Fiuba
Facultad de Ingeniería
Universidad de Buenos Aires

Laboratorio de Microprocesadores (86.07)

Proyecto:

Trabajo Practico Nº2

Entradas / Salidas - Interrupciones Externas

Profesor:	Ing. Guillermo Campiglio
Cuatrimestre / Año:	1 er cuatrimestre 2021
Turno de clases prácticas:	Miércoles
Jefe de Trabajos Prácticos:	Ing. Pedro Ignacio Martos
Docente guía:	Ing. Fabricio Baglivo

Autores			**Seguimiento del proyecto**								
Nombre	**Apellido**	**Padrón**									
Gonzalo	Puy	99784									

Observaciones:

Fecha de aprobación		

Firma J.T.P

COLOQUIO	
Nota Final	
Firma Profesor	

Índice

1. Introducción

El presente trabajo tiene como objetivo controlar un display de 7 segmentos a partir de pulsadores. Para las entradas se trabajará tanto con lectura programada como con interrupciones. Se buscará comprender las características DC del microcontrolador, analizando los consumos de corriente requeridos y disponibles, de acuerdo a las hojas de datos.

2. Desarrollo

2.1. Banco de mediciones

Para la realización del trabajo se utilizará el siguiente banco de mediciones, compuesto por:

- Placa de desarrollo Arduino "UNO" y su respectivo cable para conectar la placa a la PC.
- El microcontrolador a usar, es el que viene integrado en la placa Arduino: `Atmega328P`.
- Display de 7 segmentos cátodo común `5611AH`.
- Protoboard
- Cables macho-macho para conexión del protoboard y de la placa Arduino.
- 7 resistencias de 330 Ω.
- 2 Pulsadores.

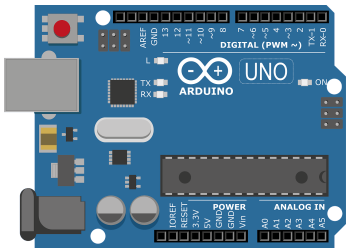

Figura 1: Placa de desarrollo Arduino UNO.

