

## TRABAJO PRÁCTICO N° 4

### Interrupción Externa

16/Jun/2020

---

98934 Vazquez, Rodrigo rvazquez@fi.uba.ar

#### Docentes:

Campiglio, Guillermo Carlos

Stola, Gerardo Luis

Cofman, Fernando

Salaya Velazquez, Juan Guido

---

#### Resumen

En el presente trabajo se desarrollará el diseño de programa que consiste en detectar y manejar la funcionalidad de **interrupción de hardware** del microcontrolador *Atmel MEGA 328P*. Para ello el programa deberá controlar el comportamiento de dos diodos LED a través de una entrada digital que se detectara mediante dichas interrupciones.

---

## Índice

1. Objetivo	2
2. Descripción del proyecto	2
3. Diagrama de conexión en bloques	2
4. Circuito esquemático	3
5. Listado de componentes y tabla de gastos	3
6. Software	3
7. Resultados	3
8. Conclusiones	4
9. Anexo - Código	5

## 1. Objetivo

Realizar programa que realice una secuencia que prenda y apague dos LED detectando el accionar del un botón por flanco. El programa principal debe mantener encendido el primer LED y al detectarse la pulsación del botón realizar una rutina de parpadeo en el segundo de los LED mientras el otro se apague. Al final la rutina de parpadeo, el primer LED se volverá a encender.

## 2. Descripción del proyecto

Se realizará un programa en lenguaje *Assembly* para la familia de microcontroladores *AVR*. Se utilizará la plataforma *Arduino* y el *Shield I/O* para utilizar las conexiones de dos LED y un botón pulsador, en función de esto se eligieron los puertos de entrada y salida.

