Fiuba
Facultad de Ingeniería
Universidad de Buenos Aires

(6609) LABORATORIO DE MICROCOMPUTADORAS

Proyecto:
Interrupciones

Profesor:	**Ing. Pedro Ignacio Martos**
Cuatrimestre / Año:	**Segundo/ 2020**
Turno de clases prácticas:	**miércoles**
Jefe de Trabajos Prácticos:	**Ing. Fernando Pucci**

Autores			**Seguimiento del proyecto**								
Nombre	**Apellido**	**Padrón**									
Joseph	**Valer Torres**	**98330**									

Observaciones:

Fecha de aprobación		

Firma J.T.P.

COLOQUIO	
Nota final	
Firma Profesor	

1 Introducción

Las interrupciones externas en el ATmega32 son activadas con los pines INT0 y INT1, en caso de que se habiliten las interrupciones los pines INT siempre activaran alguna interrupción sin importar como se haya configurado el puerto en el que estos pines se encuentren. Las interrupciones externas se habilitan cuando la entrada del pin, cambia de estado, se puede configurar si se requiere que se active cuando cambia de un estado bajo a uno alto o viceversa y también por flancos.

La interrupción externa AVR es útil para el manejo de pulsadores, detectores de cruce por 0, teclados matriciales y mucho más; hay 2 tipos de interrupciones externas en los microcontroladores AVR: INT0 e INT1, son pines que están preparados para producir una interrupción externa AVR por diversos eventos (que se pueden configurar por programa), mientras que los pines nombrados como PCINT0, PCINT1 así hasta PCINT23, son pines que están preparados para producir una interrupción externa AVR,cuando en estos pines se produce un cambio de estado, esto es si sus estados pasan de alto a bajo o de bajo a alto; las interrupción externa AVR se producirá no importando si el pin elegido es una entrada o salida digital.

2 Diagrama de bloques

relación de los componentes físicos que se utilizaron para el proyecto.

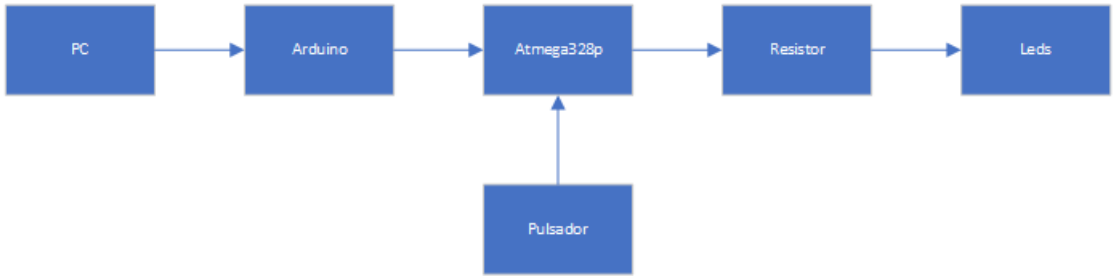

Figura 1: Diagrama de bloques

3 Diagrama de flujos del software

Diagrama de flujo del programa grabado en la memoria del microcontrolador.

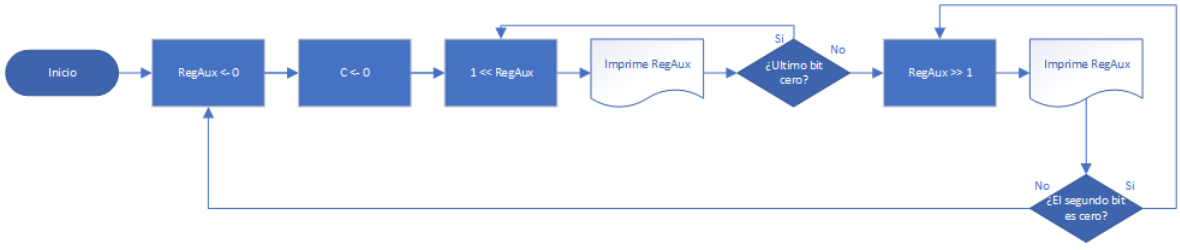

Figura 2: Diagrama de flujo de programa principal

Figura 3: Diagrama de flujo de interrupción 0

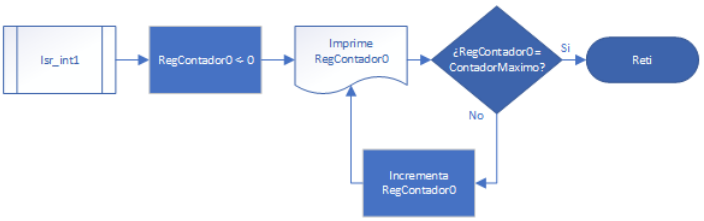

Figura 4: Diagrama de flujo de interrupción 1

4 Diagrama circuital

Conexiones circuitales de los distintos componentes utilizados para la realización del trabajo.