La conexión se muestra en las siguientes figuras

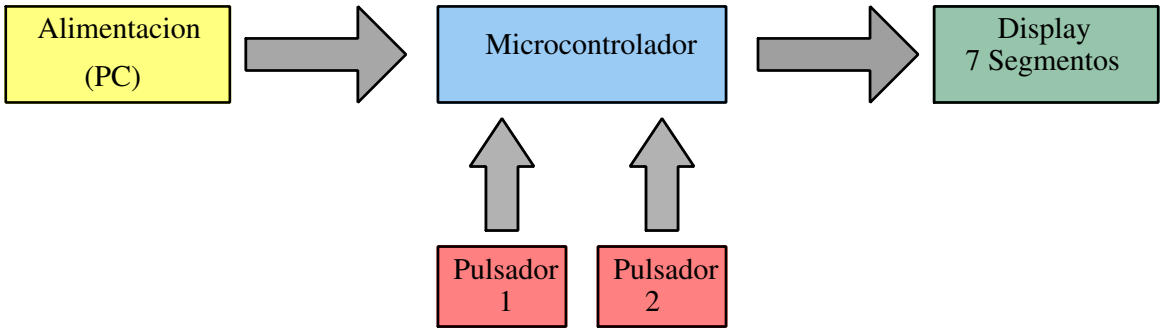

Figura 2: Diagrama de bloques

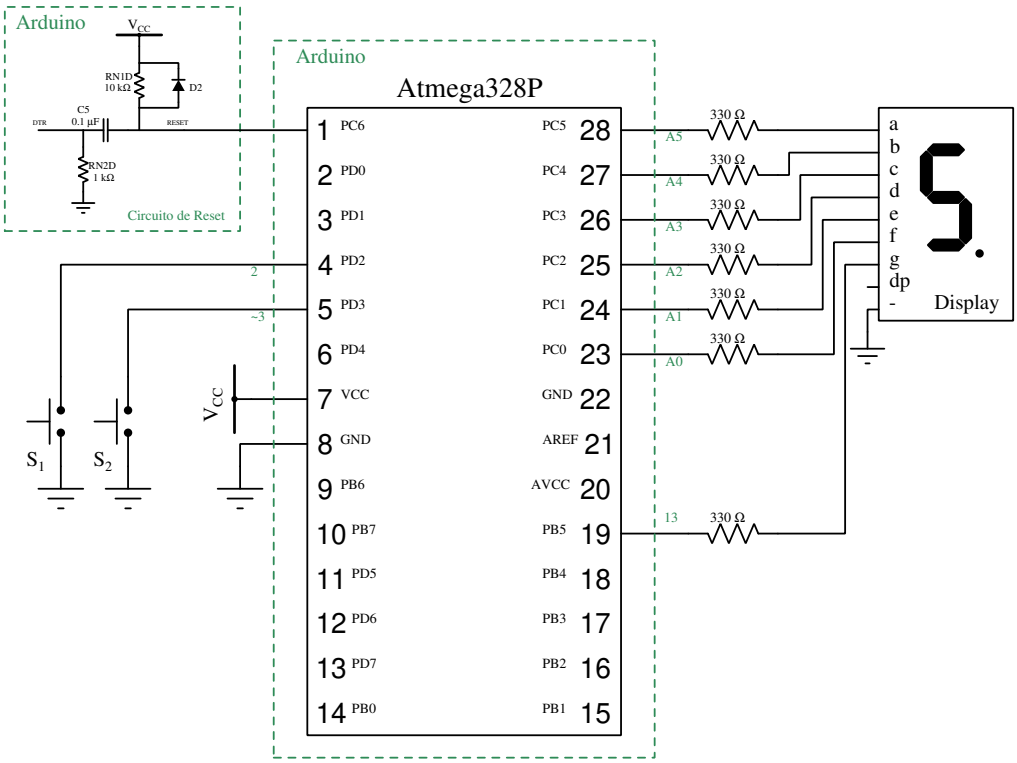

Figura 3: Conexión utilizada para el trabajo.

2.2. Primeros cálculos

Como la lista de los materiales fue provista por la cátedra, se procederá a verificar las resistencias elegidas para este trabajo.

Para comenzar, analizando la hoja de datos del microcontrolador, cada pin I/O puede dar una corriente de hasta 20 mA para $V_{CC} = 5\,\text{V}$.

Por otro lado, como el display de 7 segmentos es de cátodo común, se tiene una conexión que se puede ver simplificadamente por la siguiente figura.

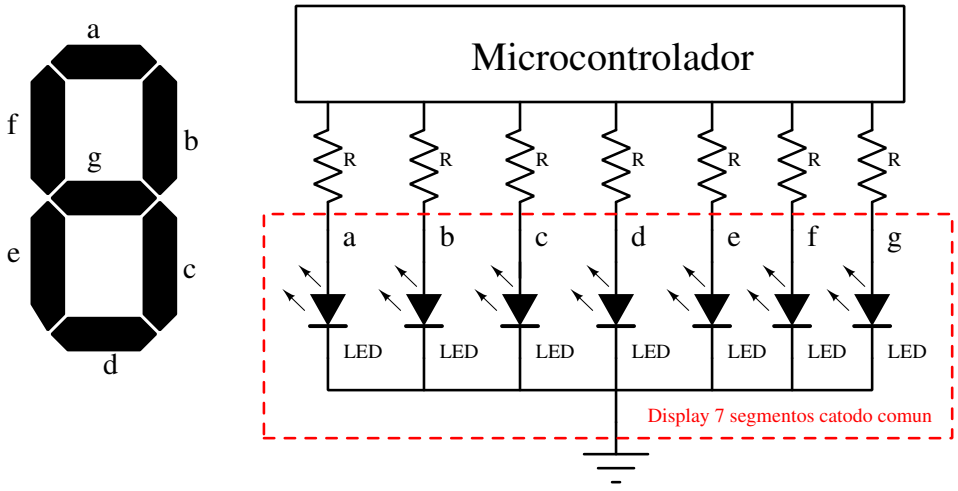

Figura 4: Modelo del display de 7 segmentos cátodo común.

Dada esta conexión, el LED se encenderá con un '1' logico. La tensión que otorga un '1' logico

en el pin de salida del microcontrolador V_{OH} es, según la hoja de datos, de aproximadamente 4,2 V.

Además, viendo la hoja de datos del display de 7 segmentos, se obtuvo un valor típico de la tensión en directa de los LEDS. Para este display en particular se tiene que $V_F = 1{,}8\,\text{V}$ ($@\, I_F = 10\,\text{mA}$).

