Universidad de Buenos Aires
Facultad De Ingeniería
Año 2021 – 1.er Cuatrimestre

(82.07) Laboratorio de microprocesadores

Entrega N.º 2: Entradas/Salidas e interrupciones externa s
Curso 01
25 de Mayo de 2020

Docentes a cargo:
Stola, Gerardo Luis
Salaya, Juan Guido
Cofman, Fernando

Integrante		Padrón		Correo electrónico
Lützelschwab, Nahila	—	100686	—	nlützelschwab@fi.uba.ar

Índice

1. Objetivos del proyecto

Los objetivos del presente trabajo práctico son controlar un display de 7 segmentos a partir de pulsadores, trabajando con las entradas tanto en modo lectura programada como con interrupciones. Se busca comprender las características DC del microcontrolador, analizando los consumos de corriente requeridos y disponibles, de acuerdo a las hojas de datos.

2. Descripción del proyecto

El presente trabajo consistió en diseñar un programa que permita visualizar los dígitos de un display 7 segmentos ánodo común mediante el uso de una placa Arduino UNO y mediante el manejo de pulsadores que permitan incrementar o decrementar los dígitos. El programa inicializa su valor en el dígito 5, el primer pulsador controla el cambio de dígito, mientras que el segundo permite incrementar el dígito en caso de estar pulsado y decrementarlo en caso contrario. En caso que el dígito llegue al mínimo o máximo posible (0 o 9), reinicia a su valor inicial.

2.1. Lista de componentes

A continuación se listan los componentes utilizados para la implementación del proyecto.

- Placa Arduino UNO
- Protoboard de 830 puntos
- Cables macho-macho para protoboard
- Dos pulsadores
- Siete resistencias de valor 330Ω
- Un display 7 segmentos ánodo común

3. Esquemático

Como muestra el esquemático 1, se hizo uso de una placa Arduino UNO basada en el microcontrolador ATmega328p, a la cual se le conectaron dos pulsadores (configurados como entradas) a los pines PD2 y PD3 que se vinculan con las interrupciones externas INT0 e INT1 respectivamente y se hizo uso de las resistencias pull-up internas del microcontrolador. A su vez se conectó un display 7 segmentos ánodo común, cada segmento conectado a una resistencia de 330Ω. Dado que el display es ánodo común (nodo común conectado a VCC), se tuvo en cuenta que para encender un segmento se setea un '0' lógico y para apagarlo un '1' lógico, formando los dígitos de 0 al 9 como lo muestra la tabla 1. Los segmentos se conectaron a los pines de los puertos C y D configurados como salida de la siguiente manera: a - PC0, b - PC1, c - PC2, d - PC3, e - PC4, f - PC5, g - PD6.

El primer pulsador (INT0) genera un cambio de dígito, incrementando en caso que el segundo pulsador (INT1) esté pulsado y decrementando en caso contrario. Al inicializarse el display muestra el dígito 5 y en caso que el dígito llegue al mínimo o máximo posible (0 o 9), reinicia a su valor inicial.

