FACULTAD DE INGENIERIA
Universidad de Buenos Aires

(6609) LABORATORIO DE MICROCOMPUTADORAS

Proyecto:
TP2 - Interrupciones externas y Memoria EEPROM

Profesor:	**Ing. Guillermo Campiglio**
Cuatrimestre / Año:	1er cuatrimestre - 2022
Turno de clases prácticas:	Jueves - 19 a 22
Jefe de Trabajos Prácticos:	Graciela Ratto
Docente guía:	Gavinowich Gabriel

Autores			**Seguimiento del proyecto**								
Nombre	**Apellido**	**Padrón**									
Violeta	Perez Andrade	101456									

Observaciones:

Fecha de aprobación

Firma J.T.P.

COLOQUIO	
Nota final	
Firma Profesor	

Índice

1. Objetivo del proyecto

El objetivo del presente trabajo es familiarizarse con el manejo tanto de interrupciones externas como de la memoria EEPROM.Para esto se realizó un programa que lee una tabla de la EEPROM(o la carga en caso de no estar) y muestra el dato leído en un display de segmentos.

2. Descripción del proyecto

En este trabajo práctico se desarrolló un programa en lenguaje ensamblador. El mismo, se encarga de mostrar en un display los valores leídos en una tabla ubicada en la EEPROM, avanzando o retrocediendo a lo largo de la misma en base al pulsador que se presione. Para ello, a diferencia del TP pasado se utilizaron interrupciones en vez de estar constantemente chequeando el estado del pulsador.

3. Diagrama de conexiones en bloques

En el siguiente diagrama se puede observar cómo se conectaron los distintos bloques del trabajo

