FACULTAD DE INGENIERIA

Universidad de Buenos Aires

Laboratorio de Microprocesadores - 86.07

Trabajo Práctico N°4:

Interrupción Externa

Profesor:			Ing. Guillermo Campiglio							
Cuatrimestre/Año:			1°/2020							
Turno de las clases prácticas			Miércoles							
Jefe de trabajos prácticos:			Ing. Pedro Ignacio Martos							
Docente guía:			-							
Autor			Seguimiento del proyecto							
Francisco	Rossi	99540								

Observaciones:

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

Fecha de aprobación				Firma J.T.P

Coloquio	
Nota final	
Firma profesor	

Índice

1 Introducción

En el siguiente informe se explica el diseño de un programa escrito en lenguaje Assembler para hacer parpadear un LED cinco veces a una velocidad aproximada de 1 Hz conectado al pin *PB1* de un *ATMEGA328p* cuando se genera una interrupción externa.

1.1 Objetivo

El objetivo es configurar el microcontrolador para que ante una interrupción externa ejecute una subrutina que realice una acción sobre los puertos I/O del mismo.

1.2 Descripción

Se conecta un LED con un resistor en serie al pin *PB0*, el mismo se apaga cuando se genera una interrupción externa y se enciende cuando termina la subrutina asociada, al generarse la interrupción además de apagarse el LED en *PB0* se hace titilar 5 veces a una frecuencia aproximada de 1 Hz a un LED conectado al pin *PB1*.

Para la interrupción externa se utiliza un pulsador conectado a $INT0$ (*PD2*).

2 Materiales

Se utilizaron los siguiente materiales para el proyecto:

a. 1 Pulsador (30$ (Pesos Argentinos))

b. 2 LEDs (40$ (Pesos Argentinos))

c. 2 Resistor de 220Ω (8$ (Pesos Argentinos))

d. 1 Microcontrolador ATmega328p (Utilizando el integrado con el Arduino Uno) (700$ (Pesos Argentinos))

3 Diagrama en Bloques

En la **Fig. 1** se muestra un diagrama en bloques del circuito.

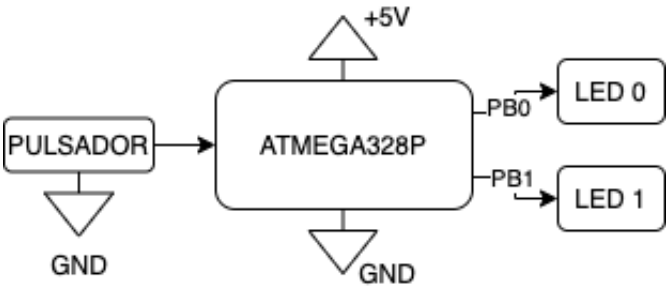

Figure 1: Diagrama en bloques.

4 Esquemático

En las **Fig. 4** y **Fig. 2** se muestra como se conectó el arduino con los LEDs, el pulsador y los resistores de 220 Ω. En las **Fig. 5** y **Fig. 3** el caso en el cual se utiliza un resitor de pull-down externo.

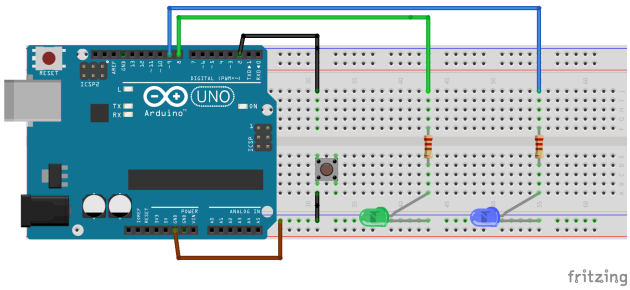

Figure 2: Circuito implementado usando la R interna del puerto.

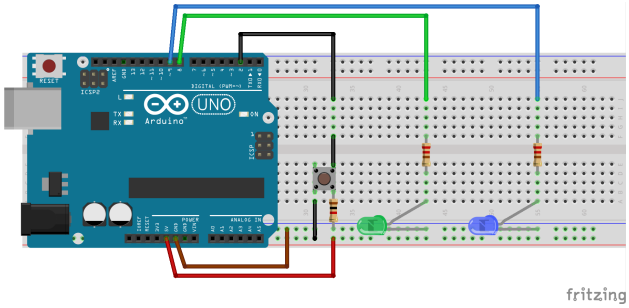

Figure 3: Circuito implementado usando una R externa.