Con los datos obtenidos de las hojas de datos podemos ver que que la corriente que pasa por cada uno de los LEDS, cuando se quiere encender, es aproximadamente

$$I_{LED} = \frac{4{,}2\,\text{V} - 1{,}8\,\text{V}}{330\,\Omega} = 7{,}27\,\text{mA}$$

Lo cual tiene sentido, ya que se busca una corriente de entre 5 mA y 10 mA, para que el LED tenga una buena luminosidad que no sea la máxima (la cual se da aproximadamente a 20 mA) para prolongar su vida útil. Además, como se menciono anteriormente, cada pin I/O del microcontrolador puede entregar hasta 40 mA. Por lo que el valor encontrado esta dentro del rango de valores posibles que puede entregar cada pin del microcontrolador.

Cuando se impone un '0' logico. EL LED queda en inversa, por lo tanto, no se encenderá.

2.3. Programa a implementar

La idea del programa es que el display inicialmente mostrará el dígito "5". Presionando el pulsador conectado a PD2 se ordenara un cambio de dígito y el pulsador PD3 proverá el sentido (incremento o decremento). Es decir que, si al presionar el pulsador conectado a PD2, el pulsador en PD3 también esta presionado, entonces se incrementará el dígito decimal. En cambio, si PD3 no esta presionado y se presiona PD2, se decrementará el valor decimal mostrado. Cuando se esta incrementando, al llegar al valor máximo (dígito "9"), si se ordena otro incremento el display se quedara en dicho valor máximo. Lo mismo sucede en el caso contrario con el valor minino (dígito "0").

El manejo de PD2 será usando interrupciones con modalidad de flanco descendente. Por otro lado, el manejo de PD3 será por lectura explícita del valor instantáneo del pin.

A continuación se presentará el diagrama de flujo del programa en la figura 5, con el cual se explica de forma simplificada el funcionamiento del programa.