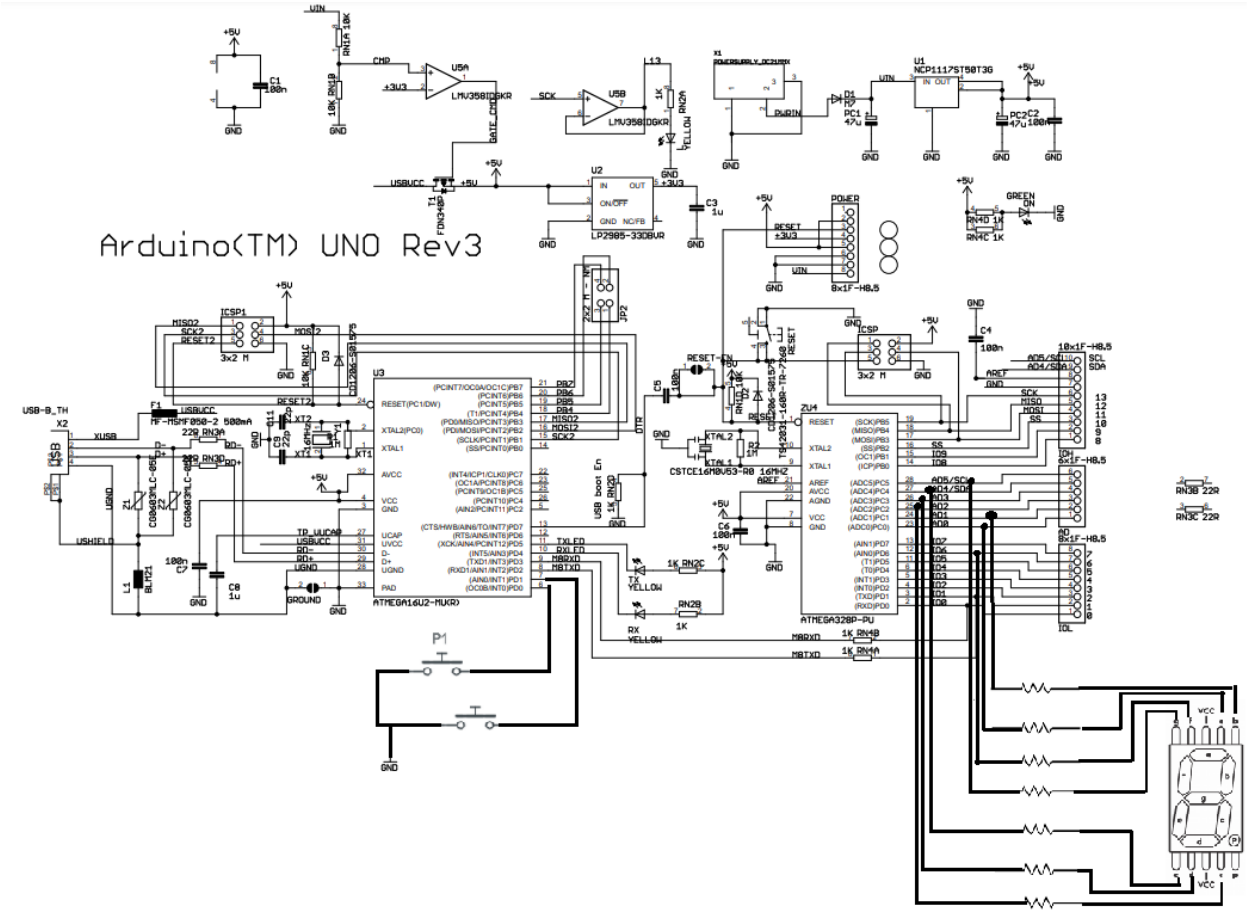

Figura 1: Esquematico completo del circuito implementado

Dígito	dp	PD6	PC5	PC4	PC3	PC2	PC1	PC0
0	1	1	0	0	0	0	0	0
1	1	1	1	1	1	0	0	1
2	1	0	1	0	0	1	0	0
3	1	0	1	1	0	0	0	0
4	1	0	0	1	1	0	0	1
5	1	0	0	1	0	0	1	0
6	1	0	0	0	0	0	1	0
7	1	1	1	1	1	0	0	0
8	1	0	0	0	0	0	0	0
9	1	0	0	1	0	0	0	0

Tabla 1: Valores lógicos de los pines necesarios para encender los segmentos formando los digitos 0 - 9

4. Software

Mediante el software Microchip Studio, se desarrolló el programa a implementar en código Assembly.

4.1. Diagrama de flujo

En la figura 2 se puede observar el diagrama de flujo correspondiente al software principal.

Para hacer uso de las interrupciones AVR, fue necesario utilizar un registro de control (EICRA), un registro que habilite el uso de la interrupción (EIMSK) y un registro que detecte cuando se ha producido la interrupcion externa (EIFR). Estos fueron configurados de manera tal que un flanco descendente ocurrido en el pin INT0 genere una interrupción y el INT1 se configuró como lectura explícita del valor instantáneo del pin.

