Laboratorio de Microprocesadores - 86.07

Trabajo Práctico Obligatorio N°4

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									
Ivan Eric	Rubin	100 577									

Observaciones:

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

Fecha de aprobación				Firma J.T.P

Coloquio	
Nota final	
Firma profesor	

1. Objetivo del trabajo

El objetivo de este trabajo es iniciar una rutina con LEDs a partir de una interrupción externa generada por un pulsador.

2. Descripción del trabajo

Se conectara un pulsador al Arduino UNO en uno de los pines de interrupciones externa que posee el microprocesador ATMEGA328P. Esto iniciará una subrutina que apagará un Led que comienza encendido y hará titilar otro 5 veces antes de volver al estado inicial (El primer led encendido y el otro apagado).

3. Diagrama de conexiones en bloques

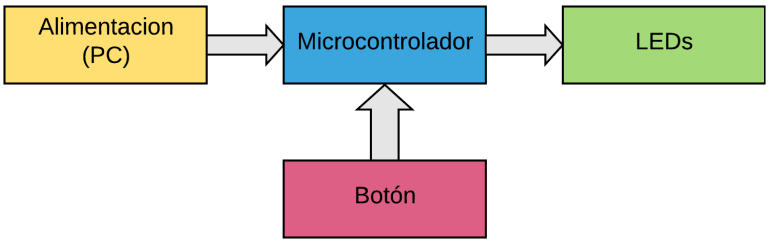

Figura 3.1: Diagrama de bloques.

4. Circuito Esquemático

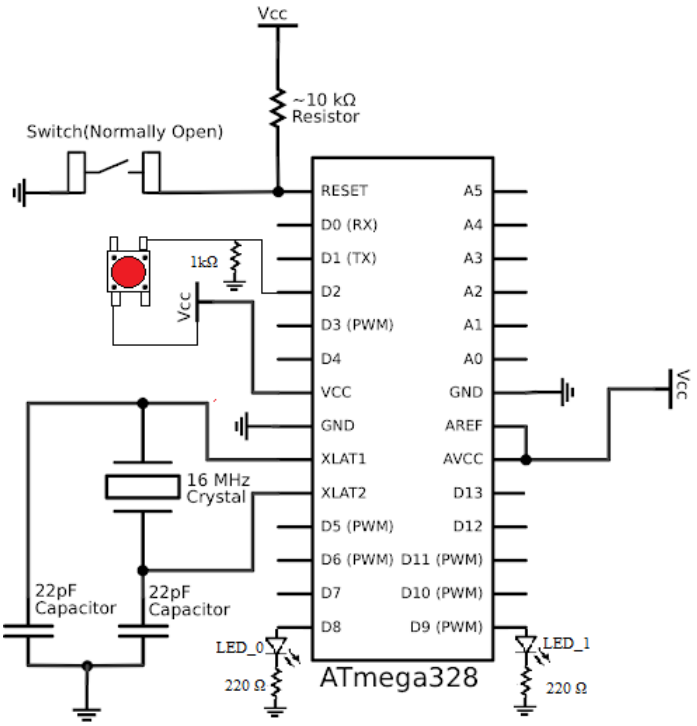

Figura 4.1: Circuito Esquemático.

5. Listado de componentes

- Arduino UNO + Conector PC (864$).
- Microprocesador ATMEGA 328P (Integrado en Arduino).
- Resistencia de $220\Omega \pm 1\,\%$ (2 x 10$).
- Resistencia de $1k\Omega$ (5$)
- Un pulsador (20$)
- Diodo Emisor de Luz (LED) (2 x 20$).
- Cables Macho-Macho (40 cables x 100$).
- Protoboard (200$).