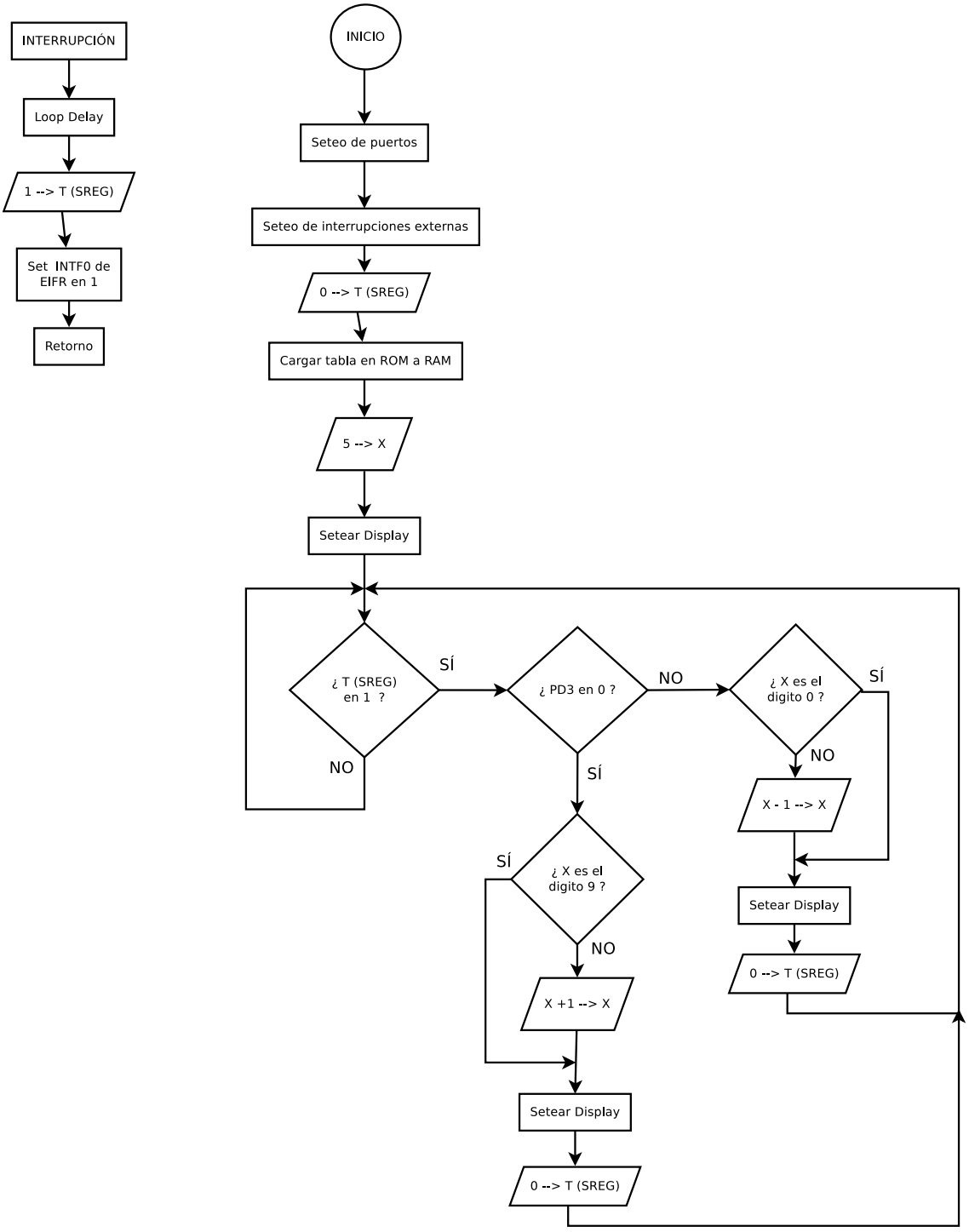

Figura 5: Diagrama de flujo del programa.

2.4. Problemas encontrados y soluciones

2.4.1. Conexión de los pulsadores

Según el esquemático de la conexión dado en la figura 3, se puede ver que los pulsadores S_1 y S_2 tienen una conexión de tipo *Pull-up*. Esta conexión fuerza en el pin un '1' logico al no estar los pulsadores presionados y un '0' logico cuando estos se presionan.

Como se puede notar en el esquema, en este caso, si los pulsadores no se presionan queda circuitalmente "flotando", es decir, no es ni un '1' ni un '0' lógico. Para evitar esto se puede

incorporar un resistor adicional para cada uno de los pulsadores. Dicho resistor se conectaría a V_{CC} y entre el pin y el pulsador. Sin embargo, el microcontrolador nos permite activar las resistencias de *Pull-up* en los pines correspondientes a los pulsadores (PD2 y PD3), por lo que tenemos la posibilidad de solucionar este problema por *software*, evitándose así el agregado de un componente externo adicional.

2.4.2. Valores de los siete segmentos del display

Según el enunciado provisto para este trabajo práctico, los valores de los siete segmentos de cada uno de los diez símbolos decimales deben estar declarados en una tabla almacenada en la memoria de programa (ROM).

Esto presento una dificultad a la hora de implementar las rutinas de incremento y decremento, ya que la instrucción para la lectura de datos indirectos (mediante punteros) de la memoria de programa (`LPM` o "*Load program memory*"), permite solamente una lectura y un post-incremento del puntero. Esto presento una complicación a la hora de el decremento de los dígitos, ya que se necesito volver a las posiciones anteriores de la tabla almacenada en ROM.

La solución implementada fue la de copiar la tabla que esta almacenada en ROM a la memoria SRAM. Con esto, es posible mediante la instrucción `LD` o "*Load indirect*", leer los datos con un pre-decremento del puntero. Debido a esto, se pudo avanzar secuencialmente para "atrás" y "adelante" en la tabla almacenada en SRAM, simplificando la tarea de incrementar y decrementar.

2.4.3. Problema de '*switch bounce*' o rebote del pulsador

Durante la realización del trabajo se noto que al presionar el pulsador S_1, se producía de vez en cuando un avance de mas de un dígito en el display. Esta circunstancia indeseada se llama "*bounce*" o "rebote" y esta vinculada al ruido originado en la mecánica del pulsador.

Básicamente, cuando se presiona el pulsador, aunque este haya sido presionado una sola vez, debido al rebote mecánico se producen falsas "pulsaciones". Esto hace que al terminar la rutina de interrupción del programa, el microcontrolador tome alguna de las "pulsaciones" extra como otra interrupción, por lo que se aumenta (o decrementa) un dígito nuevamente.