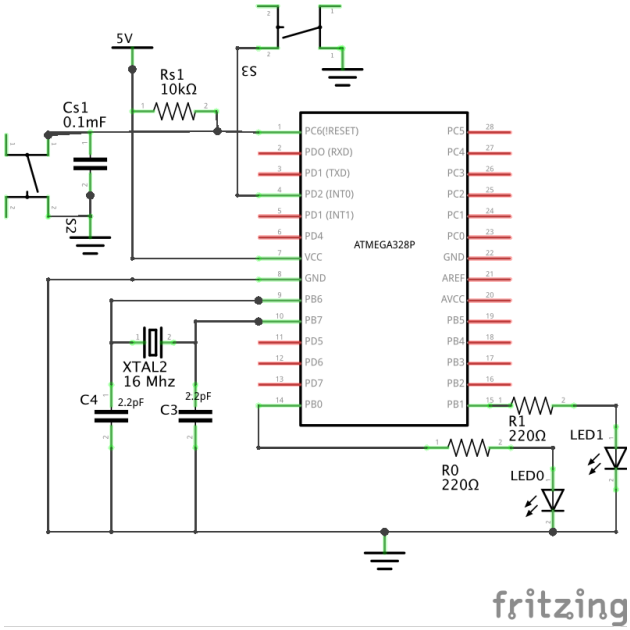

Figure 4: Esquemático del circuito implementado usando la R interna del puerto.

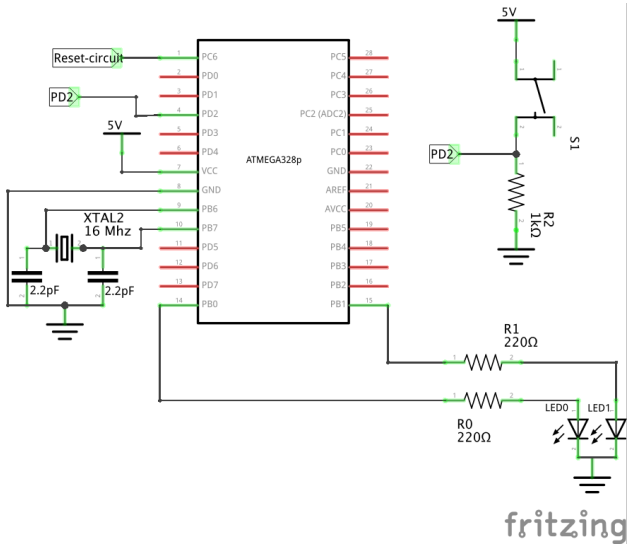

Figure 5: Esquemático del circuito implementado usando una R externa.

5 Diagrama de flujo

En la **Fig. 6** se muestra el diagrama de flujo del programa.