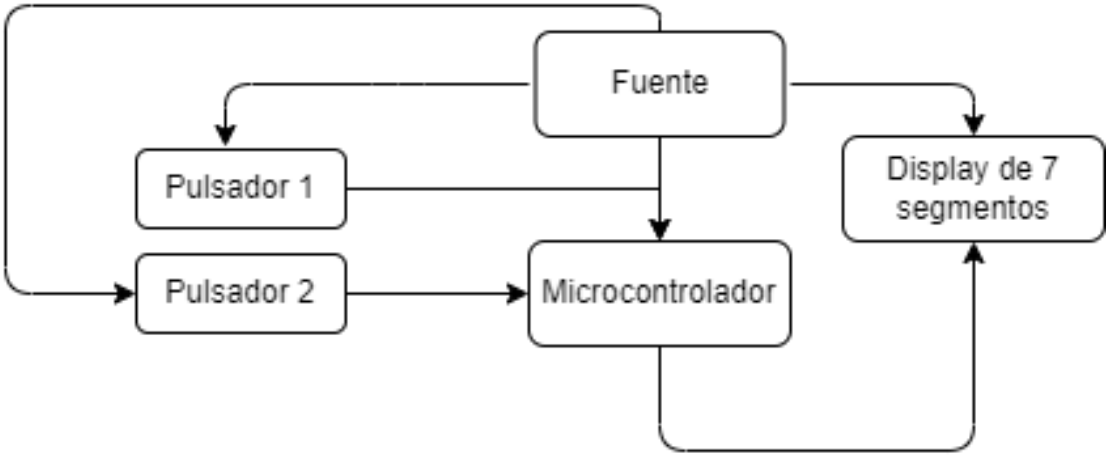

Figura 1: Diagrama de conexión en bloques

4. Circuito esquemático

Se incluye el diagrama esquemático del trabajo

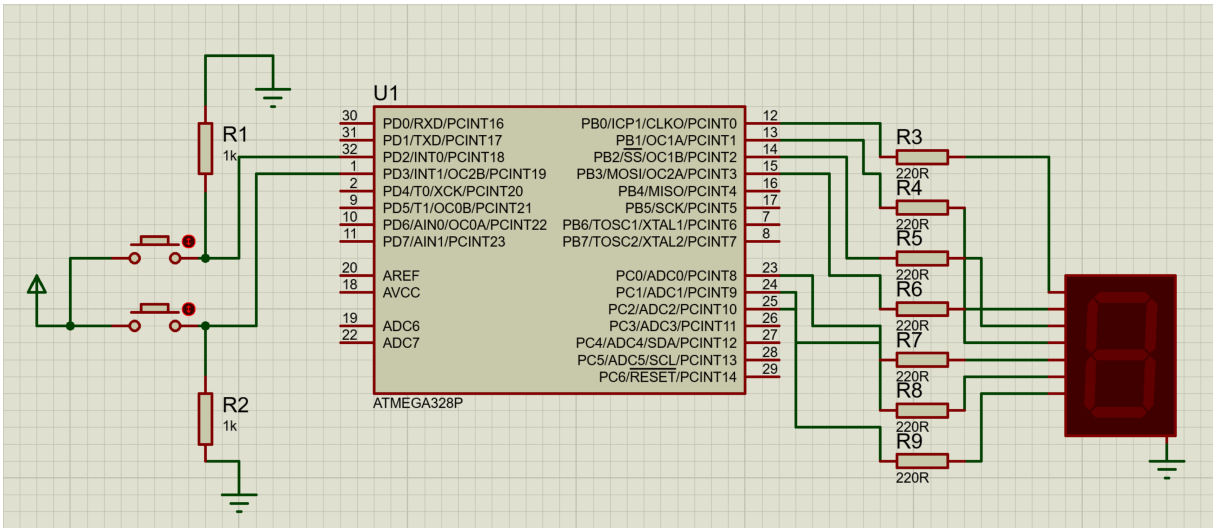

Figura 2: Circuito esquemático

5. Listado de componentes y tabla de gastos

Listado de componentes		
Componente	Cantidad	Precio uni.
Display 7 segemtnos	1	$ 70,91
Touch Switch 5mm	2	$ 22, 28
Resistencia 220 Ω	7	$ 35,52
Resistencia 10 Ω	2	$ 16,30
Atmega 328P	1	$ 878.90
Tira de 40 cables	1	$ 727, 91
Total		$ 2003,52