Dicho efecto también sucede al soltar el pulsador, pero debido a las características del programa, esto no presenta una dificultad.

Este problema puede solucionarse mediante el agregado de un capacitor en paralelo al switch, donde el circuito RC formado "filtrará" los rebotes.

A pesar de esto, en este trabajo se propuso una solución mediante "software". Donde se agrego un pequeño *loop* con un delay de 10 ms al principio la rutina de interrupción. En este *loop* se hace el hace el pequeño delay de 10 ms y luego se lee el valor del pin donde esta conectado el interruptor S_1. Si este valor es '1', significa que la interrupción que se produjo fue a causa de un rebote, por lo tanto se retorna de la rutina. En cambio, si el valor del pin es '0' entonces la interrupción se produjo efectivamente por presionar el pulsador S_1, entonces la rutina de interrupción continuara normalmente.

Además, antes de retornar mediante la instrucción `RETI`, se hace un *clear* del bit `INTF0` del registro `EIFR`. Esto hace que se ignoren las interrupciones que puedan haberse producido mientras la rutina de interrupción estaba ejecutándose.

2.5. Planteos extra

El enunciado del trabajo practico propone algunas preguntas extras. Estas preguntas fueron colocadas en esta sección del trabajo, y son las siguientes

1. ¿Si en vez de colocar siete resistores se coloca uno solo en el nodo común, qué ocurre?
2. ¿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?

1.

Esta pregunta esta relacionada con lo visto en la sección 2.2. Si se colocase un solo resistor en el nodo común, para que circule por los LEDS aproximadamente una corriente de entre 5 mA y 10 mA (como en el caso de este trabajo), se necesitaría un resistor de resistencia mas pequeña que 330 Ω. Estro provocará que la corriente que circule por este resistor sea de una magnitud elevada (aproximadamente 7 veces mayor a la que circula por cada LED). Esto parece tener sentido, solo si se supone que se encienden todos los LEDS. El problema se encuentra cuando se necesitan prender algunos si y otros no. En este caso, la corriente que circula por el nuevo resistor no se dividirá en partes iguales provocando en en alguno de los LEDS encendidos circule mas corriente que la deseada. Esto puede provocar que los LEDS se quemen y un mal funcionamiento del circuito en general.

Con este simple análisis realizado, se llega a la conclusión de que cambiar los siete resistores por uno solo conectado en común perjudica al funcionamiento del circuito. Por lo que no conviene ahorrar en los resistores de los LEDS en este caso.

2.

Para que el programa funcionase de la misma forma, eliminándose la interrupción del pin PD2, se propusieron 2 soluciones.

Para la primer solución, se asumió que se busca que el programa siga funcionando mediante interrupciones. En este caso, si eliminamos la interrupción del pin PD2, podemos simplemente conectar el pulsador a cualquiera de los pinchange (PCINT23:0), y seguir usando interrupciones. La única diferencia es que esta interrupción actúa sin modalidad de flanco, por lo tanto, se ejecuta la interrupción con cualquier cambio de estado del pin.

En cambio, paras la segunda solución, se asumió que la idea es que el programa funcione de la misma forma pero sin interrupción alguna. En este caso se puede hacer una lectura directa del pin PD2 como se realizo en el caso del pin PD3. La desventaja de esto, es que se pierde el flujo asincrónico provisto por las interrupciones.