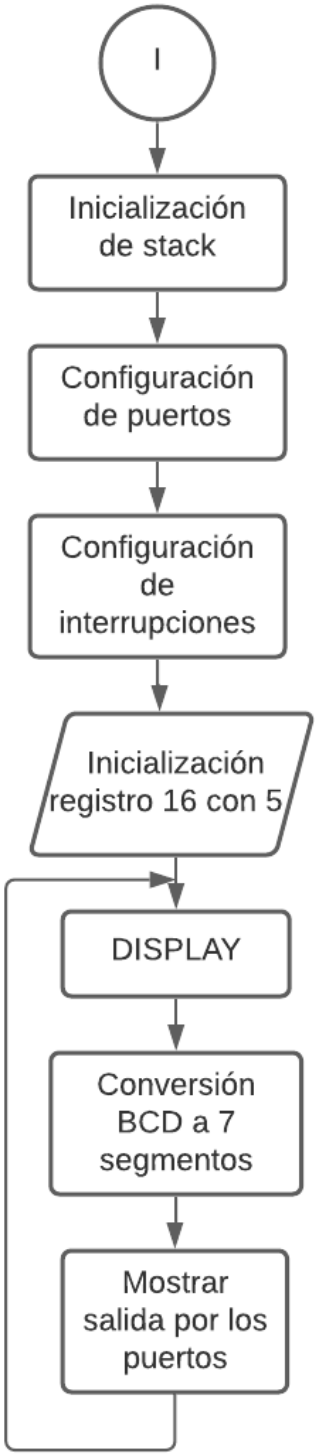

Figura 2: Diagrama de flujo

En la figura 3 se puede ver el diagrama de flujo correspondiente a la interrupción. Para desarrollar el algoritmo correspondiente, se tuvo en cuenta que la mecánica de los pulsadores genera ruido indeseado, por lo tanto para evitar que el display avance más de un dígito se desarrolló una rutina de retardo iterativa, llamada "DEBOUNCE_INTERRUPT", que permita determinar si es ruido o no. Otra manera de resolverlo es mediante hardware, incorporando al circuito un filtro RC, amplificador inversor o filp-flops , pero podrían ralentizar la respuesta del circuito.

