al PB1
54      RCALL retardo_1000ms
55
56      DEC     CONTADOR
57      BRNE  loop_parpadeo             ;Si LED_1 no parpadeo 5 veces, repito el ciclo
58
59      SBI     PORTB, 0                ;Enciendo el LED_0 conectado a PB0
60      RETI
```

8. Resultados

Luego de desarrollar el código descripto en las secciones anteriores y armar el circuito dispuesto, se logra controlar el encendido y apagado del led mediante el accionar y reposo del pulsador posicionado en uno de los puertos de entrada del microcontrolador asociado a las interrupciones externas de este.

Por otra parte, se consigue analizar que para utilizar la resistencia pull-up interna del puerto de entrada, se debe modificar el circuito de la siguiente manera:

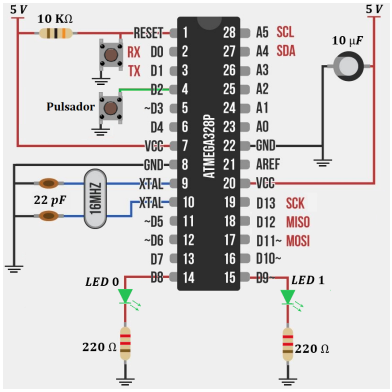

Figura 4: Circuito esquemático usando resistencia interna pull-up.

De esta forma ya no es necesario utilizar el resistor de $1k\Omega$ externo, sin embargo, se debe modificar el programa. Para empezar, luego de indicar que pines se utilizan como entradas y salidas, se debe poner un 1 en el bit 2 del registro PORTD para así poder habilitar la resistencia interna.

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

Componentes	Costos
Leds verde de 5mm	$14,00
Resistores de 220 Ω	$11,00
Cables de conexión	$12,00
Protoboard	$232,00
Arduino Uno	$720,00
TOTAL	$989,00

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.

Si bien en este caso de aplicación, se utilizan solo los pines de un determinado puerto, tanto para encender el led, como para detectar el estado del pulsador, el desarrollo del programa da una idea bastante completa de que cosas deberían modificarse en este si fuera necesario utilizar varios puertos. Por otra parte, hacer uso de las resistencias pull-up internas es una práctica cuya utilidad se extiende más allá de este trabajo, y puede ser útil a futuro.

Así mismo, la implementación de una interrupción externa para la detección del estado en que se encuentra el pulsador, resulta de gran utilidad al momento de tener que analizar otros problemas en los que no se pueda o no sea conveniente usar la técnica de polling.