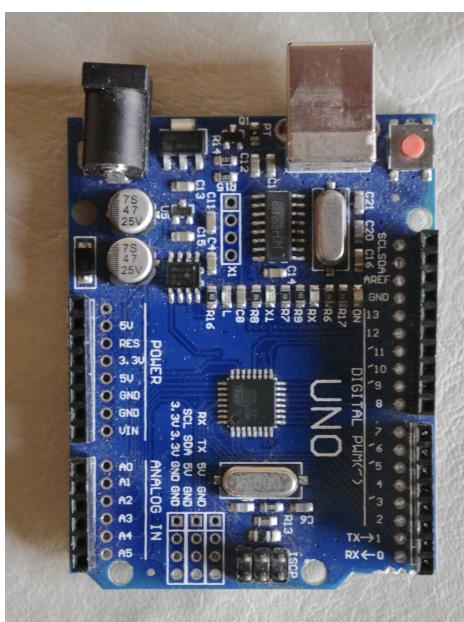


Figura 1: Placa Arduino UNO utilizada



Figura 2: Placa *Shield I/O* utilizada para la conexión LED

## 3. Diagrama de conexión en bloques

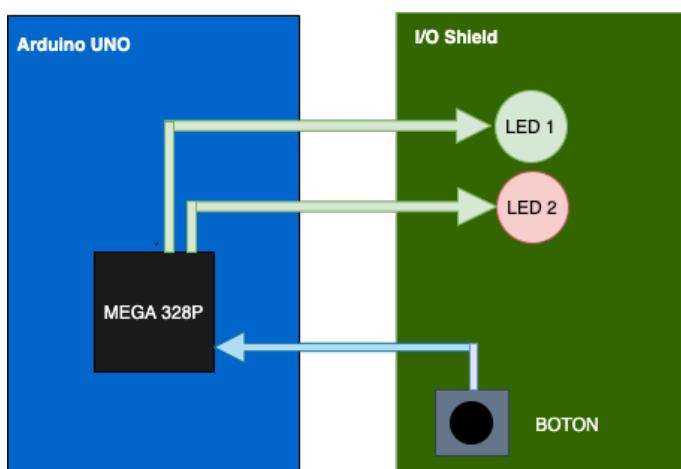


Figura 3: Diagrama de conexión del hardware

## 4. Circuito esquemático

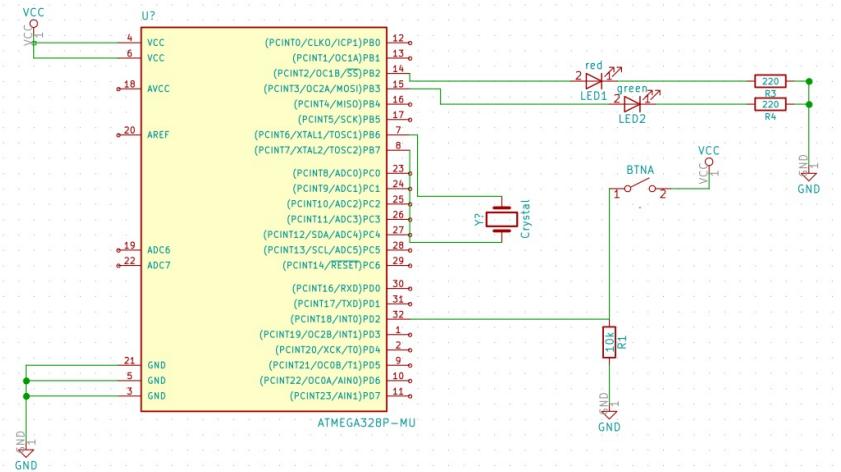


Figura 4: Circuito esquemático

## 5. Listado de componentes y tabla de gastos

Componente	Precio
Arduino UNO	\$750
I/O Shield	\$950

Tabla 1: Tabla de componentes

## 6. Software

El programa comienza con la configuración de los puertos B y D según la conexión del *Shield* como salida y entrada respectivamente. Luego, a diferencia de los trabajos anteriores, se realizó la configuración de las interrupciones. Esto consiste en cargar el registro EICRA con unos en los bits correspondientes a la interrupción INT0, de esta manera programando que la interrupción se realice con flanco ascendente. Luego se cargo el registro EIMSK el cual se utiliza para activar las interrupciones, por lo que se activo la correspondiente a INT0. Habiendo así terminado la configuración correspondiente a interrupciones agregando la instrucción SEI en la subrutina llamada *INIT\_INT0*, se procedió a la configuración de entrada y salida en la subrutina *INIT\_HARDW*. Luego se comienza con el *loop* principal el cual solo consiste en encender el **LED 1**. La subrutina que corresponde a la interrupción, llamada *parpadeo*, primero se llama a una subrutina llamada *debounce* en la cual se realiza una pequeña espera y luego se hace una segunda lectura del pin de entrada, si la segunda lectura no coincide con la primera, se sale de la interrupción. Retomando con la rutina de interrupcion, luego, se apaga el **LED 1** y luego se hace parpadear el **LED 2** 5 veces con esperas de 500 ms así logrando una frecuencia de parpadeo de 1 Hz. Terminada la secuencia de parpadeo, se vuelve a encender el **LED 1** y así se termina la secuencia de interrupción, volviendo al *loop* principal.

## 7. Resultados

Se logró realizar la animación y visualizar correctamente el comportamiento esperado. Aunque hubo dificultades el principio por haber omitido la instrucción SEI, esto provocó un comportamiento errático del circuito y en definitiva sin cumplir lo indicado. Luego de detectado este error, el circuito funcionó correctamente. Por ultimo, se pregunta qué se modificaría si se cambian las resistencias de *pulldown* por *pullup*, esto cambiaría el estado lógico normalmente leído por el circuito, por lo que en principio se tendría

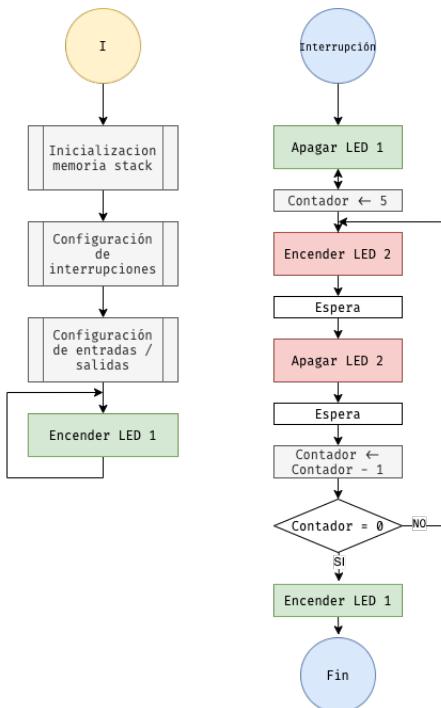


Figura 5: Diagrama de flujo del programa e interrupción para controlar dos LED con botón pulsador

que cambiar el evento de lectura de flanco ascendente por descendente y luego ajustar la segunda lectura dentro la subrutina de *debounce*.