6. Diagrama de flujo del software

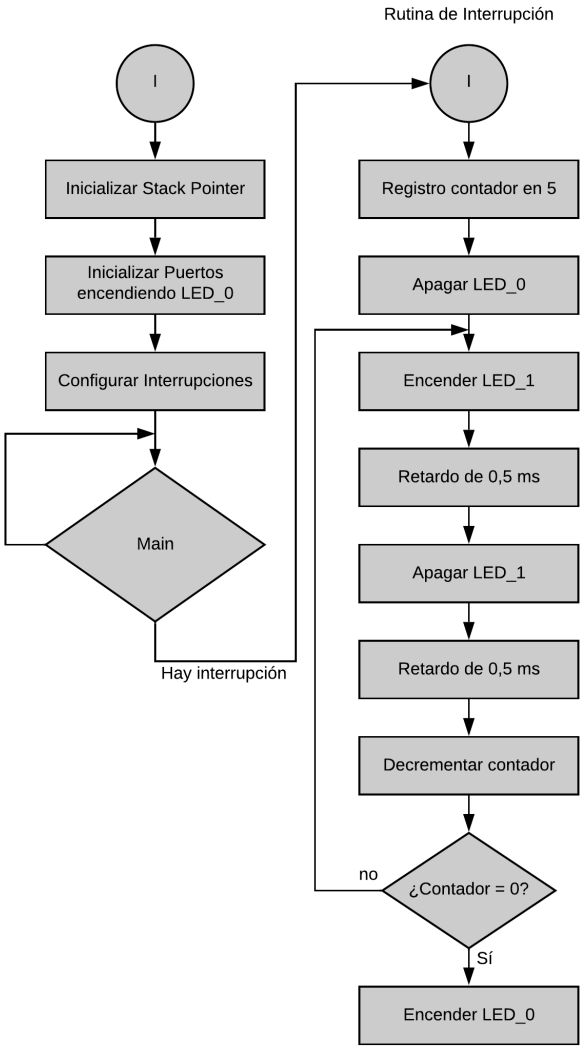

Figura 6.1: Diagrama de flujo.

7. Código de programa

```
/*
 *
   Trabajo Práctico Obligatorio 4
   Autor: Ivan Eric Rubin
 *
 */

.include "m328pdef.inc"

.def dummyr = R16
.def count = R17
.def delay_reg1 = R18
.def delay_reg2 = R19
.def delay_reg3 = R20

.equ pinLED_0 = 0
.equ pinLED_1 = 1

; Código en memoria Flash
.cseg
.org 0x0000
   jmp config_puertos

.org INT0addr
   jmp   hay_int0

.org INT_VECTORS_SIZE

config_puertos:

   ldi   dummyr, HIGH(RAMEND)
   out   SPH, dummyr
   ldi   dummyr, LOW(RAMEND)
   out   SPL, dummyr

   ldi dummyr, 0xFF           ; Declaro al puerto D como entrada
   out DDRD, dummyr
   ldi dummyr, 0x00           ; Cargo 0x04 si quiero activar la resistencia de
   pullup
   out PORTD, dummyr

   ldi   dummyr, 0x00            ; Declaro al puerto B como salida
   out   DDRB, dummyr
   ldi   dummyr, 0x01            ; Arranca encendido el LED_0
   out   PORTB, dummyr

   ; Configuro las interrupciones por flanco descendente
   ldi   dummyr, (1 << ISC01)
   sts   EICRA, dummyr
   ldi   dummyr, (1 << INT0)
   out   EIMSK, dummyr           ; Habilito IE0

   sei                        ; Habilito las interrupciones globales

main:
   rjmp main

hay_int0:
   ldi count, 5
```

```
59      cbi PORTB, pinLED_0
60
61  blink:
62      sbi PORTB, pinLED_1
63      rcall delay                 ; Retardo de 0,5 segundos
64      cbi PORTB, pinLED_1
65      rcall delay                 ; Retardo de 0,5 segundos
66      dec count
67      brne blink
68
69      sbi PORTB, pinLED_0
70      reti
71
72  delay:
73          ldi delay_reg1, 32
74
75  loop1:   ldi delay_reg2, 250
76  loop2:   ldi delay_reg3, 250
77  loop3:   nop                      ; 4 Ciclos (250 nS)
78
79          dec delay_reg3          ; 250 nS x 250 = 62.5 uS
80          brne loop3
81          dec delay_reg2          ; 62.5 uS x 250 = 15.625 mS
82          brne loop2
83          dec delay_reg1          ; 15.625 mS x 32 = 0.5 S
84          brne loop1
85
86          ret
```

8. Resultados

Se pudo lograr la interrupción con la configuración de Pull-Down. Si se configurara con Pull-Up, Habria que inicializar el puerto D poniendo un 1 en el bit que representa el pin de interrupción utilizado (Como se explica en un comentario del código). También se podría configurar las interrupciones por flanco ascendente pero al ser un pulso el que interrumpe hay tanto un flanco ascendente como uno descendente (En ambas configuraciones se detectan ambos flancos). Por esto no se consideró necesario para la configuración Pull-Up.

Para lograr esta configuración se conectó la pata del botón que esta conectada a VCC a tierra y se quitó la resistencia de $1k\Omega$ (ahorrándola). Se puede ver en el video adjuntado en GitHub el funcionamiento con ambas configuraciones.

9. Conclusiones

Se destaca de este trabajo la utilización de las interrupciones externas y su mejor entendimiento en la aplicación pedida. Se puede ver la utilidad de esto ya que no se debe codificar un loop para esperar un cambio de flanco sino que esta interrupción puede ocurrir en cualquier momento del código. Esto hace que sea necesario hacer un código robusto para que no haya momento inoportuno en el cual se detecte una interrupción.