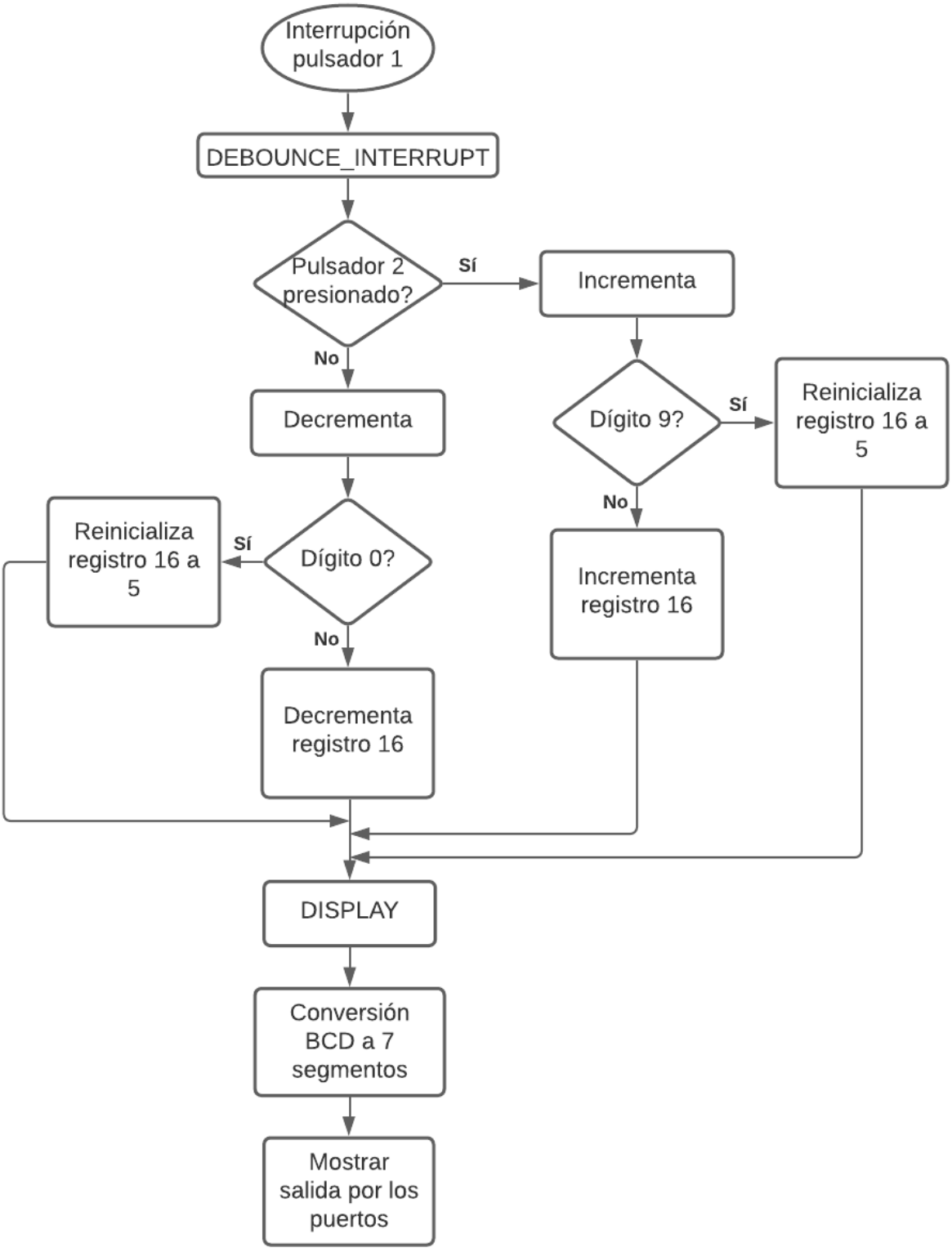

Figura 3: Diagrama de flujo de la interrupción

4.2. Diagrama en bloques

La figura 4 muestra el diagrama en bloques. Como se puede observar se conectó la placa Arduino UNO con una computadora mediante un cable USB como alimentación del circuito y para cargar el software correspondiente.

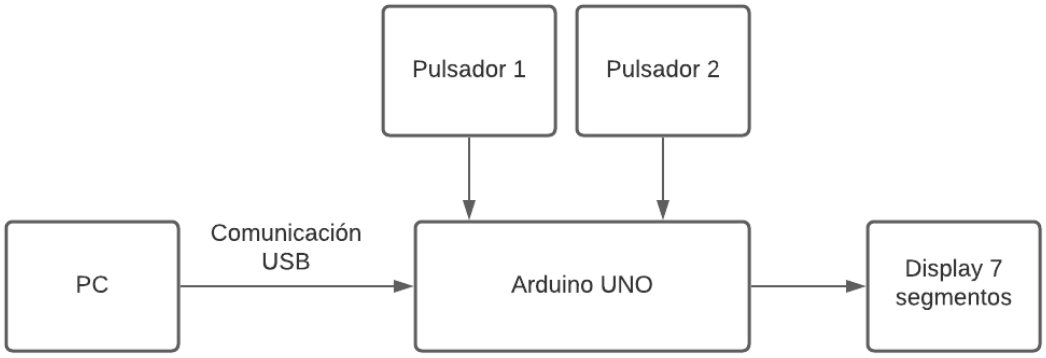

Figura 4: Diagrama en bloques

5. Preguntas

- ¿ Si en vez de colocar siete resistores se coloca uno solo en el nodo común, qué ocurre?

  Si se utilizara un único resistor conectado al nodo común del display de 7 segmentos en vez de siete, a medida que se encienden más cantidad de segmentos menor será la luminosidad. Por lo tanto, encendiendo N segmentos la corriente de cada led será $I_{LED} = \frac{I_{TOTAL}}{N}$, de esta manera el dígito 1 que se forma con sólo dos segmentos será el que mayor luminocidad tendrá, mientras que el dígito 8 tendrá la menor luminocidad.

- ¿ Cuánta corriente puede proveer cada PIN y un puerto completo? ¿Es la misma corriente que se provee en estado alto que en estado bajo ?

  De acuerdo a la hoja de datos del fabricante del microcontrolador, cada pin puede suministrar una corriente máxima de $40mA$ y cada puerto un máximo de $30mA$. A su vez, la suma de todas las corrientes de salida de los terminales C0 - C5, D0- D4 manteniendo el estado alto no debe exceder los $150mA$. La suma de todas las I_{OL} de los terminales C0 - C5 no debe exceder los $100mA$ y para los terminales D0 - D4 tampoco debe exceder los $100mA$.