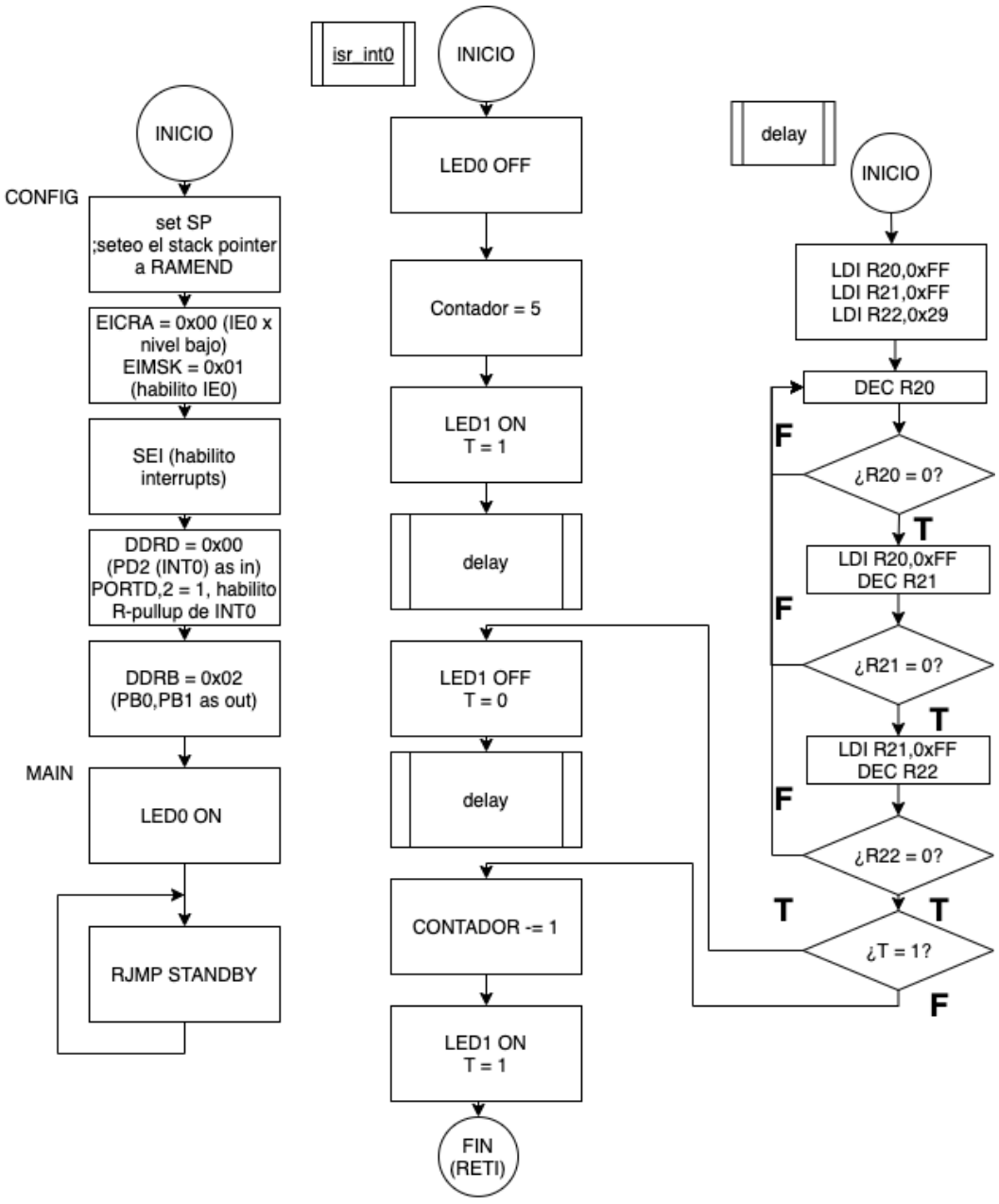

Figure 6: Diagrama de flujo.

6 Códigos

El siguiente código se puede dividir en tres partes, primero la configuración `Config` donde se setea el stack pointer, las interrupciones y los puertos. Luego, el `main` donde se enciende el *LED0* y se deja el micro en espera a una interrupción sin hacer nada. Este estado se modifica solo si se genera una interrupción externa, la cuál es posible generarla con un nivel bajo en INT0, dado por el pulsador correspondiente y *PD2* con R-pullup habilitada.
Por último, cuando la interrupción es generada se dispara la subrutina `isr_int0`, en la cuál se apaga el *LED0* y se hace titilar 5 veces el *LED1* a una frecuencia aproximada de 1 Hz (Ver **Frecuencia de oscilación del LED**), se apaga el *LED0* al salir de la interrupción y el micro vuelve a la posición de stand-by.

En el caso del resistor externo cambia la linea 33: ldi dummy, (1¡¡ISC01—0¡¡ISC00).

```
1  ; Autor: Francisco Rossi
2  ; Padron: 99540
3  ; 86.07 Laboratorio de Microprocesadores - FIUBA
4  ; Catedra: Miercoles
5  ; Fecha: 24 de junio de 2020
6
7  .include "m328pdef.inc"
8
9  .def contador = r16
10 .def dummy = r25
11
12 .macro set_sp
13   ldi dummy, low(RAMEND)
14   out spl, dummy
15   ldi dummy, high(RAMEND)
16   out sph, dummy
17 .endm
18
19 .cseg
20 .org 0x0000
21       jmp config
22
23 ; EXT INT
24
25 .org INT0addr
26       jmp isr_int0
27 .org INT_VECTORS_SIZE
28
29  config:
30       set_sp
31
32       ; configuro interrupcion externa INT0,INT1
33       ldi    dummy,(0 << ISC01 | 0 << ISC00  ) ;0x03 ; IE0 por flanco descendente
34       sts    EICRA,dummy              ;(ISC01=1;ISC00=1)
35       ldi    dummy,(1 << INT0) ;0x01 ; habilito IE0
36       out    EIMSK,dummy
37
38       ;habilito interrupuciones
39       sei ;(I en 1)
40
41       ;Configuracion de puertos
42       ; portd,2 como entrada
43       ldi    dummy, (0 << 2)
44       out    DDRD, dummy
45       ldi    dummy, (1 << 2)
46       out    PORTD,dummy ;(R pull up activa)
47
48       ; PB0 y PB1 como salidas
49       ldi    dummy, (1 << 0 | 1 << 1)
50       out    DDRB, dummy
51       clr    dummy
52       out    PORTB,dummy ; inicializo en cero
53
54 main:
```

```
55          sbi     PORTB,0
56  standby:
57          rjmp    standby
58
59
60  isr_int0:
61          cbi     PORTB,0
62  twink:
63          ldi     contador,0x05
64  on:
65          sbi     PORTB,1
66          set
67          rjmp    delay
68  off:
69          cbi     PORTB,1
70          clt
71          rjmp    delay
72  rt:
73          dec     contador
74          brne    on
75          sbi     PORTB,0
76          reti
77
78  delay:
79          ldi     r20,0xff
80          ldi     r21,0xff
81          ldi     r22,0x29
82  cycle:
83          dec     r20
84          brne    cycle
85          ldi     r20,0xff
86          dec     r21
87          brne    cycle
88          ldi     r21,0xff
89          dec     r22     ; con este valor es facil variar de forma apreciable la frecuencia ya que es un
        multiplicador de todos los incrementos anteriores
90          brne    cycle
91
92          brts    off
93          jmp     rt
```