## 8. Conclusiones

Las principales conclusiones de este trabajo es en el manejo apropiado del mecanismo de interrupciones del microcontrolador *Atmel Mega 328P* vía el lenguaje de *Assembly* con especial atención dado que para el correcto funcionamiento de interrupciones, se llevan a cabo varias configuraciones sobre distintos registros de *I/O* y su uso conlleva mas código del hasta ahora utilizado, para la configuración de los parámetros interrupción. Cumpliendo con esto, el uso de interrupciones brinda una claridad de funcionalidad y eficiencia que hace valer su implementación.

## Bibliografía

MAZIDI, Muhammad Ali; NAIMI, Sarmad; NAIMI, Sepehr (2011). "The AVR microcontroller and embedded systems. Embedded system using Assembly and C". Pearson Education Inc. New Jersey.

## 9. Anexo - Código

```

1 .include "m328Pdef.inc"
2
3 .equ BTN = 2      ;PD2
4 .equ LED_1 = 2    ;PB2
5 .equ LED_2 = 3    ;PB3
6 .equ LED_PORT = PORTB
7 .equ BTN_PIN = PIND
8 .def COUNTER = r20
9
10 .org 0
11   rjmp reset
12 .org INT0addr
13   rjmp parpadeo
14
15 reset:
16 ; Inicializador de stack
17 ldi r16, HIGH(RAMEND)
18 out SPH, r16
19 ldi r16, LOW(RAMEND)
20 out spl, r16          ; inicializa el stack al fondo de la memoria
21 clr r16
22 ######
23 call INIT_INTERRUPT
24 call INIT_HARDW
25
26 ######
27
28 ###### Main Loop#####
29 main:
30   sbi LED_PORT, LED_1
31   rjmp main
32
33
34 ###### Interrupt Service Routine#####
35 parpadeo:
36   call debounceINTO
37   cbi LED_PORT, LED_1
38   ldi COUNTER, 5
39   loop:
40     sbi LED_PORT, LED_2
41     call DELAY_500ms
42     cbi LED_PORT, LED_2
43     call DELAY_500ms
44     dec COUNTER
45     brne loop
46     sbi LED_PORT, LED_1
47   reti
48
49 debounceINTO:
50   call DELAY_1ms
51   sbis BTN_PIN, BTN
52   reti
53   ret
54
55
56 ###### Hardware Config#####
57 INIT_INTERRUPT:
58   push r16
59   ;Establece que la interrupcion es por flanco asc de INT0

```

```

60    ldi r16, (1<<ISC01)|(1<<ISC00) ; 11 es flanco asc
61    sts EICRA, r16
62    ;Habilita a INT0 a ser interrupcion
63    ldi r16, (1<<INT0)
64    out EIMSK, r16
65    pop r16
66    sei
67    ret
68
69 INIT_HARDW:
70    ;Pone unos en las posiciones de los pines declarados
71    push r16
72    sbr r16, (1<<LED_1)|(1<<LED_2)
73    out DDRB, r16
74    sbr r16, (1<<BTN)
75    out DDRD, r16
76    pop r16
77    ret
78
79 ;#####DELAYS#####
80 DELAY_1ms:
81    push r17
82    push r18
83    ldi r18, 21
84 L2:
85    ldi r17, 255
86 L1:
87    dec r17
88    brne L1
89    dec r18
90    brne L2
91    pop r18
92    pop r17
93    ret
94 ;#####
95 DELAY_500ms:
96    push r16
97    push r17
98    push r18
99    ldi r18, 41
100   LOOP3:           ; repite LOOP2 9 veces = 1,00045 s
101   ldi r17, 255
102   LOOP2:           ; repite LOOP1 por 89 veces x 1249us= 111.16 ms
103   ldi r16, 255
104   LOOP1:           ; (5x250)-1 =1249us
105   dec r16
106   brne LOOP1
107   dec r17
108   brne LOOP2
109   dec r18
110   brne LOOP3
111   pop r18
112   pop r17
113   pop r16
114   ret

```