  De acuerdo al circuito implementado que utiliza PC0-PC5 y PD6 como salida, conectados a los segmentos color rojo cuyo $V_{LED} = 2V$ y utlizando resistencias de 330Ω, se tiene una corriente de $9mA$ por segmento. Esto da como resultado una corriente máxima entregada por el microcontrolador de $63mA$, inferior a la cota máxima establecida, trabajando entonces en condiciones deseadas.

- ¿ Si en el programa se eliminase el manejo por la interrupción del pin PD2 y se quisiera conseguir, no obstante, que el programa siga funcionando de la misma forma, cómo lo modificaría?

  En dicho caso se podría utilizar algún otro pin disponible del microcontrolador dejando libre el pin INT0 que está reservado para realizar interrupciones externas AVR. De esta manera, se deberá verificar iterativamente el estado del pin. Esto es ineficiente, ya que en caso de que el microcontrolador este ejecutando otras instrucciones, puede no detectar un cambio de estado del pin, no generando la interrupcion deseada y perdiendo información. Es por ello que se utlizan las interrupciones externas, de manera que suspende lo que este realizando para atender las rutinas de alta prioridad.

6. Resultados

Se lograron los resultados deseados, estos se pueden observar en el video del siguiente link: https://youtu.be/EwDNH8

7. Conclusiones

Se puede concluir que este trabajo práctico permitió conocer el manejo de entradas en modo lectura programada y de interrupciones externas; y comprender las características DC del microcontrolador. Se puede decir que los resultados conseguidos fueron los deseados.

8. Anexo

8.1. Código fuente

```
;
; Laboratorio de microprocesadores (86.07)
; Nahila Lutzelschwab
; Padron : 100686
; 1er cuatrimestre 2021
; Turno Martes 19hs
;

.INCLUDE "m328pdef.inc"

.CSEG          ;Segmento de codigo

.EQU INITIALIZE = 0x05


.ORG 0
  RJMP START

.ORG INT0addr      ; Vector de INT0 (pin PD2)
  RJMP ISR_INT0


START:

; Inicia el stack en la parte alta de la memoria
  LDI R16, LOW(RAMEND)
  OUT SPL, R16
  LDI R16, HIGH(RAMEND)
  OUT SPH, R16


; Configura puertos
  LDI R17, (1<<DDD7)|(1<<DDD6)|(1<<DDD5)|(1<<DDD4)   ; Habilita el puerto D como entrada
                                                      ; en la parte baja y en la parte alta como salida
  OUT DDRD, R17

  LDI R17, (1<<PD3)|(1<<PD2)   ; Habilita las resistencias pull-up de los pulsadores
  OUT PORTD, R17

  LDI R17, 0xFF    ; Carga 11111111 en el registro R17
  OUT DDRC, R17    ; Configura el puerto C como salida

  LDI R17, 0x00
  OUT PORTC, R17

; Configura interrupciones
  LDI R18, (1<<ISC01)      ; Habilita los pines INT0 en modalidad flanco descendiente
  STS EICRA, R18

  LDI R18, (1<<INT0)       ; Habilita el uso de la interrupcion del pin INT0
  OUT EIMSK, R18

  SEI              ; Habilita las interrupciones globales

  LDI R16, INITIALIZE      ; Inicializa con valor 5