6. Diagrama de flujo

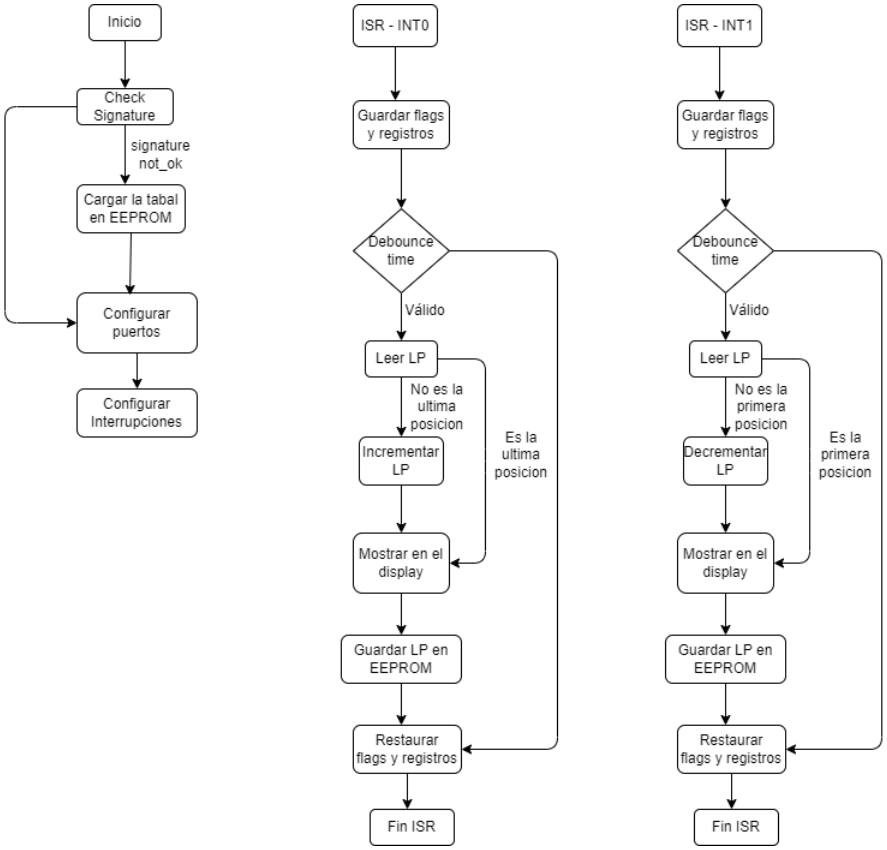

Figura 3: Diagrama de flujo

7. Resultados

Los resultados fueron los esperados, como se desarrolló en el código, se muestran correctamente los números leídos y al presionar los pulsadores sucede lo esperado

8. Conclusiones

En el presente trabajo se aprendió a utilizar las interrupciones del microcontrolador para detectar entradas sin tener que verificar manualmente. A su vez, se aprendió también sobre la memoria EEPROM, como funciona la misma y como es el mecanismo para leer y escribir datos desde y hacia la misma, que es diferente a la memoria RAM.
Por ultimo, se observó también que al utilizar interrupciones para detectar entradas provenientes de pulsadores todavía sigue siendo necesario contar con un retardo antirrebote dentro de la rutina de interrupción, aún cuando la rutina de interrupción se ejecute durante más tiempo que el rebote del pulsador. Esto se debe a que el microcontrolador es capaz de detectar una interrupción mientras otra está siendo atendida.