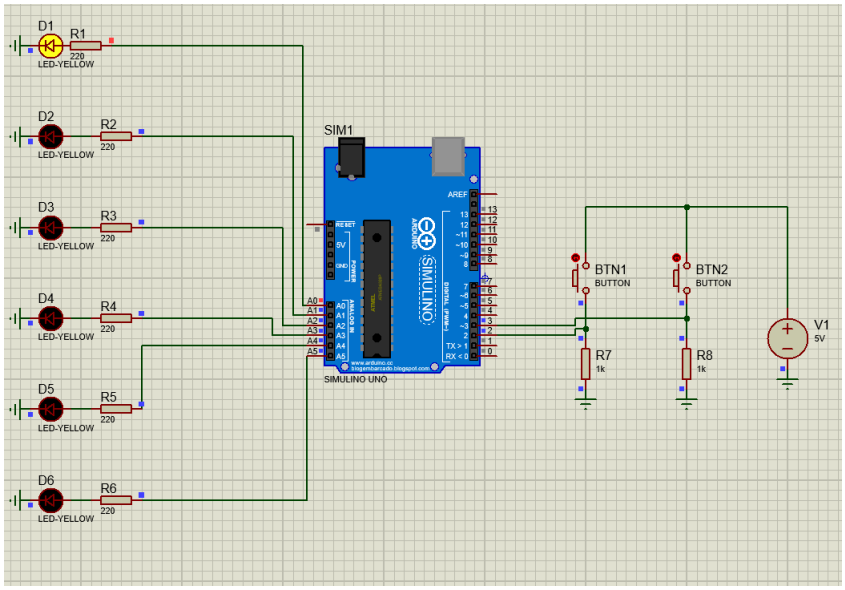

Figura 5: Esquemático de las conexiones realizadas

5 Preguntas

¿ si se presiona primero el pulsador 1 y luego inmediatamente el 2 que pasa?
El programa fue realizado para que no ocurran interrupciones anidadas, por lo tanto al presionar el pulsador 1 y seguido el 2 se ejecutara solo la primera interrupción y la otra sera descartada. Para que ocurran las interrupciones anidadas haría falta activar el flag de interrupciones globales una vez se ejecuta la rutina de la primera interrupción.
¿ Y si es al a inversa ?
De igual forma que en el caso anterior, se ejecutara la rutina de la interrupción 2 y la primera se descartara.
¿Si se presionan los dos al mismo tiempo?
Solo se ejecutara una de las interrupciones, en el registro EIFR se activaran los flags de las 2 interrupciones.

6 Resultados y conclusiones

Las interrupciones se ejecutan cuando existen cambios en las entradas INT0 o INT1, esto depende de la configuración de las interrupciones (flanco o cambio de estado), este método,

comparado con el polling, es mas rápido y eficiente por que evita preguntar periódicamente si hubo una interrupción, por lo tanto consume menos recursos al microcontrolador.

Se puede modificar el programa para habilitar las interrupciones anidadas, haría falta activar las interrupciones globales, en el registro de estado, al comienzo del código de las rutinas de interrupción

7 Código

```
.include "m328pdef.inc"

.def RegAux = R16
.def RegContador0 = R20
.def RegContador1 = R21

.equ UltimoBit = 5  ; Bit de rebote en el extremo superior
.equ PrimerBit = 0  ; Bit de rebote en e extremo inferior
.equ PuertoSalida = PORTC
.equ PuertoConfiguracion = DDRC
.equ Repeticiones = 15
.equ ContadorMaximo = 63


.cseg
.org 0x0000
  jmp Configuracion

.org INT0addr
  jmp isr_int0

.org INT1addr
  jmp isr_int1

.org INT_VECTORS_SIZE


Configuracion:

  /* Inicializacion de stack pointer*/
  ldi RegAux, LOW(RAMEND)
  out spl, RegAux
  ldi RegAux, HIGH(RAMEND)
  out sph, RegAux

  /* configuracion interrupciones */
  ldi RegAux, (1<<ISC01 | 1<<ISC11) ; La interrupcion es por flanco
   descendente
  sts EICRA, RegAux

  ldi RegAux, (1<<INT0 | 1<<INT1)   ; Se activa la interrupcion 0/1
  out EIMSK, RegAux

  SEI

  /* Configuraion de puerto */
  ldi RegAux, 0xff
  out PuertoConfiguracion, RegAux

  /* Inicializacion de puerto*/
```

```
  clr RegAux
  out PuertoSalida, RegAux

inicio:
  clr RegAux
  sec
ida:
  rol RegAux
  call delay500mseg

  out PuertoSalida, RegAux
  sbrc RegAux, UltimoBit
  jmp vuelta
  jmp ida
vuelta:
  ror RegAux
  call delay500mseg

  out PuertoSalida, RegAux
  sbrc RegAux, (PrimerBit+1)
  jmp inicio
  jmp vuelta


final:  rjmp final

delay10ms:  ; para 16Mhz
  push RegContador0
  push RegContador1

  ldi RegContador0, 209
  LOOP0:
    ldi RegContador1, 255
    LOOP1:
      dec RegContador1
      brne LOOP1

    dec RegContador0
    brne LOOP0

  pop RegContador1
  pop RegContador0
ret

delay500mseg:
  push RegContador0

  ldi RegContador0, 50
retardo:
  call delay10ms
  dec RegContador0
  brne retardo

  pop RegContador0
ret

isr_int0:
  push RegAux

  ldi RegContador0, Repeticiones
```

```
repetir:
  cpi RegContador0, 0x00
  breq regresar
  ldi RegAux, (1<<PrimerBit | 1<<UltimoBit)
  out PuertoSalida, RegAux
  call delay500mseg
  clr RegAux
  out PuertoSalida, RegAux
  call delay500mseg
  dec RegContador0
  jmp repetir

regresar:
  pop RegAux
reti


isr_int1:
  push RegContador0

  ldi RegContador0, 0x00
repetir_int1:
  out PuertoSalida, RegContador0
  call delay500mseg
  cpi RegContador0, ContadorMaximo
  breq final_int1
  inc RegContador0
  jmp repetir_int1

final_int1:
  pop RegContador0
reti
```