2.6. Código del programa

A continuación, se adjuntará el código realizado. Y por último, se agrega un enlace al vídeo en el cual se muestra el resultado del trabajo funcionando.

```
1  ;
2  ; Macros.inc
3  ;
4  ;  Created: 5/6/2021 01:43:15
5  ;  Autor  : Puy Gonzalo
```

```
6  ;  Padron : 99784
7  ;
8
9  ; Esta macro inicializa el puntero Z a una posicion en ROM
10 ; Uso:      initZ   <Etiqueta>
11 .macro  initZ
12             ldi     Zh,HIGH(@0<<1)
13             ldi     Zl,LOW(@0<<1)
14 .endmacro
15
16 ; Esta macro inicializa el puntero X a una posicion en RAM
17 ; Uso:      initX   <Etiqueta>
18 .macro  initX
19             ldi     Xh,HIGH(@0)
20             ldi     Xl,LOW(@0)
21 .endmacro
22
23 ; Esta macro inicializa el stack pointer
24 ; Uso:      initSP
25 .macro  initSP
26             ldi dummyreg,LOW(RAMEND)
27             out spl,dummyreg
28             ldi dummyreg,HIGH(RAMEND)
29             out sph,dummyreg
30 .endmacro
31
32 ; Esta macro: setea el puerto C como salida, el puerto PB5 como
      salida
33 ;el puerto D como entrada
34 ;activa las resistencias de pull-up en los puertos D
      correspondientes
35 ;Uso:   setPorts
36 .macro  setPorts
37         ldi     dummyreg,0x7F
38         out     DDRC,dummyreg              ;PC0-PC5 como salida
39         sbi     DDRB,5                     ;PB5 como salida
40         ldi     dummyreg,0x00
41         out     DDRD,dummyreg              ;Puerto D como
      entrada
42         sbi     PORTD,2                    ; Activo las
      resistencias de Pull-up en el puerto PD2 y PD3
43         sbi     PORTD,3
44 .endmacro
45
46 ; Esta macro setea la interrupcion externa en INT0
47 ; Uso: setINT0
48 .macro setInt0
49         ldi     dummyreg,EICRA_MASK
50         sts     EICRA,dummyreg
51         ldi     dummyreg,EIMSK_MASK
52         out     EIMSK,dummyreg
53         sei
```

```
54 .endmacro
55
56 ; Esta macro shiftea un registro 5 veces hacia la izquierda
57 ; Uso: shift5BitsL  <Rd>
58 .macro  shift5BitsL
59         swap     @0
60         andi     @0,0xF0
61         lsl      @0
62 .endmacro
63
64 ; Esta macro decrementa el registro que se pasa como argumento 1
       y si llega a cero,
65 ; salta a la etiqueta que se pasa como argumento 2
66 ; Uso:      djnz     <Rd>,<Etiqueta>
67 .macro  djnz
68             dec      @0
69             tst      @0
70             brne     @1
71 .endmacro
```