El ciclo para realizar los retardos `Delay` esta basado en decrementar registros, dependiendo del valor del `bit T` del status register se determina si el *LED1* se debe apagar o prender. En la siguiente sección se explicará en profundidad la relación entre el decremento de los registros y los tiempos de encendido y apagado del LED. El código fue pensado para lograr un tiempo en alto y en bajo similares es decir, que el *Duty Cycle* sea lo más cercano posible al 50%.

7 Frecuencia de oscilación del LED

Para generar una frecuencia de aproximadamente 1 Hz se comenzó utilizando el mismo delay para cuando el *LED1* este encendido y cuando este apagado, de manera de generar una cuadrada de un *Duty Cycle* del 50%.

Dao que la frecuencia por default del clock del *Arduino UNO* es $f_{ck} = 16\ MHz$.

Luego, conociendo los ciclos de clock que toma cada operación del ciclo podemos obtener los tiempos de encendido y apagado como $\Delta_t = \frac{ciclos_{delay}}{f_{ck}}$, y con esto podemos obtener la frecuencia de oscilación del LED como $f = \frac{1}{2\Delta_t}$.

El `Delay` se basa en decrementar 3 registros desde 3 valores máximos, usandolos como un sistema *pesado*:

$$R22R21R20$$

donde por cada vez que se decrementa $R20$ desde $R20_{max}$ hasta cero se decremente una vez $R21$ y cada vez que se decrementa $R21$ desde $R21_{max}$ hasta cero se decrementa $R22$ una vez.

De manera que la cantidad de ciclos totales se podrá calcular como la cantidad de ciclos que se toma en decrementar el registro R20 (CC_{R20}) sumado a la cantidad de ciclos que se toma en decrementar el registro R21 (CC_{R21}) sumado a la cantidad de ciclos que se toma en decrementar el registro R22 (CC_{R22}).

Podemos representar lo dicho anteriormente de manera matemática como:

$$ciclos_{delay} = 3 + CC_{R20} + CC_{R21} + CC_{R22}$$

$$CC_{R20} = R22_{max} \cdot R21_{max} \cdot [R20_{max}\,(2+1)] - \overbrace{\overbrace{R21_{max} \cdot R22_{max}}^{\text{veces que el brne es falso}}}^{\text{veces que el brne es falso}}$$

$$CC_{R21} = R22_{max} \cdot [R21_{max}\,(2+1+1)] - \overbrace{R22_{max}}^{\text{veces que el brne es falso}}$$

$$CC_{R22} = R22_{max} \cdot (1+1+2) - \overbrace{1}^{\text{veces que el brne es falso}}$$

En este caso se utilizaron los siguientes valores:

$$R20_{max} = R21_{max} = 255$$

De manera que el valor más cercano a $f = 1\ Hz$ se obtiene con:

$$R22_{max} = 41 = 0x29$$

Reemplazando por los valores son

Luego,

$$ciclos_{delay} = 3 + 7987620 + 41779 + 163 = 8029565 \implies \Delta_t = \frac{8029565}{16 \cdot 10^6} = 0,5018\ seg$$

Finalmente se obtiene la frecuencia de oscilación del *LED1*:

$$f = \frac{1}{2 \cdot \Delta_t} = 0,9963 Hz$$

Con estos resultados obtenemos la frecuencia de oscilación del LED y el *Duty Cycle*:

$$\boxed{Duty\ Cycle = 50\%}$$

$$\boxed{f = 0,9963\ Hz}$$

8 Resultados

Se logró diseñar un programa para el microcontrolador ATMEGA328p que haga titilar cinco veces un LED colocado en el puerto B cuando se genera una interrupción externa por nivel bajo en *INT0* a una frecuencia de $f = 0,9963\ Hz$ y *Duty Cycle* $\approx 50\%$.

9 Conclusiones

Se logró generar un programa que realicé la tarea de configuración y este listo ante una interrupción externa a correr una subrutina la cual apague un LED en el $PB0$ y haga titilar otro, en $PB1$ cinco veces a una frecuencia aproximada de 1 Hz. Es posible también modificar la frecuencia de oscilación y el duty cycle del mismo.