```

```
LOOP:              ; Mantiene el valor inicial hasta que se produzca interrupcion
  RCALL DISPLAY
  RJMP LOOP

DISPLAY:
  RCALL RESET_EIFR      ; Resetea los bits de interrupciones del registro EIFR para proximas
                        ;     interrupciones
  RCALL BCD_SEG        ; Muestra los segmentos en el display
  RET

; BCD a 7 segmentos
BCD_SEG:                ; Inicializo puntero Z al comienzo de la tabla
  LDI ZL, LOW (SEGMENTS << 1)        ; R30 apunta a la parte baja de la direccion
  LDI ZH, HIGH (SEGMENTS << 1)       ; R31 apunta a la parte alta de la direccion
  LDI R17, 0
  ADD ZL, R16               ; Le suma la posicion a  ZL
  ADC ZH, R17               ; Si la suma tiene carry se lo asigna a ZH
  LPM R17, Z                ; Carga lo apuntado por Z en R16
  RCALL SHOW_SEG              ; Muestra los segmentos en el display
  RET


; Muestra display
SHOW_SEG:
  OUT PORTC, R17         ; Carga lo apuntado por Z en PORTC
  SBRS R17, 6
  CBI PORTD, 6
  SBRC R17, 6
  SBI PORTD, 6
  RET

; Rutina de interrupcion INT0 (PD2)
ISR_INT0:
  LDI R23, 25
  RCALL DEBOUNCE_INTERRUPT  ; Retardo para evitar efecto de ruido transitorio del pulsador
  RCALL CONTROL_INT1       ; Se fija si esta presionado el interruptor PD3
  RETI

; Rutina de retardo para evitar efecto de ruido del pulsador
DEBOUNCE_INTERRUPT:
  PUSH R23
  RCALL DELAY          ; Retardo de 10ms para 16MHz
  POP R23
  SBIC PIND, 2         ; Se fija si se ha producido una interrupcion en el INT0 o era ruido
  RJMP RETURN
  DEC R23
  CPI R23, 0
  BRNE DEBOUNCE_INTERRUPT
  RET


RETURN:
  RETI

; Control del segundo pulsador (INT1/PD3)
CONTROL_INT1:
  PUSH R17
  IN R17, PIND
  SBRC R17, 3        ; Se fija si se ha pulsado PD3 (valor logico 0) incrementa, sino
    decrementa
  RCALL DECREMENT        ; Rutina de decremento
  SBRS R17, 3
  RCALL INCREMENT        ; Rutina de incremento
  POP R17
  RET


; Decrementa el digito en caso que no haya llegado al minimo
DECREMENT:
```

```
124   RCALL DEC_INDICATOR      ; Se fija si el digito es el minimo posible
125    ;LD R16, -Z             ; Decrementa el puntero
126   DEC R16
127   RET
128
129
130 ; Se fija si el digito es el minimo posible
131 DEC_INDICATOR:
132   CPI R16, 0
133   BREQ REINITIALIZE        ; Llama rutina que resete el digito a 5
134   RET
135
136
137 ; Incrementa el digito en caso que no haya llegado al maximo
138 INCREMENT:
139   RCALL INC_INDICATOR      ; Se fija si el digito es el maximo posible
140    ;LPM R16, Z+            ; Incrementa el puntero
141   INC R16m
142   RET
143
144
145 ; Se fija si el digito es el maximo posible
146 INC_INDICATOR:
147   CPI R16, 9
148   BREQ REINITIALIZE        ; Llama rutina que resete el digito a 5
149   RET
150
151 ; Resetea los bits de interrupciones del registro EIFR para proximas interrupciones
152 RESET_EIFR:
153   LDI R20, 0X00
154   OUT EIFR, R20
155   RET
156
157 ; Resetea el digito a 5
158 REINITIALIZE:
159   LDI R16, INITIALIZE
160   RJMP DISPLAY
161   RET
162
163 ; Delay de 8ms para el rebote del interruptor
164 DELAY:
165   LDI R18, 167
166     LDI R19, 59
167   LOOP1:
168     DEC  R19
169     BRNE LOOP1
170     DEC  R18
171     BRNE LOOP1
172     NOP
173   RET
174
175
176 ; PUERTO C, PC0->a, PC1->b, ..., PC5->f, PD6->g
177 ; Display 7 segmentos anodo comun -> enciende con '0'
178 .ORG 0X500       ; Vector de la tabla
179
180 SEGMENTS:
181 .db 0b11000000, 0b11111001,  0b10100100, 0b10110000
182 ;     0              1            2            3
183 .db 0b10011001 , 0b10010010, 0b10000010, 0b11111000
184 ;     4              5            6            7
185 .db 0b10000000, 0b10010000
186 ;     8              9
```

9. Bibliografía

- Mazidi, M. A., Naimi, S., Naimi, S. (2010). AVR Microcontroller and Embedded Systems: Using Assembly and C (Pearson Custom Electronics Technology). New Jersey, United States of America: Pearson
- ATMEGA328P Datasheet (PDF) - ATMEL.http://ww1.microchip.com/downloads/ Corporation. en/DeviceDoc/Atmel-7810-Automotive-Microcontrollers-ATmega328P$_{D}atasheet.pdf$