Listing 1: Archivo '.inc' utilizado para las macros usadas en el código

```
1  ;
2  ; TP2.asm
3  ;
4  ; Created: 4/6/2021 20:57:45
5  ; Autor : Puy Gonzalo
6  ; Padron : 99784
7  ;
8  ; Correcion del codigo del TP2
9
10 .include "m328pdef.inc"
11 .include "Macros.inc"
12
13 ; .equ y .def necesarios para el codigo
14 .equ     NUEVE = 0x7B
15 .equ     CERO = 0x7E
16 .equ     POS_INICIAL = 6     ;Define la posicion inicial del
       display. Si quiero que empiece en el digito 'n' POS_INICIAL =
       n+1
17 .equ     LEN_TABLA = 10
18 .equ     BYTES_TABLA_RAM = 10
19 .equ     PIN_S1 = 2
20 .equ     PIN_S2 = 3
21 .equ     C_MASK = 0b01111110
22 .equ     B_MASK = 0b00000001
23 .equ     EICRA_MASK = 0b0010
24 .equ     EIMSK_MASK = 0b01
25
26
27 .def     dummyreg = r16
28 .def     aux1 = r17
```

```
29 .def     aux2 = r18
30 .def     contador = r19
31
32 .dseg
33 .org SRAM_START
34 DISPLAY_NUMEROS_RAM: .byte BYTES_TABLA_RAM
35
36
37 .cseg
38 .org 0x0000
39         jmp     config
40
41 .org INT0addr
42         jmp     isr_int0
43
44 .org INT_VECTORS_SIZE
45
46 config:
47 ; Inicializo el stack pointer
48         initSP
49 ; Seteo los puertos de entrada y salida. Ademas activo las
     resistencias de pull-up correspondientes
50         setPorts
51 ; Seteo interrupciones externas
52         setINT0
53
54 main:
55         call    copiar_a_RAM
56         call    inicio
57         clr     dummyreg
58         clt
59
60 loop:
61         brts    cambio_digito
62         jmp     loop
63
64 ;Copio la tabla ROM en la memora SRAM
65 ;Con esto tengo mejor control sobre los datos
66 copiar_a_RAM:
67             initZ   DISPLAY_NUMEROS_ROM
68             initX   DISPLAY_NUMEROS_RAM
69             ldi     contador,LEN_TABLA
70 copy:       lpm     dummyreg,z+
71             st      x+,dummyreg
72             djnz    contador,copy
73             ret
74
75 inicio:
76         initX   DISPLAY_NUMEROS_RAM         ;Vuevlo a iniciar el
     puntero X
77         clr     dummyreg                    ;ya que despues de
     copiar a RAM quedo en la ultima posicion de la tabla
```

```
78          ldi     contador,POS_INICIAL
79
80  posicion_inicial:                       ;Seteo la posicion
    inicial del display ('5')
81          ld      dummyreg,x+
82          djnz    contador,posicion_inicial
83          call    setear_display
84          ld      dummyreg,-x             ;Con esto me aseguro
    de dejar el puntero X apuntando en la direccion de 5
85          ret
86
87  ;Rutina para setear el display con el numero que esta en dummyreg
88  setear_display:
89          mov     aux1,dummyreg
90          mov     aux2,dummyreg
91
92          andi    aux1,C_MASK
93          lsr     aux1
94          out     PORTC,aux1
95
96          andi    aux2,B_MASK
97          shift5BitsL     aux2
98          out     PORTB,aux2
99          ret
100
101 ;Rutina para el incremento o decremento del digito.
102 cambio_digito:
103         sbis    PIND,PIN_S2
104         jmp     incrementar
105         jmp     decrementar
106
107 incrementar:
108         ld      dummyreg,x
109         cpi     dummyreg,NUEVE
110         breq    fin_incremento  ;Si el valor que lei es el numero
    '9' -> Voy hacia fin_incremento
111         ld      dummyreg,x+     ;Hago una lectura solo para
    avanzar al suiguiente valor
112         ld      dummyreg,x      ;Leo el valor y dejo el puntero
    en esta direccion.
113
114         call    setear_display
115         jmp     retorno
116 fin_incremento:
117         call    setear_display
118         jmp     retorno
119
120 decrementar:
121         ld      dummyreg,x
122         cpi     dummyreg,CERO
123         breq    fin_decremento  ;Si el valor que lei es '0' ->
    Voy hacia fin_decremento
```

```
124
125         ld      dummyreg,-x     ;Decremento el puntero y leo el
                valor
126
127         call    setear_display
128         jmp     retorno
129 fin_decremento:
130         call    setear_display
131         jmp     retorno
132 retorno:
133         clt                     ;Limpio el flag T
134         jmp     loop            ;Salto a Loop
135
136 ;Rutina de interrupciones
137
138 isr_int0:
139
140 bucle_delay:
141         nop
142         call    delay
143         nop
144         sbic    PIND,PIN_S1
145         jmp     retorno_isr
146
147 activar_cambio_digito:
148         set                     ;Seteo el bit T del SREG
149
150 ;Seteo en 1 el bit 0 de EIFR (INTF0) antes de RETI
151 ;Con esto, me aseguro de ignorar las interrupciones
152 ;que pueden haberse dado mientras estaba en esta rutina.
153 retorno_isr:
154         sbi     EIFR,0
155         reti
156
157 ;Delay de 10 ms
158 delay:
159     ldi  r21, 208
160     ldi  r22, 202
161 L1:
162     dec  r22
163     brne L1
164     dec  r21
165     brne L1
166     nop
167     ret
168
169 ;Tabla ROM con los valores de los numeros para el display de 7
        segmentos
170 .cseg
171 .org 0x500
172 DISPLAY_NUMEROS_ROM: .db 0x7E,0x30,0x6D,0x79,0x33,0x5B,0X5F,0x70
        ,0x7F,0x7B
```

Listing 2: Código del programa

3. Conclusiones

En conclusión, se puede aclarar que esta experiencia permitió ver el funcionamiento de las interrupciones externas, las cuales presentan una clara ventaja cuando se requiere un accionar externo, como por ejemplo, un pulsador para realizar una determinada acción. Permitiendo un flujo asincrónico del código.

Por otro lado, también se pudo aprender mas sobre las características DC del microcontrolador y además, se obtuvo una idea clara de las posibilidades que se tienen a la hora de solucionar los problemas que se nos puedan presentar externamente o circuitalmente, mediante *software*. Si bien, a veces las soluciones no suelen ser las mas óptimas, o más elegantes, es importante tener en cuenta estas opciones que pueden, entre otras cosas, lograr el ahorro de componentes.