9. Apéndice: Código del programa

```
;
; TP2b.asm
; Author : Violeta
;


; Replace with your application code
.include "m328Pdef.inc"
;Inicio del codigo
.org 0x0000
     rjmp inicio
.org INT0Addr
     rjmp isr_int0
.org INT1Addr
     rjmp isr_int1

.org INT_VECTORS_SIZE

inicio:

; Se inicializa el Stack Pointer al final de la RAM utilizando la definicion global
; RAMEND
     ldi       r16,HIGH(RAMEND)
     out       sph,r16
     ldi       r16,LOW(RAMEND)
     out       spl,r16.

main_loop:
     ;rcall    load_table_in_eeprom
     rcall     check_signature
     brtc table_ok
     rcall     load_table_in_eeprom
table_ok:
     rcall     configure_ports
     rcall     configurar_interrupciones
     sei
     here:     jmp here

check_signature:
     push r17
     push r18
     push r19
     push r24
     ldi       r19, 3
     bst       r19, 5
     ldi       r24, 3
     ldi       r17, 0x20
     ldi       r18, 0x21
     ldi       xl, 0x0
     ldi       xh, 0x0
     rcall     read_eeprom ;X y r20
     cpse r20, r17
     dec       r19
     inc       xl
     rcall     read_eeprom
     cpse r20, r18
     dec       r19
     cpse r19, r24
     bst       r24, 1
     pop  r24
     pop  r19
     pop  r18
     pop  r17
     ret
```

```
65
66  load_table_in_eeprom:
67  ;tabla â 0 ,2,4,6,8,A,C,E,F,D,B,9,7,5,3,1,0,F,0,A,8"
68       ldi        xl, 0x0
69       ldi        xh, 0x0
70       ldi        r20, 0x20
71       rcall      write_eeprom ;r20
72       inc        xl
73       ldi        r20, 0x21
74       rcall      write_eeprom
75       inc        xl
76       ldi        r20, 0x0
77       rcall      write_eeprom
78       inc        xl
79       ldi        r20, 0x2
80       rcall      write_eeprom
81       inc        xl
82       ldi        r20, 0x4
83       rcall      write_eeprom
84       inc        xl
85       ldi        r20, 0x6
86       rcall      write_eeprom
87       inc        xl
88       ldi        r20, 0x8
89       rcall      write_eeprom
90       inc        xl
91       ldi        r20, 0xA
92       rcall      write_eeprom
93       inc        xl
94       ldi        r20, 0xC
95       rcall      write_eeprom
96       inc        xl
97       ldi        r20, 0xE
98       rcall      write_eeprom
99       inc        xl
100      ldi        r20, 0xF
101      rcall      write_eeprom
102      inc        xl
103      ldi        r20, 0xD
104      rcall      write_eeprom
105      inc        xl
106      ldi        r20, 0xB
107      rcall      write_eeprom
108      inc        xl
109      ldi        r20, 0x9
110      rcall      write_eeprom
111      inc        xl
112      ldi        r20, 0x7
113      rcall      write_eeprom
114      inc        xl
115      ldi        r20, 0x5
116      rcall      write_eeprom
117      inc        xl
118      ldi        r20, 0x3
119      rcall      write_eeprom
120      inc        xl
121      ldi        r20, 0x1
122      rcall      write_eeprom
123      inc        xl
124      ldi        r20, 0x0
125      rcall      write_eeprom
126      inc        xl
127      ldi        r20, 0xF
128      rcall      write_eeprom
129      inc        xl
130      ldi        r20, 0x0
```

```
131       rcall     write_eeprom
132       inc       xl
133       ldi       r20, 0xA
134       rcall     write_eeprom
135       inc       xl
136       ldi       r20, 0x8
137       rcall     write_eeprom
138       inc       xl
139       ldi       r20, 0x2
140       rcall     write_eeprom
141       ret
142
143  configure_ports:
144       push r20
145       push r21
146       ldi r21, 0x0
147       ldi  r20, 0xFF
148       out  DDRC, r20
149       out  DDRB, r20
150       out  DDRD, r21
151       pop r21
152       pop r20
153       ret
154
155  configurar_interrupciones:
156       ; INT0 e INT1 responden al flanco ascendente
157       ldi r16, (1 << ISC11) | (1 << ISC10) | (1 << ISC01) | (1 << ISC00)
158       sts EICRA, r16
159       ; Activar interrupciones para INT0 e INT1
160       ldi r16, (1 << INT1) | (1 << INT0)
161       out EIMSK, r16
162       ret
163
164  display_number:
165       push r4
166       push r18
167       push r17
168
169       ldi  zl,LOW(TABLA<<1)     ;ZL = 0x00 (low byte of address)
170       ldi  zh,HIGH(TABLA<<1)    ;ZH = 0x05 (high byte of address)
171
172       l1:
173            inc zl
174            dec r17
175            brne l1
176
177       lpm  r17,z
178       mov r4, r17
179       mov r18, r17
180       andi r18, 0b00001111
181
182       lsr  r4
183       lsr  r4
184       lsr  r4
185       lsr  r4
186
187       out  PORTB, r4
188       out PORTC, r18
189
190       pop r17
191       pop r18
192       pop r4
193       ret
194
195  read_eeprom:
196       ; Wait for completion of previous write
```

```
197       sbic EECR,EEPE
198       rjmp read_eeprom
199       ; Set up address X in address register
200       out EEARH, XH
201       out EEARL, XL
202       ; Start eeprom read by writing EERE
203       sbi EECR,EERE
204       ; Read data from Data Register
205       in r20,EEDR
206       ret
207
208 write_eeprom:
209       ; Wait for completion of previous write
210       sbic EECR,EEPE
211       rjmp write_eeprom
212       ; Set up address X in address register
213       out EEARH, XH
214       out EEARL, XL
215       ; Write data (r16) to Data Register
216       out EEDR,r20
217       ; Write logical one to EEMPE
218       sbi EECR,EEMPE
219       ; Start eeprom write by setting EEPE
220       sbi EECR,EEPE
221       ret
222
223 isr_int0: ; Pulsador 1
224       push r24
225       push r22
226       push r21
227       push r17
228       in r24, SREG  ; Guardar registro de estado
229       call delay50ms
230       sbis PIND, 2  ; antirebote
231       rjmp isr_int0_fin
232       ldi       xl, 0x17 ;pos de LP
233       ldi       xh, 0x0
234       rcall     read_eeprom
235       cpi  r20, 0x16 ;si ya llegue al fin de la tabla salgo
236       brge draw_isr_int0
237       inc  r20
238 draw_isr_int0:
239       mov  r21, r20 ; me guardo lo que voy a dejar en lp
240       mov  xl, r20 ;cargo en xl el indice del prox dato(que quiero dibujar)
241       rcall     read_eeprom
242       mov  r17, r20 ;dejo en r17 lo que quiero dibujar para llamar a display number
243       rcall display_number
244       ;ahora quiero dejar en lp el indice del dato dibujado (r21)
245       ldi  xl, 0x17
246       mov  r20, r21
247       rcall write_eeprom
248
249 isr_int0_fin:
250       out SREG, r24  ; Restaurar registro de estado
251       pop r17
252       pop  r21
253       pop  r22
254       pop r24
255
256       reti
257
258 isr_int1: ; Pulsador 2
259       push r24
260       push r22
261       push r21
262       push r17
```

```
263       in r24, SREG  ; Guardar registro de estado
264       call delay50ms
265       sbis PIND, 3  ; antirebote
266       rjmp isr_int1_fin
267       ldi       xl, 0x17 ;pos de LP
268       ldi       xh, 0x0
269       rcall     read_eeprom
270       cpi  r20, 0x3 ;si estoy al comienzo de la tabla salgo
271       brlt draw_isr_int1
272       dec  r20
273  draw_isr_int1:
274       mov  r21, r20 ; me guardo lo que voy a dejar en lp
275       mov  xl, r20 ;cargo en xl el indice del prox dato(que quiero dibujar)
276       rcall     read_eeprom
277       mov  r17, r20 ;dejo en r17 lo que quiero dibujar para llamar a display number
278       rcall display_number
279       ;ahora quiero dejar en lp el indice del dato dibujado (r21)
280       ldi  xl, 0x17
281       mov  r20, r21
282       rcall write_eeprom
283
284  isr_int1_fin:
285       out SREG, r24  ; Restaurar registro de estado
286       pop r17
287       pop  r21
288       pop  r22
289       pop r24
290
291       reti
292
293  delay50ms:
294    ldi r19, 4
295    loop0:
296      ldi r18, 201
297      loop1:
298         ldi r17, 248
299         loop2:
300             nop
301              dec r17
302              brne loop2
303         dec r18
304         brne loop1
305       dec r19
306       brne loop0
307    ret
308
309  TABLA:    .DB  243, \
310                 96, \
311                 181, \
312                 244, \
313                 102, \
314                 214, \
315                 215, \
316                 112, \
317                 247, \
318                 246, \
319                 119, \
320                 199, \
321                 147, \
322                 229, \
323                 151, \
324                 23
325
326  ;DATA_0:  .DB 0b11110011
327  ;DATA_1:  .DB 0b01100000
328  ;DATA_2:  .DB 0b10110001
```

```
;DATA_3:  .DB 0b11110100
;DATA_4:  .DB  0b01100110
;DATA_5:  .DB  0b11010110
;DATA_6:  .DB  0b11010111
;DATA_7:  .DB  0b01110100
;DATA_8:  .DB  0b11110111
;DATA_9:  .DB  0b11110110
;DATA_A:  .DB  0b01110111
;DATA_B:  .DB  0b11110111
;DATA_C:  .DB  0b10010011
;DATA_D:  .DB  0b11110011
;DATA_E:  .DB  0b10010111
;DATA_F:  .DB  0b00010111
```